aws

Guide du développeur

Deadline Cloud

Deadline Cloud: Guide du développeur

Table of Contents

Qu'est-ce que AWS Deadline Cloud ?

AWS Deadline Cloud est un AWS service entièrement géré qui vous permet de disposer d'une ferme de traitement évolutive opérationnelle en quelques minutes. Il fournit une console d'administration pour gérer les utilisateurs, les fermes, les files d'attente pour la planification des tâches et les flottes de travailleurs chargés du traitement.

Ce guide du développeur s'adresse aux développeurs de pipelines, d'outils et d'applications dans un large éventail de cas d'utilisation, notamment les suivants :

- Les développeurs de pipelines et les directeurs techniques peuvent intégrer Deadline Cloud APIs et ses fonctionnalités dans leurs pipelines de production personnalisés.
- Les fournisseurs de logiciels indépendants peuvent intégrer Deadline Cloud à leurs applications, ce qui permet aux créateurs de contenu numérique et aux utilisateurs de soumettre des travaux de rendu Deadline Cloud de manière fluide depuis leur poste de travail.
- Les développeurs de services Web et basés sur le cloud peuvent intégrer le rendu de Deadline Cloud à leurs plateformes, ce qui permet aux clients de fournir des ressources pour visualiser virtuellement les produits.

Nous fournissons des outils qui vous permettent de travailler directement à n'importe quelle étape de votre pipeline :

- Interface de ligne de commande que vous pouvez utiliser directement ou à partir de scripts.
- Le AWS SDK pour 11 langages de programmation populaires.
- Interface Web basée sur REST que vous pouvez appeler depuis vos applications.

Vous pouvez également en utiliser d'autres Services AWS dans vos applications personnalisées. Par exemple, vous pouvez utiliser :

- AWS CloudFormationpour automatiser la création et la suppression de fermes, de files d'attente et de flottes.
- Amazon va CloudWatch recueillir des statistiques relatives aux offres d'emploi.
- Amazon Simple Storage Service pour stocker et gérer les actifs numériques et les résultats des tâches.
- AWS IAM Identity Centerpour gérer les utilisateurs et les groupes de vos fermes.

Description du poste ouvert

Deadline Cloud utilise la spécification Open Job Description (OpenJD) pour spécifier les détails d'une tâche. OpenJD a été développé pour définir des tâches portables entre les solutions. Vous l'utilisez pour définir une tâche qui est un ensemble de commandes exécutées sur des hôtes de travail.

Vous pouvez créer un modèle de tâche OpenJD à l'aide d'un émetteur fourni par Deadline Cloud, ou vous pouvez utiliser n'importe quel outil pour créer le modèle. Après avoir créé le modèle, vous l'envoyez à Deadline Cloud. Si vous utilisez un émetteur, il se charge d'envoyer le modèle. Si vous avez créé le modèle d'une autre manière, vous pouvez appeler une action de ligne de commande de Deadline Cloud, ou vous pouvez utiliser l'une des actions AWS SDKs pour envoyer le travail. Dans tous les cas, Deadline Cloud ajoute le travail à la file d'attente spécifiée et planifie le travail.

Concepts et terminologie pour Deadline Cloud

Pour vous aider à démarrer avec AWS Deadline Cloud, cette rubrique explique certains de ses principaux concepts et termes.

Ressources agricoles

Ce diagramme montre comment les ressources de la ferme Deadline Cloud fonctionnent ensemble.

Farm

Un parc contient toutes les autres ressources liées à la soumission et à l'exécution de tâches. Les fermes sont indépendantes les unes des autres, ce qui les rend utiles pour séparer les environnements de production.

File d’attente

Une file d'attente contient les tâches à planifier sur les flottes associées. Les utilisateurs peuvent soumettre des tâches à une file d'attente et gérer leur priorité et leur statut dans la file d'attente. Une file d'attente doit être associée à une flotte associée à une file d'attente et à une flotte pour que ses tâches puissent être exécutées, et les files d'attente peuvent être associées à plusieurs flottes.

Flotte

Un parc contient la capacité de calcul nécessaire à l'exécution de tâches. Les flottes peuvent être gérées par le service ou par le client. Les flottes gérées par des services fonctionnent

dans Deadline Cloud et incluent des fonctionnalités intégrées telles que le dimensionnement automatique, les licences et l'accès aux logiciels. Les flottes gérées par le client fonctionnent sur vos propres ressources informatiques, telles que des EC2 instances Amazon ou des serveurs sur site.

Budget

Un budget définit des seuils de dépenses pour votre activité professionnelle et vous permet de prendre des mesures lorsque les seuils sont atteints, comme arrêter la planification des tâches.

Environnement de file d'attente

Un environnement de file d'attente définit les scripts qui s'exécutent sur chaque travailleur pour configurer ou démonter l'environnement de charge de travail. Ils sont utiles pour définir des variables d'environnement, installer des logiciels et configurer le stockage des actifs.

Profil de stockage

Un profil de stockage est une configuration pour un groupe d'hôtes et de postes de travail, qui indique où se trouvent les données dans le système de fichiers. Deadline Cloud utilise des profils de stockage pour cartographier les chemins lors de l'exécution de tâches sur des hôtes configurés différemment, comme une tâche soumise Windows et exécutée surLinux.

Limite

Une limite vous permet de suivre l'utilisation des ressources partagées, telles que les licences flottantes, et de contrôler la manière dont elles sont réparties entre les tâches. Les limites sont associées à des files d'attente associées à des limites de file d'attente.

Surveillance

Le moniteur configure l'URL de l'application Web Deadline Cloud Monitor, permettant aux utilisateurs finaux de surveiller et de gérer les tâches. Il est accessible dans un navigateur ou via l'application de bureau Deadline Cloud Monitor.

Ressources pour l'exécution des tâches

Ce diagramme montre comment les ressources de travail de Deadline Cloud fonctionnent ensemble.

Tâche

Une tâche est un ensemble de tâches qu'un utilisateur soumet à Deadline Cloud pour être planifié et exécuté sur les travailleurs disponibles. Une tâche peut effectuer le rendu d'une scène 3D ou exécuter une simulation. Les tâches sont créées à partir de modèles de tâches réutilisables, qui définissent l'environnement d'exécution et les processus, ainsi que des paramètres spécifiques aux tâches. Les tâches contiennent des étapes et des tâches qui définissent le travail à effectuer, et elles peuvent être configurées avec des priorités, le nombre maximum de travailleurs et des paramètres de nouvelle tentative.

Priorité du job

La priorité des tâches est l'ordre approximatif dans lequel Deadline Cloud traite une tâche dans une file d'attente. Vous pouvez définir la priorité des tâches entre 1 et 100, les tâches ayant une priorité numérique plus élevée sont généralement traitées en premier. Les tâches ayant la même priorité sont traitées dans l'ordre de réception.

Propriétés de la tâche

Les propriétés de la tâche sont des paramètres que vous définissez lorsque vous soumettez une tâche de rendu. Parmi les exemples, citons la plage d'images, le chemin de sortie, les pièces jointes aux tâches, la caméra rendable, etc. Les propriétés varient en fonction du DCC à partir duquel le rendu est soumis.

Step (Étape)

Une étape fait partie d'une tâche qui fournit un modèle pour exécuter de nombreuses tâches identiques, à l'exception des valeurs des paramètres de tâche. Les étapes peuvent être dépendantes d'autres étapes, ce qui vous permet de créer des flux de travail complexes avec des chemins d'exécution séquentiels ou parallèles. Dans les tâches de rendu, une étape définit souvent la commande de rendu d'un cadre et utilise le numéro du cadre comme paramètre de tâche.

Sous-tâche

Une tâche est la plus petite unité de travail dans Deadline Cloud. Les tâches font partie des étapes et sont exécutées par les travailleurs, ce qui représente des opérations individuelles qui doivent être effectuées dans le cadre d'un travail. Les tâches peuvent être configurées avec des paramètres spécifiques et sont attribuées aux travailleurs en fonction de leurs capacités et de leur disponibilité. Dans les tâches de rendu, une tâche effectue souvent le rendu d'une seule image.

Nœuds

Les travailleurs font partie d'une flotte et exécutent des tâches à partir de leur poste. Les travailleurs peuvent être configurés avec des fonctionnalités spécifiques telles que les accélérateurs GPU, l'architecture du processeur et le système d'exploitation. Dans les flottes gérées par des services, les travailleurs sont créés automatiquement au fur et à mesure que le parc évolue.

Instance

Les flottes utilisent des instances pour les ressources du processeur. Une instance est une instance Amazon EC2 Performance. Deadline Cloud utilise des instances On-Demand et Spot.

Instance à la demande

Les instances à la demande sont facturées à la seconde, n'ont aucun engagement à long terme et ne seront pas interrompues.

Instance ponctuelle

Les instances ponctuelles sont des capacités non réservées que vous pouvez utiliser à un prix réduit, mais qui peuvent être interrompues par des demandes à la demande.

Attendez et économisez

La fonction Wait and Save permet une planification différée des tâches à moindre coût et peut être interrompue par des demandes ponctuelles ou à la demande. Wait and Save n'est disponible que dans les flottes gérées par le service Deadline Cloud.

Wait and Save permet de gérer l'exécution des charges de travail informatiques visuelles dans AWS Deadline Cloud. Consultez les conditions AWS de service pour plus de détails.

Session

Une session représente la séquence de travail d'un travailleur sur une tâche. Au cours d'une même session, un travailleur peut se voir attribuer plusieurs tâches qu'il exécute les unes après les autres. Les sessions comportent souvent des actions de configuration qui configurent les environnements et chargent les actifs avant d'exécuter les actions de tâche.

Action de session

Une action de session représente des opérations spécifiques effectuées au cours d'une session, telles que la configuration de l'environnement, l'exécution d'une tâche et la synchronisation des actifs.

Autres concepts et terminologies importants

Explorateur d'utilisation

L'explorateur d'utilisation est une fonctionnalité du moniteur Deadline Cloud. Il fournit une estimation approximative de vos coûts et de votre utilisation.

Directeur du budget

Le gestionnaire de budget fait partie du moniteur Deadline Cloud. Utilisez le gestionnaire de budget pour créer et gérer des budgets. Vous pouvez également l'utiliser pour limiter les activités afin de respecter le budget.

Bibliothèque cliente de Deadline Cloud

La bibliothèque client open source inclut une interface de ligne de commande et une bibliothèque pour gérer Deadline Cloud. Les fonctionnalités incluent la soumission de lots de tâches basés sur la spécification Open Job Description à Deadline Cloud, le téléchargement des résultats des pièces jointes aux tâches et la surveillance de votre ferme à l'aide de l'interface de ligne de commande (CLI).

Application de création de contenu numérique (DCC)

Les applications de création de contenu numérique (DCCs) sont des produits tiers dans lesquels vous créez du contenu numérique. Deadline Cloud intègre des intégrations avec de nombreuses entreprises DCCs telles qu'Autodesk Maya, Blender et Maxon Cinema 4D, ce qui vous permet de soumettre des tâches depuis le DCC et d'effectuer des rendus sur des flottes gérées par des services avec des logiciels et des licences préconfigurés.

Pièces jointes aux offres d'emploi

Les pièces jointes aux tâches sont une fonctionnalité de Deadline Cloud qui vous permet de charger et de télécharger des éléments dans le cadre d'une tâche, tels que des textures, des modèles 3D et des appareils d'éclairage. Les pièces jointes aux tâches sont stockées dans Amazon S3 et évitent le besoin d'un stockage réseau partagé.

Modèle de tâche

Un modèle de tâche définit l'environnement d'exécution et tous les processus exécutés dans le cadre d'une tâche Deadline Cloud.

Expéditeur de Deadline Cloud

Un émetteur Deadline Cloud est un plugin pour un DCC qui permet aux utilisateurs de soumettre facilement des tâches depuis le DCC.

Endpoint de licence

Un point de terminaison de licence rend les licences basées sur l'utilisation de Deadline Cloud pour les produits tiers disponibles dans votre VPC. Ce modèle est payant au fur et à mesure, et vous êtes facturé en fonction du nombre d'heures et de minutes que vous utilisez. Les points de terminaison de licence ne sont pas connectés aux parcs de serveurs et peuvent être utilisés indépendamment.

Étiquettes

Une étiquette est une étiquette que vous pouvez attribuer à une AWS ressource. Chaque balise est composée d'une clé et d'une valeur facultative que vous définissez. Les balises vous permettent de classer vos AWS ressources de différentes manières, par exemple par objectif, propriétaire ou environnement.

Licences basées sur l'utilisation (UBL)

Les licences basées sur l'utilisation (UBL) sont un modèle de licence à la demande disponible pour certains produits tiers. Ce modèle est payant au fur et à mesure, et vous êtes facturé en fonction du nombre d'heures et de minutes que vous utilisez.

Conseils sur l'architecture cloud de Deadline

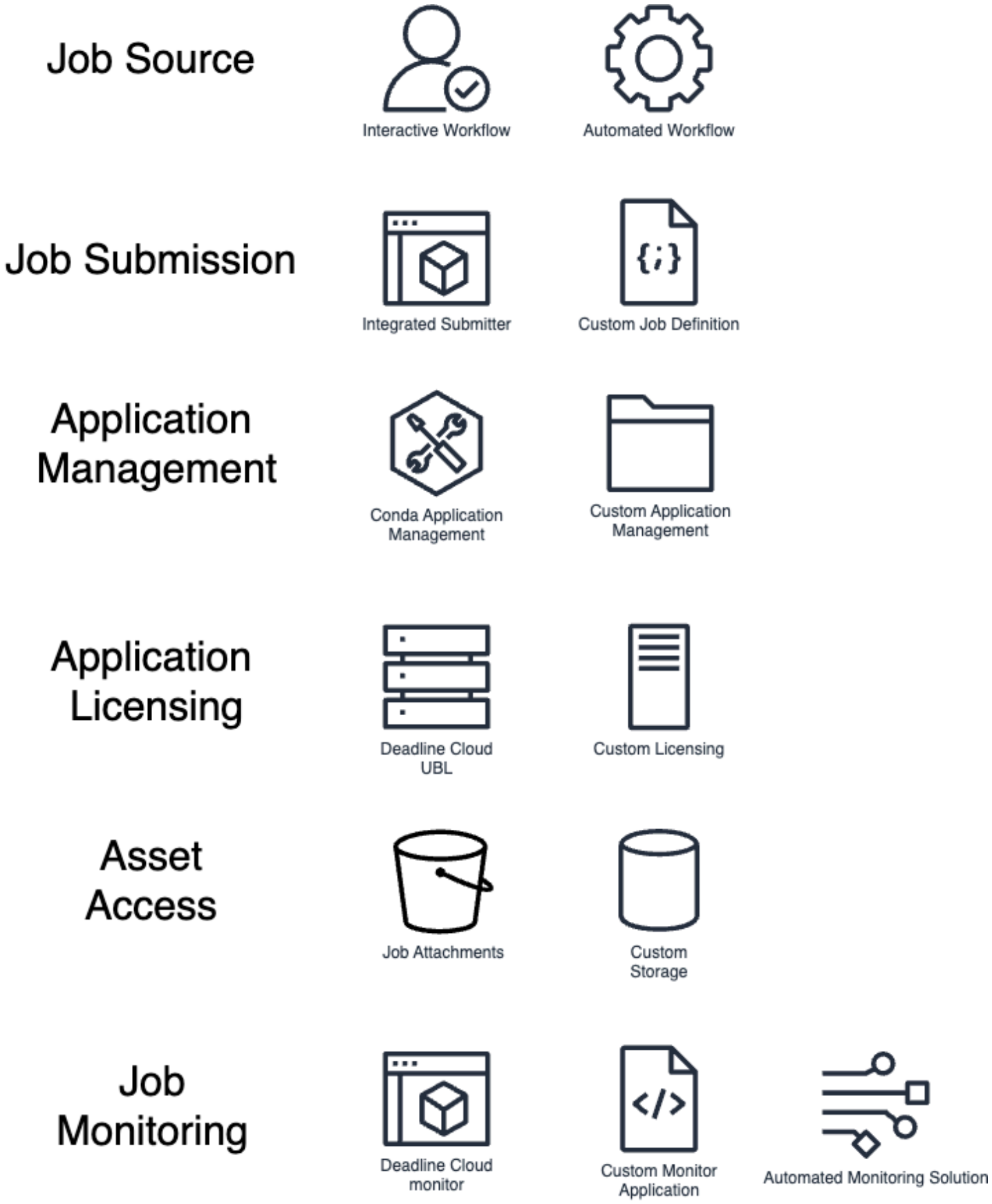
Cette rubrique fournit des conseils et des bonnes pratiques pour concevoir et créer des fermes de rendu fiables, sécurisées, efficaces et économiques pour vos charges de travail à l'aide de Deadline Cloud. L'utilisation de ces conseils peut vous aider à créer des charges de travail stables et efficaces, vous permettant de vous concentrer sur l'innovation, de réduire les coûts et d'améliorer l'expérience des clients.

Ce contenu est destiné aux directeurs de la technologie (CTOs), aux architectes, aux développeurs et aux membres de l'équipe opérationnelle.

Un flux de end-to-end rendu nécessite des solutions à plusieurs niveaux du processus, tels que la génération de tâches, l'accès aux actifs et le suivi des tâches. Deadline Cloud propose plusieurs solutions pour chaque couche du processus de rendu. En sélectionnant l'une des options de Deadline Cloud dans chaque couche, vous pouvez concevoir un flux de travail adapté à votre cas d'utilisation.

Pour chaque couche, vous devez choisir l'approche la mieux adaptée à votre cas d'utilisation. Il ne s'agit pas de définitions de scénarios strictes et ne constituent pas le seul moyen d'utiliser Deadline Cloud. Il s'agit plutôt d'un ensemble de concepts de haut niveau destinés à vous aider à comprendre comment Deadline Cloud peut s'intégrer à votre activité ou à votre flux de travail. Vous pouvez séparer les charges de travail de Deadline Cloud dans les couches suivantes : source des tâches, soumission des tâches, gestion des applications, licences des applications, accès aux actifs, gestion des sorties et gestion de l'infrastructure des travailleurs.

En général, vous pouvez utiliser mix-and-match n'importe quel scénario dans une couche avec n'importe quel autre scénario dans une autre couche, à l'exception des combinaisons spécifiques spécifiées ci-dessous.


Job Source
Interactive Workflow
Automated Workflow
Job Submission
Integrated Submitter
Custom Job Definition
Application Management
Conda Application Management
Custom Application Management
Application Licensing
Deadline Cloud UBL
Custom Licensing
Asset Access
Job Attachments
Custom Storage
Job Monitoring
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution
Worker Infrastructure

Source du job

La source des tâches est le point d'accès par lequel les nouvelles tâches entreront dans le système pour être affichées par Deadline Cloud. À haut niveau, il existe deux sources principales d'emplois : l'interactivité humaine et les systèmes informatiques automatisés.

Flux de travail interactif

Dans ce scénario, un artiste ou un autre acteur créatif est le principal générateur de travail à traiter dans la ferme Deadline Cloud. En général, le résultat de ces tâches est un artefact principal pour le projet ou l'équipe de plus grande envergure. Ils effectuent leur travail à l'aide de logiciels tels qu'un outil de création de contenu numérique (DCC) conforme aux normes de l'industrie. Ils soumettent manuellement des tâches à la ferme Deadline Cloud et consultent ensuite les résultats pour les examiner. Le poste de travail lui-même n'est pas géré par AWS.

Dans la plupart des cas, ces artistes utilisent les émetteurs intégrés de Deadline Cloud et le moniteur Deadline Cloud dans les couches Application et Monitoring de la charge de travail.

Flux de travail automatisé

Dans ce scénario, un système programmatique appartenant au client est le principal générateur de tâches dans la ferme Deadline Cloud. Il peut s'agir de la génération d'actifs dans un pipeline de vente au détail, comme une vidéo sur plaque tournante générée à partir d'un modèle 3D ou d'un scan. Il pourrait s'agir de la composition automatisée de graphiques diffusés et de cartes de joueur pour le sport. Le thème de ce scénario est qu'une personne ne soumet pas manuellement chaque tâche à Deadline Cloud, mais que la tâche est générée dans le cadre d'un système plus vaste.

Dans le cas des tâches automatisées, il est moins courant d'utiliser les soumetteurs intégrés à Deadline Cloud et le moniteur Deadline Cloud. Souvent, les définitions des tâches concernent le développement d'applications personnalisées que vous avez rédigées et les résultats des tâches seront automatiquement transférés dans un système de gestion des actifs numériques (DAM) ou un système de gestion des actifs multimédias (MAM) pour approbation et distribution.

Soumission d'un job

Les tâches sont soumises à Deadline Cloud à l'aide OpenJobDescriptionde modèles. OpenJobDescription est une spécification ouverte flexible permettant de définir des tâches de traitement par lots portables entre différents déploiements de systèmes de planification. Le fichier

de définition du job décrit les paramètres du job, les étapes du job, la façon dont une étape est paramétrée en fonction des entrées du job, ainsi que le script réel qui sera exécuté sur un Worker pour effectuer le traitement. L'idée de la soumission de charge de travail est de savoir comment ces définitions de tâches sont créées, qui les crée et comment elles sont soumises.

Émetteur intégré avec DCC

Un émetteur intégré de Deadline Cloud est un logiciel qui relie Deadline Cloud à un DCC ou à un progiciel standard du secteur. L'émetteur intégré détermine comment transformer les données et la configuration d'une charge de travail de rendu, composite ou autre en un modèle de tâche, ce qui peut être compris par Deadline Cloud. De nombreux soumetteurs intégrés sont créés et gérés par l'équipe de Deadline Cloud ou le créateur du progiciel, mais s'il n'en existe pas déjà un pour l'application souhaitée, vous pouvez créer et gérer votre propre émetteur. Il existe un nombre limité de DCCs solutions prises en charge par l'équipe de Deadline Cloud.

Les flux de travail interactifs impliquent généralement des émetteurs intégrés, mais pas toujours. Pour les flux de travail automatisés basés sur des modèles, un flux de travail courant consiste pour un artiste à configurer un modèle de travail dans son DCC et à effectuer une exportation unique du lot de tâches. Cet ensemble de tâches définit comment exécuter ce type de tâche spécifique sur Deadline Cloud de manière paramétrée. Cet ensemble de tâches peut être intégré dans le scénario de flux de travail automatisé à des fins d'automatisation.

Définition de tâche personnalisée

Pour les applications et les flux de travail personnalisés, il est possible de contrôler entièrement la manière dont ces définitions de tâches sont créées et soumises à Deadline Cloud. Par exemple, un site de commerce électronique peut demander aux vendeurs de télécharger des modèles 3D de l'objet qu'ils vendent. Après ce téléchargement, la plateforme de commerce électronique pourrait générer dynamiquement une définition de tâche à soumettre à Deadline Cloud afin de générer automatiquement une animation de plaque tournante sur un fond commun en utilisant un éclairage commun correspondant aux autres objets 3D disponibles sur le site. Au cours du développement de la plateforme de commerce électronique, un développeur de logiciel créerait une définition de tâche, l'intégrerait dans la plate-forme de commerce électronique avec les paramètres éventuellement fournis par les vendeurs, et coderait la plateforme pour soumettre cette tâche pendant le flux de travail de téléchargement des produits de la plateforme.

Deadline Cloud fournit un certain nombre d'exemples de définitions de tâches dans le référentiel d'exemples sur github.

Gestion des applications

Une fois qu'un travail est soumis à Deadline Cloud et attribué à un travailleur, le script issu de la définition du travail est exécuté sur le travailleur. Dans la plupart des cas, ce script invoque une application pour effectuer le traitement proprement dit, comme un moteur de rendu, un composite, un encodage, un filtrage ou toute autre tâche parmi un certain nombre de tâches gourmandes en ressources informatiques. La gestion des applications est le concept visant à garantir que la version nécessaire du logiciel requis est disponible pour les travailleurs.

Vous pouvez gérer les applications à l'aide de n'importe quel système de gestion de packages de votre choix, mais Deadline Cloud fournit un certain nombre d'outils pour permettre facilement l'utilisation des packages conda. Conda est un gestionnaire de packages et un système de gestion d'environnement open source, multiplateforme et indépendant de la langue.

Canal Conda géré par Deadline Cloud pour les flottes gérées par des services (SMF)

Lorsque vous utilisez des flottes gérées par des services, un canal conda géré par Deadline Cloud est automatiquement configuré et configuré pour être utilisé par vos tâches. Le service Deadline Cloud fournit un certain nombre d'applications DCC partenaires et les rend sur ce canal conda. Pour plus d'informations, voir Création d'un environnement de file d'attente dans le guide de l'utilisateur de Deadline Cloud. Ces packages sont automatiquement mis à jour par le service Deadline Cloud et ne nécessitent aucune maintenance de votre part. Ce canal conda n'est disponible que lorsque vous utilisez des flottes gérées par des services et n'est pas disponible lorsque vous utilisez des flottes gérées par le client.

Canal Conda autogéré

Si vous n'êtes pas en mesure d'utiliser le canal conda géré par Deadline Cloud, vous devez déterminer comment installer, corriger et gérer les applications de votre flotte Deadline Cloud. L'une des options consiste à créer un canal conda que vous configurez et gérez. Cela interagira très étroitement avec le canal conda géré par Deadline Cloud. Par exemple, vous pouvez utiliser un DCC issu du canal conda géré par Deadline Cloud, mais apporter votre propre package contenant un plugin DCC spécifique. Pour plus d'informations sur ce processus, consultez Créer un canal conda à l'aide de S3.

Gestion personnalisée des applications

Pour la gestion des applications, Deadline Cloud exige que l'application soit disponible dans le PATH lorsque le script de tâche est exécuté sur le worker.

Si vous créez et gérez déjà des packages Rez, vous pouvez utiliser un environnement de file d'attente pour installer les applications à partir des référentiels Rez. Un exemple d'environnement de file d'attente est disponible sur l' GitHub organisation AWS Deadline Cloud.

Si vous gérez déjà des applications dans des flottes gérées par le client avec des employés de longue date ou dans des images système, aucun environnement de file d'attente n'est requis pour la gestion des applications. Assurez-vous que la candidature apparaît sur le chemin de l'utilisateur du job et soumettez le job.

Licences de l'application

De nombreuses charges de travail généralement exécutées sur Deadline Cloud nécessitent une licence logicielle auprès du fournisseur du logiciel. Ces applications font souvent l'objet d'une licence par siège, par processeur ou par hôte. Il est de votre responsabilité de vous assurer que votre utilisation de logiciels tiers sur Deadline Cloud respecte le contrat de licence tiers. Si vous utilisez un logiciel open source, un logiciel personnalisé ou un logiciel sans licence, il n'est pas nécessaire de configurer cette couche. N'oubliez pas que Deadline Cloud ne prend en charge que les licences de rendu et non les licences de station de travail.

Flottes gérées par des services et licences basées sur l'utilisation

Lorsque vous utilisez des flottes gérées par le service Deadline Cloud, les licences basées sur l'utilisation (UBL) sont automatiquement configurées pour les logiciels pris en charge. Les tâches exécutées sur des flottes gérées par des services sont automatiquement définies pour les variables d'environnement pour les applications prises en charge afin de les inciter à utiliser les serveurs de licences Deadline Cloud. Lorsque vous utilisez Deadline Cloud UBL, vous n'êtes facturé que pour le nombre d'heures pendant lesquelles vous utilisez l'application sous licence.

Flottes gérées par le client et licences basées sur l'utilisation

Les licences basées sur l'utilisation (UBL) de Deadline Cloud sont également disponibles lorsque vous n'utilisez pas de flottes gérées par des services. Dans ce scénario, vous allez configurer des

points de terminaison de licence Deadline Cloud qui fournissent des adresses IP dans les sous-réseaux VPC que vous avez sélectionnés et qui donnent accès aux serveurs de licences Deadline Cloud. Une fois que vous avez configuré les variables d'environnement spécifiques au logiciel appropriées pour vos travailleurs et configuré la connectivité réseau entre les travailleurs et les adresses IP des points de terminaison de licence, les travailleurs peuvent récupérer et enregistrer les licences des logiciels pris en charge. Les licences vous sont facturées à l'heure de la même manière que lorsque vous utilisez UBL dans des flottes gérées par des services.

Licences personnalisées

Vous pouvez utiliser une application qui n'est pas prise en charge par Deadline Cloud UBL ou vous avez peut-être des licences préexistantes toujours valides. Dans ce scénario, vous êtes responsable de la configuration du chemin réseau entre vos employés (géré par le client ou le service) et les serveurs de licences. Pour plus d'informations sur les licences personnalisées, consultezConnectez des flottes gérées par des services à un serveur de licences personnalisé.

Accès aux actifs

Une fois qu'une tâche est soumise à un travailleur et que l'application est configurée, le travailleur doit être configuré pour accéder aux données d'actifs requises pour le travail. Il peut s'agir de données 3D, de données de texture, de données d'animation, d'images vidéo ou de tout autre type de données utilisées dans le cadre de votre travail.

Commencez par réfléchir à l'endroit où vos données sont actuellement stockées. Cela peut se trouver sur le disque dur du poste de travail, sur un outil de collaboration utilisateur, sur le contrôle de source, sur un système de fichiers partagé sur site ou dans le cloud, sur Amazon S3 ou sur d'autres sites.

Ensuite, déterminez ce qui est nécessaire pour qu'un travailleur accède à ces données. Ces données sont-elles uniquement disponibles sur le réseau de votre entreprise ? Quelle est l'identité ou les informations d'identification requises pour accéder aux données ? La source de données est-elle adaptée à la tâche en fonction du nombre de travailleurs que vous prévoyez de traiter ?

Pièces jointes aux offres d'emploi

Le mécanisme d'accès aux ressources le plus simple pour commencer est celui des pièces jointes aux tâches de Deadline Cloud. Lorsqu'une tâche est soumise à l'aide de pièces jointes, les données requises par la tâche sont chargées dans un compartiment Amazon S3 avec un fichier manifeste spécifiant les fichiers requis par la tâche. Avec les pièces jointes aux tâches, aucune configuration

complexe de mise en réseau ou de stockage partagé n'est requise. Les fichiers ne sont chargés qu'une seule fois, de sorte que les téléchargements suivants se terminent plus rapidement. Une fois qu'un travailleur a terminé de traiter une tâche, les données de sortie sont chargées sur Amazon S3 afin qu'elles puissent être téléchargées par l'artiste ou un autre client. Les accessoires Job s'adaptent aux flottes de toutes tailles et sont simples et rapides à intégrer et à utiliser.

Les pièces jointes aux tâches ne sont pas le meilleur outil pour toutes les situations. Si vos données sont déjà présentes AWS, les pièces jointes aux tâches ajoutent une copie supplémentaire de vos données, y compris le temps de transfert et les coûts de stockage associés. Les pièces jointes aux tâches nécessitent que la tâche puisse spécifier complètement les données dont elle a besoin au moment de la soumission, afin que les données puissent être téléchargées.

Pour utiliser des pièces jointes à des tâches, votre file d'attente Deadline Cloud doit être associée à un compartiment de pièces jointes et le rôle de file d'attente doit être utilisé pour fournir l'accès à ce compartiment. Par défaut, les soumissionnaires intégrés à Deadline Cloud prennent tous en charge les pièces jointes aux tâches. Si vous n'utilisez pas d'émetteur intégré à Deadline Cloud, les pièces jointes aux tâches peuvent être utilisées avec votre logiciel personnalisé en intégrant la bibliothèque python de Deadline Cloud.

Accès au stockage personnalisé

Si vous n'utilisez pas de pièces jointes aux offres d'emploi, vous devez vous assurer que les travailleurs ont accès aux données requises pour les offres d'emploi. Deadline Cloud fournit un certain nombre d'outils pour y parvenir et pour garantir la portabilité des tâches. Vous souhaiterez peut-être utiliser une solution de stockage personnalisée lorsque vous disposez déjà d'un stockage réseau partagé pour les artistes et les travailleurs, si vous préférez utiliser un service externe LucidLink, ou pour d'autres raisons.

Utilisez des profils de stockage pour modéliser les systèmes de fichiers de votre poste de travail et de vos ordinateurs de travail. Chaque profil de stockage décrit le système d'exploitation et la structure du système de fichiers de l'une de vos configurations système. À l'aide de profils de stockage, lorsqu'un artiste utilisant un poste de travail Windows soumet une tâche traitée par un Linux collaborateur, Deadline Cloud veille à ce que le chemin soit mappé afin que le collaborateur puisse accéder au stockage de données que vous avez configuré.

Lorsque vous utilisez des flottes gérées par les services Deadline Cloud, les scripts de configuration de l'hôte et les points de terminaison des ressources VPC permettent aux employés de monter et d'accéder directement au stockage partagé ou à d'autres services disponibles dans votre VPC.

Surveillance des tâches et gestion des résultats

Une fois les tâches soumises à Deadline Cloud terminées avec succès, une personne ou un processus téléchargera le résultat de la tâche pour l'utiliser dans le flux de travail commercial en dehors de Deadline Cloud. En cas d'échec d'une tâche, les journaux des tâches et les informations de surveillance permettent de diagnostiquer les problèmes.

Moniteur Deadline Cloud

L'application de surveillance Deadline Cloud est disponible sur le Web et sur ordinateur. Cette solution convient parfaitement aux studios utilisant des flux de travail interactifs pour un large éventail d'utilisations de pièces jointes DCCs à des fins de stockage. Le moniteur ne vous prend en charge que lorsque vous utilisez IAM Identity Center. IAM Identity Center est un produit Workforce Identity, et non une solution d'identité grand public (B2C). Il n'est donc pas adapté à de nombreux scénarios B2C.

Application de moniteur personnalisée

Si vous souhaitez personnaliser l'expérience de surveillance de vos utilisateurs, si vous créez un produit B2C ou si vous créez un système hautement spécialisé à l'aide de Deadline Cloud, vous choisissez de créer une application de surveillance personnalisée. Vous pouvez utiliser l'API AWS Deadline Cloud pour créer cette application personnalisée, en combinant le contexte de votre flux de travail global avec les concepts de Deadline Cloud. Par exemple, votre produit B2C peut avoir son propre concept de projet que les utilisateurs configurent et votre application peut imbriquer les tâches Deadline Cloud dans la même interface.

Solution de surveillance automatisée

Dans certains scénarios, aucune application de surveillance dédiée n'est nécessaire pour Deadline Cloud. Ce scénario est courant dans les flux de travail automatisés où Deadline Cloud est utilisé pour afficher automatiquement les actifs d'un pipeline, tels que des graphiques de diffusion pour le sport ou les actualités. Dans ce scénario, l'API et les EventBridge événements de Deadline Cloud sont utilisés pour s'intégrer à un système de gestion des actifs multimédias externe pour les approbations et le transfert des données vers l'étape suivante du processus.

Gestion de l'infrastructure des travailleurs

Les flottes Deadline Cloud sont un groupe de serveurs (travailleurs) capables de traiter les tâches soumises à une file d'attente Deadline Cloud et constituent l'infrastructure principale de toute ferme Deadline Cloud.

Flottes gérées par des services

Dans un parc géré par des services, Deadline Cloud prend en charge les hôtes, le système d'exploitation, le réseau, les correctifs, le dimensionnement automatique et les autres facteurs liés à la gestion d'un parc de rendu. Vous spécifiez le nombre minimum et maximum de travailleurs que vous souhaitez, ainsi que les spécifications du système requises pour votre application, et Deadline Cloud se charge du reste. Les flottes gérées par des services sont la seule option de flotte qui peut utiliser les canaux conda gérés par Deadline Cloud pour gérer facilement les applications DCC du secteur. En outre, Deadline Cloud UBL est automatiquement configuré avec des flottes gérées par des services. Les flottes Wait and Save pour des charges de travail moins coûteuses et tolérantes aux retards ne sont disponibles qu'à l'aide de flottes gérées par des services.

Flottes gérées par le client

Vous utilisez des flottes gérées par le client lorsque vous avez besoin d'un meilleur contrôle sur les hôtes de travail et leur environnement. Les flottes gérées par le client sont particulièrement adaptées à l'utilisation de Deadline Cloud sur site. Pour en savoir plus, veuillez consulter la section Créez et utilisez des flottes gérées par les clients de Deadline Cloud.

Exemples d'architectures

Studio de production traditionnel

Le studio de production traditionnel nécessite une infrastructure de calcul, de stockage et de réseau importante capable de couvrir plusieurs sites physiques pour répondre aux charges de travail de rendu. Chaque progiciel et chaque fournisseur ont des exigences uniques en matière de matériel, de logiciel, de réseau et de licence qui doivent être respectées tout en résolvant les conflits de version, de compatibilité et de ressources.

Il est courant d'avoir des exigences d'infrastructure distinctes pour les postes de travail des artistes, les nœuds de rendu, le stockage réseau, les serveurs de licences, les systèmes de mise en file

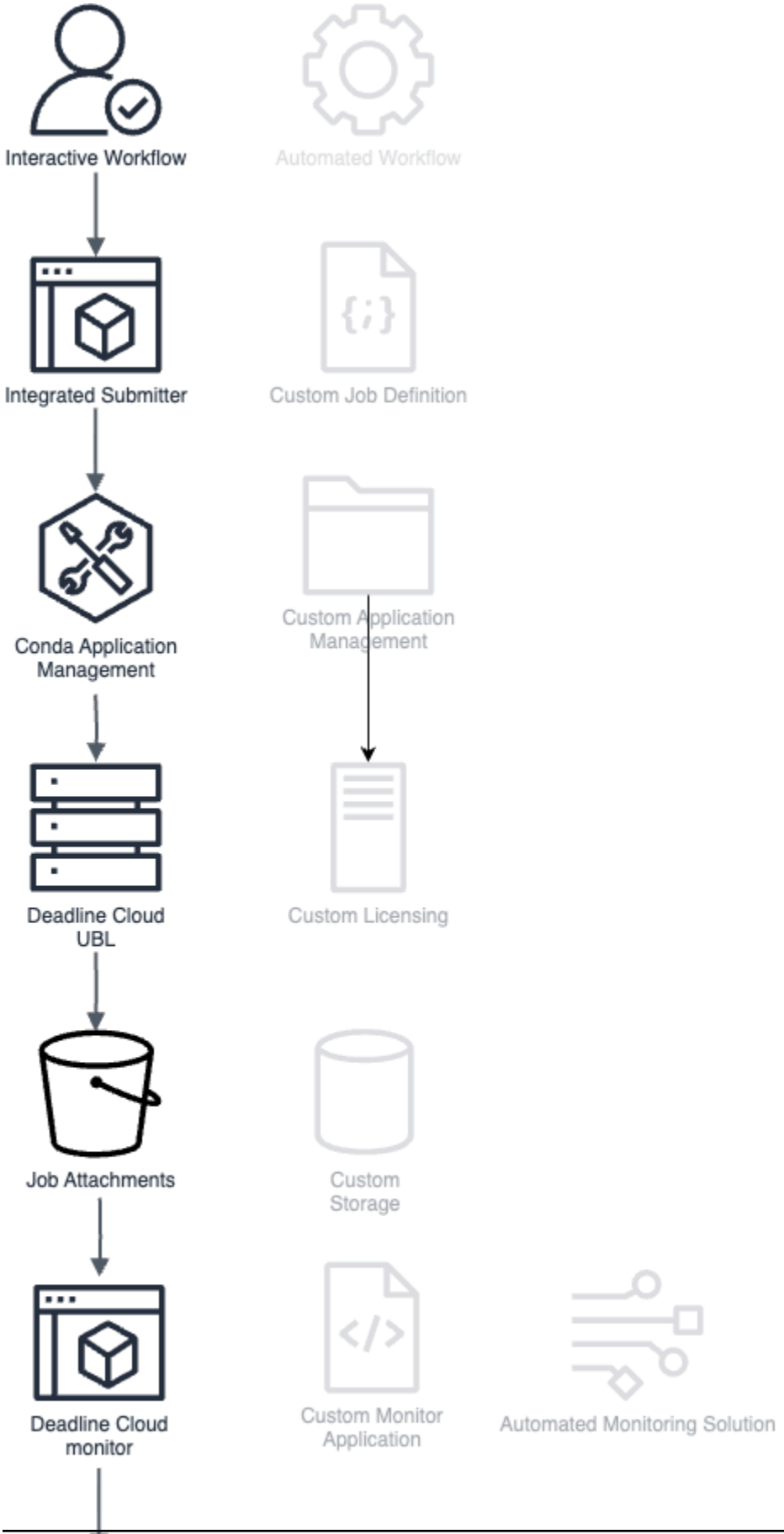
d'attente des tâches, les outils de surveillance et la gestion des actifs. Les studios ont généralement besoin de gérer plusieurs versions des outils, moteurs de rendu, plugins et outils personnalisés DCC tout en gérant des accords de licence complexes au sein de leur parc de rendu. L'infrastructure de votre studio devient plus complexe lorsque vous considérez les environnements de développement, d'assurance qualité et de production.

Un déploiement typique de Deadline Cloud à l'aide d'options gérées par les services permet de résoudre ou de réduire bon nombre de ces défis en :

- Soumission de tâches de flux de travail interactive via des soumetteurs DCC intégrés
- Gestion des applications via les canaux conda gérés par Deadline Cloud
- Licences basées sur l'utilisation configurées automatiquement pour les logiciels pris en charge
- Gestion des actifs grâce à des offres d'emploi
- Surveillance via l'application de surveillance Deadline Cloud
- Gestion de l'infrastructure par le biais de flottes gérées par des services

Grâce à cette approche, les artistes peuvent soumettre des travaux directement depuis leurs outils DCC habituels vers une ferme de rendu cloud évolutive sans avoir à gérer une infrastructure complexe. Le service gère automatiquement le déploiement des logiciels, les licences, le transfert de données et le dimensionnement de l'infrastructure. Les artistes peuvent suivre leur travail via une interface Web ou une application de bureau, et les résultats sont automatiquement stockés dans Amazon S3 pour un accès facile.

Grâce à cette configuration, les studios peuvent créer des environnements de développement et de production en quelques minutes, ne payer que pour le calcul et les licences qu'ils utilisent, et se concentrer sur le travail créatif plutôt que sur la gestion de l'infrastructure. L'approche basée sur la gestion des services constitue le moyen le plus rapide d'adopter le rendu dans le cloud tout en préservant les flux de travail habituels des artistes.


Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Studio dans le cloud

Les studios d'effets visuels et d'animation modernes transfèrent de plus en plus l'ensemble de leur pipeline vers le cloud, y compris les postes de travail des artistes. Cette approche élimine le besoin d'une infrastructure sur site, permet une collaboration mondiale et permet une mise à l'échelle fluide pour le travail interactif et le rendu. Cependant, cela pose également de nouveaux défis en termes de gestion des ressources cloud, de garantie d'un accès aux données à faible latence et d'intégration des stations de travail basées sur le cloud aux fermes de rendu.

Un studio cloud natif typique nécessite une approche unifiée pour gérer les postes de travail cloud, le stockage partagé, l'infrastructure de rendu et le déploiement de logiciels sur tous ces composants. Les approches traditionnelles ont souvent donné lieu à des systèmes complexes gérés manuellement qui avaient du mal à trouver un équilibre entre performance, coût et flexibilité.

Un déploiement de Deadline Cloud pour un studio cloud natif peut être mis en œuvre en utilisant :

- Soumission de tâches de flux de travail interactive via des émetteurs DCC intégrés sur des postes de travail cloud
- Gestion des applications via les nœuds de rendu Conda Channels gérés par Deadline Cloud
- Licences basées sur l'utilisation configurées automatiquement pour les logiciels pris en charge
- Accès au stockage personnalisé à l'aide du serveur de Windows fichiers FSx pour les données de projet partagées
- Surveillance via l'application de surveillance Deadline Cloud
- Gestion de l'infrastructure à l'aide de flottes gérées par des services

Cette approche permet aux artistes de travailler sur des postes de travail basés sur le cloud avec un accès direct à un stockage partagé haute performance et de soumettre facilement des tâches à la ferme Deadline Cloud. Le studio peut gérer le déploiement des logiciels sur les postes de travail et les nœuds de rendu en utilisant les mêmes canaux conda, garantissant ainsi la cohérence et réduisant les frais de maintenance.

Les principaux avantages de cette configuration sont les suivants :

- Collaboration mondiale avec des artistes capables d'accéder aux postes de travail de n'importe où
- Environnements logiciels cohérents entre les postes de travail et les nœuds de rendu
- Stockage partagé hautes performances accessible à la fois aux postes de travail et aux nœuds de rendu

- Mise à l'échelle flexible des ressources de calcul interactives et par lots
- Gestion centralisée de toute l'infrastructure du studio dans le cloud

Dans ce scénario, la configuration du stockage implique généralement :

- FSx pour Windows le serveur de fichiers pour les données de projet, accessible à la fois par les postes de travail cloud et par les employés de Deadline Cloud
- Profils de stockage dans Deadline Cloud pour gérer le mappage des chemins entre les postes de travail et les nœuds de rendu
- Montage direct de FSx partages sur les serveurs de Deadline Cloud à l'aide de points de terminaison de ressources VPC et de scripts de configuration d'hôtes

Cette approche native du cloud permet aux studios d'éliminer l'infrastructure sur site, permettant ainsi une mise à l'échelle rapide pour des projets de toute taille tout en conservant les flux de travail habituels des artistes. Il offre la flexibilité nécessaire pour utiliser une combinaison de ressources gérées par les services et gérées par le client, en optimisant à la fois la facilité de gestion et les exigences de performance spécifiques.

En tirant parti des stations de travail cloud associées à Deadline Cloud, les studios peuvent créer un pipeline de production entièrement intégré et accessible dans le monde entier, qui s'adapte parfaitement aux petites équipes aux grandes productions.

ECommerce Automatisation

La plate-forme de commerce électronique moderne nécessite une génération d'actifs automatisée à grande échelle pour fournir une visualisation complète des produits sur des millions d'articles. Les approches traditionnelles nécessitaient d'importants investissements en infrastructure pour traiter de grands volumes de modèles 3D en supports de produits standardisés, ce qui se traduisait souvent soit par des systèmes sous-approvisionnés, ce qui créait des arriérés de traitement, soit par des systèmes surapprovisionnés avec une capacité inutilisée.

Un flux de travail de commerce électronique automatisé typique doit gérer le traitement du téléchargement des produits, la validation des modèles 3D, la gestion du parc de rendu, le traitement des sorties et l'intégration aux systèmes d'information sur les produits. La gestion de ces flux de travail nécessite traditionnellement de coordonner plusieurs applications de rendu, ressources informatiques et pipelines de traitement des données, tout en garantissant une qualité constante et en maintenant la rentabilité à grande échelle.

Un déploiement de Deadline Cloud pour l'automatisation du commerce électronique peut être mis en œuvre en utilisant :

- Soumission automatisée des tâches de flux de travail grâce à l'intégration d'une API personnalisée dans l'application d'ingestion de commerce électronique existante
- Définitions de tâches personnalisées adaptées à la visualisation standardisée des produits
- Gestion des applications via les canaux conda gérés par Deadline Cloud
- Licences basées sur l'utilisation configurées automatiquement pour les logiciels pris en charge
- Intégration directe avec Amazon S3 pour la gestion des actifs
- Application de surveillance personnalisée intégrée aux systèmes de gestion de produits existants
- Des flottes gérées par des services pour une évolutivité élastique

Cette approche permet de traiter des milliers de produits par jour, en générant automatiquement des visualisations de produits standardisées, telles que des animations de platines. L'infrastructure gérée par les services évolue automatiquement pour répondre à la demande variable tout en maintenant la rentabilité grâce à la réutilisation par les employés et au déploiement optimisé des applications.


eCommerce
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Whitelabel/OEM/B2C Client

Les logiciels de création de contenu numérique (DCC) traditionnels obligent généralement les utilisateurs à gérer leur propre infrastructure de rendu ou à traiter les rendus localement sur leur poste de travail, ce qui entraîne des investissements matériels importants ou de longs temps d'attente interrompant les flux de travail créatifs. Pour les fournisseurs de logiciels, fournir des fonctionnalités de rendu dans le cloud nécessitait traditionnellement la création et la maintenance d'infrastructures et de systèmes de facturation complexes.

Un déploiement de Deadline Cloud intégré à un logiciel B2C permet un rendu cloud fluide directement dans l'interface familière de l'utilisateur. Cette intégration combine :

- Soumission de tâches de flux de travail interactive intégrée à l'application DCC
- Date limite : canaux conda gérés dans le cloud pour le déploiement d'applications de rendu
- Licences basées sur l'utilisation configurées automatiquement
- Gestion des actifs par le biais de tâches associées à un stockage géré par le fournisseur
- Surveillance personnalisée intégrée directement dans l'interface DCC
- Flottes gérées par des services partagées entre les utilisateurs

Cette approche permet aux utilisateurs finaux de soumettre des rendus au cloud en un seul clic depuis leur logiciel, sans gérer de comptes, d'infrastructure ou de configuration complexe. Le fournisseur de logiciels gère un environnement mutualisé dans lequel :

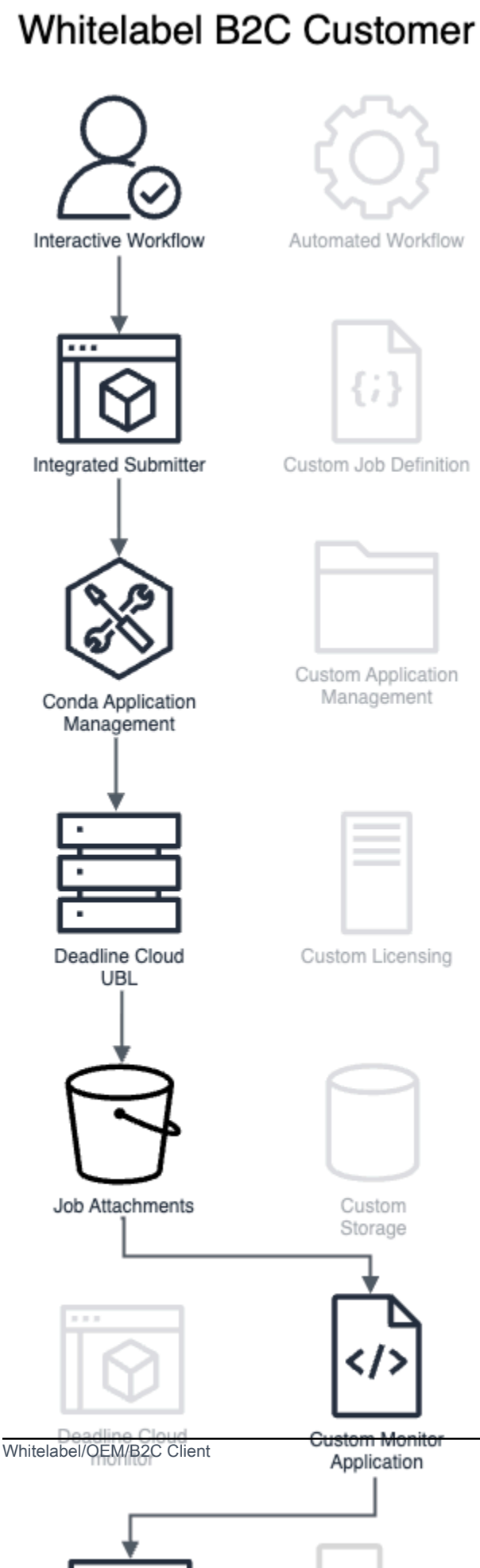
- Les utilisateurs s'authentifient à l'aide de leurs identifiants logiciels existants
- Les tâches sont automatiquement acheminées vers des files d'attente dédiées par utilisateur
- Les actifs sont isolés de manière sécurisée à l'aide de préfixes de stockage contrôlés par IAM
- La facturation est gérée par le biais des systèmes existants du fournisseur
- L'état du job et les résultats sont retransmis directement à l'application de l'utilisateur

L'approche du parc partagé garantit des performances optimales en maintenant un bassin de travailleurs chaleureux, en minimisant les temps de démarrage tout en maximisant l'utilisation des ressources par l'ensemble de la base d'utilisateurs. Cette configuration permet aux fournisseurs de logiciels de proposer le rendu dans le cloud en tant que fonctionnalité fluide du produit plutôt qu'un service distinct nécessitant une configuration ou des comptes supplémentaires.

Les utilisateurs finaux bénéficient des avantages suivants :

- Soumission en un clic depuis leur interface familière
- Pay-as-you-go tarification sans gestion de l'infrastructure
- Délais de démarrage rapides des tâches grâce à une infrastructure partagée
- Téléchargement automatique et organisation des rendus terminés
- Une expérience cohérente sur toutes les plateformes

Ce modèle d'intégration permet aux éditeurs de logiciels de fournir des fonctionnalités de rendu de niveau professionnel à l'ensemble de leur base d'utilisateurs tout en garantissant une expérience simple et conviviale, adaptée à leur application.


Whitelabel B2C Customer
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Qu'est-ce qu'une charge de travail Deadline Cloud

Avec AWS Deadline Cloud, vous pouvez soumettre des tâches pour exécuter vos applications dans le cloud et traiter des données pour la production de contenu ou d'informations cruciales pour votre entreprise. Deadline Cloud utilise Open Job Description (OpenJD) comme syntaxe pour les modèles de tâches, une spécification conçue pour les besoins des pipelines de calcul visuels mais applicable à de nombreux autres cas d'utilisation. Certains exemples de charges de travail incluent le rendu graphique par ordinateur, la simulation physique et la photogrammétrie.

Les charges de travail peuvent aller de simples ensembles de tâches que les utilisateurs soumettent à une file d'attente à l'aide de la CLI ou d'une interface graphique générée automatiquement, à des plug-ins d'envoi intégrés qui génèrent dynamiquement un ensemble de tâches pour une charge de travail définie par l'application.

Comment les charges de travail découlent de la production

Pour comprendre les charges de travail dans les contextes de production et comment les prendre en charge avec Deadline Cloud, réfléchissez à leur évolution. La production peut impliquer la création d'effets visuels, d'animations, de jeux, d'images de catalogues de produits, de reconstructions 3D pour la modélisation des informations du bâtiment (BIM), etc. Ce contenu est généralement créé par une équipe de spécialistes artistiques ou techniques exécutant diverses applications logicielles et des scripts personnalisés. Les membres de l'équipe se transmettent des données via un pipeline de production. De nombreuses tâches effectuées par le pipeline impliquent des calculs intensifs qui peuvent prendre des jours si elles étaient exécutées sur le poste de travail d'un utilisateur.

Voici quelques exemples de tâches dans ces pipelines de production :

- Utilisation d'une application de photogrammétrie pour traiter des photographies prises d'un plateau de tournage afin de reconstruire un maillage numérique texturé.
- Exécution d'une simulation de particules dans une scène 3D pour ajouter des couches de détails à un effet visuel d'explosion pour une émission de télévision.
- Préparation des données d'un niveau de jeu sous la forme requise pour une publication externe et application des paramètres d'optimisation et de compression.
- Rendu d'un ensemble d'images pour un catalogue de produits, y compris les variations de couleur, d'arrière-plan et d'éclairage.
- Exécution d'un script développé sur mesure sur un modèle 3D pour appliquer un look personnalisé et approuvé par un réalisateur.

Ces tâches impliquent de nombreux paramètres à ajuster pour obtenir un résultat artistique ou pour affiner la qualité de sortie. Une interface graphique permet souvent de sélectionner ces valeurs de paramètres à l'aide d'un bouton ou d'un menu pour exécuter le processus localement dans l'application. Lorsqu'un utilisateur exécute le processus, l'application et éventuellement l'ordinateur hôte lui-même ne peuvent pas être utilisés pour effectuer d'autres opérations car ils utilisent l'état de l'application en mémoire et peuvent consommer toutes les ressources du processeur et de la mémoire de l'ordinateur hôte.

Dans de nombreux cas, le processus est rapide. Au cours de la production, la vitesse du processus ralentit lorsque les exigences de qualité et de complexité augmentent. Un test de personnage qui a duré 30 secondes pendant le développement peut facilement se transformer en 3 heures lorsqu'il est appliqué au personnage de production final. Au cours de cette progression, une charge de travail née dans une interface graphique peut devenir trop importante pour être adaptée. Le portage vers Deadline Cloud peut améliorer la productivité des utilisateurs exécutant ces processus, car ils reprennent le contrôle total de leur poste de travail et peuvent suivre les itérations supplémentaires à partir du moniteur Deadline Cloud.

Il existe deux niveaux de support à viser lors du développement du support pour une charge de travail dans Deadline Cloud :

- Décharger la charge de travail du poste de travail de l'utilisateur vers une ferme Deadline Cloud sans parallélisme ni accélération. Cela sous-utilise peut-être les ressources informatiques disponibles dans la ferme, mais la possibilité de transférer de longues opérations vers un système de traitement par lots permet aux utilisateurs d'être plus productifs avec leur propre poste de travail.
- Optimisation du parallélisme de la charge de travail afin qu'elle utilise l'échelle horizontale de la ferme Deadline Cloud pour une exécution rapide.

Parfois, il est évident de savoir comment exécuter une charge de travail en parallèle. Par exemple, chaque image d'un rendu graphique par ordinateur peut être réalisée indépendamment. Il est toutefois important de ne pas rester coincé dans ce parallélisme. Sachez plutôt que le transfert d'une charge de travail de longue durée vers Deadline Cloud présente des avantages considérables, même s'il n'existe aucun moyen évident de répartir la charge de travail.

Les ingrédients d'une charge de travail

Pour spécifier une charge de travail Deadline Cloud, implémentez un ensemble de tâches que les utilisateurs soumettent à une file d'attente avec la CLI de Deadline Cloud. Une grande partie du

travail de création d'un ensemble de tâches consiste à écrire le modèle de tâche, mais d'autres facteurs entrent en ligne de compte, tels que la manière de fournir les applications requises par la charge de travail. Voici les éléments essentiels à prendre en compte lors de la définition d'une charge de travail pour Deadline Cloud :

- L'application à exécuter. La tâche doit être capable de lancer des processus d'application et nécessite donc une installation de l'application disponible ainsi que toutes les licences utilisées par l'application, telles que l'accès à un serveur de licences flottantes. Cela fait généralement partie de la configuration du parc de serveurs et n'est pas intégré au bundle de tâches lui-même.
  - Configuration des tâches à l'aide d'environnements de file d'
  - Connectez les flottes gérées par le client à un point de terminaison de licence
- Définitions des paramètres du job. L'expérience utilisateur lors de la soumission de la tâche est grandement affectée par les paramètres fournis. Les exemples de paramètres incluent les fichiers de données, les répertoires et la configuration des applications.
  - Éléments de valeurs de paramètres pour les ensembles de tâches
- Flux de données de fichiers. Lorsqu'une tâche s'exécute, elle lit les entrées des fichiers fournis par l'utilisateur, puis écrit sa sortie sous forme de nouveaux fichiers. Pour utiliser les pièces jointes aux tâches et les fonctionnalités de mappage de chemins, la tâche doit spécifier les chemins des répertoires ou des fichiers spécifiques pour ces entrées et sorties.
  - Utilisation de fichiers dans le cadre de vos tâches
- Le script d'étape. Le script step exécute le binaire de l'application avec les bonnes options de ligne de commande pour appliquer les paramètres de tâche fournis. Il gère également des détails tels que le mappage des chemins si les fichiers de données de charge de travail incluent des références de chemin absolues plutôt que relatives.
  - Éléments de modèles de tâches pour les offres d'emploi

Portabilité des charges

Une charge de travail est portable lorsqu'elle peut être exécutée sur plusieurs systèmes différents sans la modifier à chaque fois que vous soumettez une tâche. Par exemple, il peut s'exécuter sur différentes fermes de rendu sur lesquelles sont montés différents systèmes de fichiers partagés, ou sur différents systèmes d'exploitation tels que Linux ouWindows. Lorsque vous implémentez un ensemble de tâches portable, il est plus facile pour les utilisateurs d'exécuter le travail sur leur propre parc de serveurs ou de l'adapter à d'autres cas d'utilisation.

Voici quelques moyens de rendre votre offre de tâches portable.

- Spécifiez entièrement les fichiers de données d'entrée nécessaires à une charge de travail, à l'aide des paramètres de PATH tâche et des références aux actifs contenus dans le bundle de tâches. Cette approche permet de transférer le travail vers des fermes basées sur des systèmes de fichiers partagés et vers des fermes qui font des copies des données d'entrée, comme la fonctionnalité de pièces jointes aux tâches de Deadline Cloud.
- Rendez les références de chemin de fichier pour les fichiers d'entrée de la tâche délocalisables et utilisables sur différents systèmes d'exploitation. Par exemple, lorsque les utilisateurs soumettent des tâches depuis des Windows postes de travail pour les exécuter sur une Linux flotte.
  - Utilisez des références de chemin de fichier relatives. Ainsi, si le répertoire qui les contient est déplacé vers un autre emplacement, les références sont toujours résolues. Certaines applications, comme Blender, permettent de choisir entre des chemins relatifs et absolus.
  - Si vous ne pouvez pas utiliser de chemins relatifs, prenez en charge les métadonnées de mappage de chemins OpenJD et traduisez les chemins absolus en fonction de la manière dont Deadline Cloud fournit les fichiers à la tâche.
- Implémentez des commandes dans une tâche à l'aide de scripts portables. Python et bash sont deux exemples de langages de script qui peuvent être utilisés de cette façon. Vous devriez envisager de les fournir tous les deux sur tous les hôtes professionnels de vos flottes.
  - Utilisez le binaire de l'interpréteur de script, comme `python` ou`bash`, avec le nom du fichier de script comme argument. Cette approche fonctionne sur tous les systèmes d'exploitationWindows, y compris, par rapport à l'utilisation d'un fichier script dont le bit d'exécution est activéLinux.
  - Écrivez des scripts bash portables en appliquant les pratiques suivantes :
    - Développez les paramètres de chemin du modèle entre guillemets simples pour gérer les tracés comportant des espaces et des séparateurs de Windows chemin.
    - Lors de l'exécutionWindows, surveillez les problèmes liés à la traduction automatique des chemins dans MinGW. Par exemple, il transforme une AWS CLI commande similaire `aws logs tail /aws/deadline/...` en une commande similaire `aws logs tail "C:/Program Files/Git/aws/deadline/..."` et ne suit pas correctement un journal. Définissez la variable `MSYS_NO_PATHCONV=1` pour désactiver ce comportement.
    - Dans la plupart des cas, le même code fonctionne sur tous les systèmes d'exploitation. Lorsque le code doit être différent, utilisez une `if/else` construction pour gérer les cas.

```
if [[ "$(uname)" == MINGW* || "$(uname -s)" == MSYS_NT* ]]; then
    # Code for Windows
elif [[ "$(uname)" == Darwin ]]; then
    # Code for MacOS
else
```

```
    # Code for Linux and other operating systems
fi
```

  - Vous pouvez écrire des scripts Python portables `pathlib` pour gérer les différences de chemin entre les systèmes de fichiers et éviter les fonctionnalités spécifiques au fonctionnement. La documentation Python inclut des annotations à ce sujet, par exemple dans la documentation de la bibliothèque de signaux. Linux-la prise en charge de fonctionnalités spécifiques est marquée comme « Disponibilité : Linux ».
- Utilisez les paramètres de la tâche pour définir les exigences de l'application. Utilisez des conventions cohérentes que l'administrateur de la ferme peut appliquer dans les environnements de file d'attente.
  - Par exemple, vous pouvez utiliser les `RezPackages` paramètres `CondaPackages` et/ou dans votre tâche, avec une valeur de paramètre par défaut qui répertorie les noms et les versions des packages d'applications requis par la tâche. Vous pouvez ensuite utiliser l'un des exemples d'environnements de file d'attente Conda ou Rez pour fournir un environnement virtuel pour la tâche.

Commencer à utiliser les ressources de Deadline Cloud

Pour commencer à créer des solutions personnalisées pour AWS Deadline Cloud, vous devez configurer vos ressources. Il s'agit notamment d'une ferme, d'au moins une file d'attente pour la ferme et d'au moins un parc de travailleurs pour desservir la file d'attente. Vous pouvez créer vos ressources à l'aide de la console Deadline Cloud, ou vous pouvez utiliser le AWS Command Line Interface.

Dans ce didacticiel, vous allez AWS CloudShell créer une ferme de développement simple et exécuter l'agent de travail. Vous pouvez ensuite soumettre et exécuter une tâche simple avec des paramètres et des pièces jointes, ajouter un parc géré par des services et nettoyer les ressources de votre ferme lorsque vous avez terminé.

Les sections suivantes vous présentent les différentes fonctionnalités de Deadline Cloud, ainsi que la façon dont elles fonctionnent et fonctionnent ensemble. Il est utile de suivre ces étapes pour développer et tester de nouvelles charges de travail et personnalisations.

Pour obtenir des instructions sur la configuration de votre ferme à l'aide de la console, consultez Getting started dans le guide de l'utilisateur de Deadline Cloud.

Rubriques

Création d'un parc Deadline Cloud

Pour créer votre parc de développeurs et vos ressources de file d'attente dans AWS Deadline Cloud, utilisez le AWS Command Line Interface (AWS CLI), comme indiqué dans la procédure suivante. Vous allez également créer un rôle Gestion des identités et des accès AWS (IAM) et une flotte gérée par le client (CMF) et associer la flotte à votre file d'attente. Vous pouvez ensuite configurer AWS CLI et confirmer que votre ferme est configurée et fonctionne comme indiqué.

Vous pouvez utiliser cette ferme pour explorer les fonctionnalités de Deadline Cloud, puis développer et tester de nouvelles charges de travail, personnalisations et intégrations de pipelines.

Pour créer une ferme

1. Ouvrez une AWS CloudShell session. Vous allez utiliser la CloudShell fenêtre pour saisir les commandes AWS Command Line Interface (AWS CLI) afin d'exécuter les exemples de ce didacticiel. Gardez la CloudShell fenêtre ouverte pendant que vous poursuivez.
2. Créez un nom pour votre ferme et ajoutez-y le nom de la ferme à`~/.bashrc`. Cela le rendra disponible pour les autres sessions du terminal.

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. Créez la ressource agricole et ajoutez-y son identifiant de ferme`~/.bashrc`.

```
aws deadline create-farm \
    --display-name "$DEV_FARM_NAME"

echo "DEV_FARM_ID=\$(aws deadline list-farms \
        --query \"farms[?displayName=='\$DEV_FARM_NAME'].farmId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. Créez la ressource de file d'attente et ajoutez son ID de file d'attente à `~/.bashrc`.

```
aws deadline create-queue \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME Queue" \
    --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
 "runAs":"QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=\$(aws deadline list-queues \
        --farm-id \$DEV_FARM_ID \
        --query \"queues[?displayName=='\$DEV_FARM_NAME Queue'].queueId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. Créez un rôle IAM pour la flotte. Ce rôle fournit aux hôtes de travail de votre flotte les informations de sécurité nécessaires pour exécuter des tâches depuis votre file d'attente.

```
aws iam create-role \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --policy-name WorkerPermissions \
    --policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "deadline:AssumeFleetRoleForWorker",
                        "deadline:UpdateWorker",
                        "deadline:DeleteWorker",
                        "deadline:UpdateWorkerSchedule",
                        "deadline:BatchGetJobEntity",
                        "deadline:AssumeQueueRoleForWorker"
                    ],
                    "Resource": "*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:CreateLogStream"
```

```
                ],
                "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                "Condition": {
                    "StringEquals": {
                        "aws:PrincipalAccount": "${aws:ResourceAccount}"
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "logs:PutLogEvents",
                    "logs:GetLogEvents"
                ],
                "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                "Condition": {
                    "StringEquals": {
                        "aws:PrincipalAccount": "${aws:ResourceAccount}"
                    }
                }
            }
        ]
    }'
```

6. Créez la flotte gérée par le client (CMF) et ajoutez son identifiant de flotte à. `~/.bashrc`

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME CMF" \
    --role-arn $FLEET_ROLE_ARN \
    --max-worker-count 5 \
    --configuration \
        '{
            "customerManaged": {
                "mode": "NO_SCALING",
                "workerCapabilities": {
                    "vCpuCount": {"min": 1},
                    "memoryMiB": {"min": 512},
                    "osFamily": "linux",
                    "cpuArchitectureType": "x86_64"
                }
            }
```

```
        }'

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
        --farm-id \$DEV_FARM_ID \
        --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. Associez le CMF à votre file d'attente.

```
aws deadline create-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_CMF_ID
```

8. Installez l'interface de ligne de commande de Deadline Cloud.

```
pip install deadline
```

9. Pour définir le parc par défaut sur l'ID du parc de serveurs et la file d'attente sur l'ID de file d'attente que vous avez créé précédemment, utilisez la commande suivante.

```
deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID
```

10. (Facultatif) Pour vérifier que votre ferme est configurée conformément à vos spécifications, utilisez les commandes suivantes :

    - Liste de toutes les fermes — **`deadline farm list`**
    - Répertorier toutes les files d'attente de la ferme par défaut — **`deadline queue list`**
    - Répertorier toutes les flottes de la ferme par défaut : **`deadline fleet list`**
    - Obtenez la ferme par défaut — **`deadline farm get`**
    - Obtenez la file d'attente par défaut — **`deadline queue get`**
    - Obtenez toutes les flottes associées à la file d'attente par défaut — **`deadline fleet get`**

Étapes suivantes

Après avoir créé votre parc de serveurs, vous pouvez exécuter l'agent de travail Deadline Cloud sur les hôtes de votre parc pour traiter les tâches. Consultez Exécutez l'agent de travail Deadline Cloud.

Exécutez l'agent de travail Deadline Cloud

Avant de pouvoir exécuter les tâches que vous soumettez à la file d'attente de votre parc de développeurs, vous devez exécuter l'agent de travail de AWS Deadline Cloud en mode développeur sur un hôte de travail.

Dans le reste de ce didacticiel, vous allez effectuer des AWS CLI opérations sur votre ferme de développeurs à l'aide de deux AWS CloudShell onglets. Dans le premier onglet, vous pouvez soumettre des offres d'emploi. Dans le deuxième onglet, vous pouvez exécuter l'agent de travail.

> **Note**
>
> Si vous laissez votre CloudShell session inactive pendant plus de 20 minutes, le délai expirera et l'agent de travail sera arrêté. Pour redémarrer l'agent de travail, suivez les instructions de la procédure suivante.

Avant de créer un agent de travail, vous devez configurer un parc, une file d'attente et un parc Deadline Cloud. Consultez Création d'un parc Deadline Cloud.

Pour exécuter l'agent de travail en mode développeur

1. Votre ferme étant toujours ouverte dans le premier CloudShell onglet, ouvrez un deuxième CloudShell onglet, puis créez les `demoenv-persist` répertoires `demoenv-logs` et.

   ```
   mkdir ~/demoenv-logs
   mkdir ~/demoenv-persist
   ```

2. Téléchargez et installez les packages d'agents de travail de Deadline Cloud depuis PyPI :

   > **Note**
   >
   > Activé Windows, les fichiers de l'agent doivent être installés dans le répertoire global site-packages de Python. Les environnements virtuels Python ne sont actuellement pas pris en charge.

   ```
   python -m pip install deadline-cloud-worker-agent
   ```

3. Pour permettre à l'agent de travail de créer les répertoires temporaires pour exécuter les tâches, créez un répertoire :

```
sudo mkdir /sessions
sudo chmod 750 /sessions
sudo chown cloudshell-user /sessions
```

4. Exécutez l'agent de travail de Deadline Cloud en mode développeur avec les variables `DEV_FARM_ID` et `DEV_CMF_ID` celles que vous avez ajoutées au`~/.bashrc`.

```
deadline-worker-agent \
    --farm-id $DEV_FARM_ID \
    --fleet-id $DEV_CMF_ID \
    --run-jobs-as-agent-user \
    --logs-dir ~/demoenv-logs \
    --persistence-dir ~/demoenv-persist
```

Lorsque l'agent de travail initialise puis interroge l'opération d'`UpdateWorkerSchedule`API, le résultat suivant s'affiche :

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
Python Version: 3.9.16 (main, Sep  8 2023, 00:00:00)  - [GCC 11.4.1 20230605 (Red
 Hat 11.4.1-2)]
Platform: linux
...
[2024-03-27 15:51:02,528][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
 'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

5. Sélectionnez votre premier CloudShell onglet, puis listez les travailleurs de la flotte.

```
deadline worker list --fleet-id $DEV_CMF_ID
```

Des résultats tels que les suivants sont affichés :

```
Displaying 1 of 1 workers starting at 0

- workerId: worker-8c9af877c8734e89914047111f
  status: STARTED
  createdAt: 2023-12-13 20:43:06+00:00
```

Dans une configuration de production, l'agent de travail de Deadline Cloud nécessite la configuration de plusieurs utilisateurs et répertoires de configuration en tant qu'utilisateur administratif sur la machine hôte. Vous pouvez annuler ces paramètres car vous exécutez des tâches dans votre propre ferme de développement, à laquelle vous seul pouvez accéder.

Étapes suivantes

Maintenant qu'un agent de travail est en cours d'exécution sur vos hôtes de travail, vous pouvez envoyer des tâches à vos employés. Vous pouvez :

- Soumettre avec Deadline Clouden utilisant un simple bundle de tâches OpenJD.
- Soumettez des offres d'emploi avec des pièces jointes dans Deadline Cloudqui partagent des fichiers entre des postes de travail utilisant différents systèmes d'exploitation.

Soumettre avec Deadline Cloud

Pour exécuter des tâches Deadline Cloud sur vos hôtes de travail, vous devez créer et utiliser un ensemble de tâches Open Job Description (OpenJD) pour configurer une tâche. Le bundle configure la tâche, par exemple en spécifiant les fichiers d'entrée pour une tâche et en indiquant où écrire la sortie de la tâche. Cette rubrique contient des exemples de méthodes permettant de configurer un ensemble de tâches.

Avant de pouvoir suivre les procédures décrites dans cette section, vous devez effectuer les opérations suivantes :

- Création d'un parc Deadline Cloud
- Exécutez l'agent de travail Deadline Cloud

Pour utiliser AWS Deadline Cloud pour exécuter des tâches, suivez les procédures suivantes. Utilisez le premier AWS CloudShell onglet pour soumettre des offres d'emploi à votre parc de développeurs. Utilisez le deuxième CloudShell onglet pour afficher la sortie de l'agent de travail.

Rubriques

Soumettre l'simple_jobéchantillon

Après avoir créé une ferme et exécuté l'agent de travail, vous pouvez envoyer l'simple_jobéchantillon à Deadline Cloud.

Pour envoyer l'simple_jobéchantillon à Deadline Cloud

1. Choisissez votre premier CloudShell onglet.
2. Téléchargez l'exemple à partir de GitHub.

```
cd ~
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. Accédez au répertoire des exemples de lots de tâches.

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. Soumettez l'simple_jobéchantillon.

```
deadline bundle submit simple_job
```

5. Choisissez votre deuxième CloudShell onglet pour afficher les résultats de journalisation concernant les appels`BatchGetJobEntities`, l'obtention d'une session et l'exécution d'une action de session.

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
 # [session-053d77cef82648fe2] Starting new Session.
 [queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
```

```
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
 resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
 'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
 'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
 {'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
 75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
 params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
 'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'},
 'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
 '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
 request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
 [session-053d77cef82648fea9c69827182] Appended new SessionActions.
 (ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
 [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
 [session-053d77cef82648fea9c69827182] Running as the Worker Agent's
 user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
 [session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
 no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
 Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: local
 file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

> **Note**
>
> Seule la sortie de journalisation de l'agent de travail est affichée. Il existe un journal distinct pour la session qui exécute le travail.

6. Choisissez votre premier onglet, puis inspectez les fichiers journaux écrits par l'agent de travail.

   a. Accédez au répertoire des journaux de l'agent de travail et visualisez son contenu.

   ```
   cd ~/demoenv-logs
   ls
   ```

   b. Imprimez le premier fichier journal créé par l'agent de travail.

   ```
   cat worker-agent-bootstrap.log
   ```

   Ce fichier contient une sortie de l'agent de travail expliquant comment il a appelé l'API Deadline Cloud pour créer une ressource de personnel dans votre flotte, puis a assumé le rôle de flotte.

   c. Imprimez la sortie du fichier journal lorsque l'agent des travailleurs rejoint le parc.

   ```
   cat worker-agent.log
   ```

   Ce journal contient des résultats sur toutes les actions entreprises par l'agent de travail, mais pas sur les files d'attente à partir desquelles il exécute les tâches, à l'exception IDs de ces ressources.

   d. Imprimez les fichiers journaux de chaque session dans un répertoire dont le nom est identique à l'identifiant de la ressource de file d'attente.

   ```
   cat $DEV_QUEUE_ID/session-*.log
   ```

   En cas de réussite de la tâche, le résultat du fichier journal sera similaire à ce qui suit :

   ```
   cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)

   2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
   2024-03-27 16:00:22,404 INFO
   2024-03-27 16:00:22,405 INFO ================================================
   2024-03-27 16:00:22,405 INFO --------- Running Task
   2024-03-27 16:00:22,405 INFO ================================================
   2024-03-27 16:00:22,406 INFO ------------------------------------------------
   2024-03-27 16:00:22,406 INFO Phase: Setup
   2024-03-27 16:00:22,406 INFO ------------------------------------------------
   2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
   ```

```
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO --------- Session Cleanup
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a
```

7. Imprimez les informations relatives à la tâche.

   ```
   deadline job get
   ```

   Lorsque vous soumettez la tâche, le système l'enregistre par défaut afin que vous n'ayez pas à saisir l'ID de la tâche.

Soumettre un simple_job avec un paramètre

Vous pouvez soumettre des tâches avec des paramètres. Dans la procédure suivante, vous modifiez le simple_job modèle pour inclure un message personnalisé, vous soumettez le fichier journal de sessionsimple_job, puis vous l'imprimez pour afficher le message.

Pour soumettre l'simple_jobéchantillon avec un paramètre

1. Sélectionnez votre premier CloudShell onglet, puis accédez au répertoire des exemples de lots de tâches.

   ```
   cd ~/deadline-cloud-samples/job_bundles/
   ```

2. Imprimez le contenu du simple_job modèle.

```
cat simple_job/template.yaml
```

La `parameterDefinitions` section contenant le `Message` paramètre doit ressembler à ce qui suit :

```
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
```

3. Soumettez l'simple_jobexemple avec une valeur de paramètre, puis attendez que le travail soit terminé.

```
deadline bundle submit simple_job \
    -p "Message=Greetings from the developer getting started guide."
```

4. Pour voir le message personnalisé, consultez le fichier journal de session le plus récent.

```
cd ~/demoenv-logs
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

Création d'un ensemble de tâches simple_file_job avec des E/S de fichiers

Une tâche de rendu doit lire la définition de la scène, en faire le rendu d'une image, puis enregistrer cette image dans un fichier de sortie. Vous pouvez simuler cette action en demandant à la tâche de calculer le hachage de l'entrée au lieu de restituer une image.

Pour créer un ensemble de tâches simple_file_job avec des E/S de fichiers

1. Sélectionnez votre premier CloudShell onglet, puis accédez au répertoire des exemples de lots de tâches.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Faites-en une copie `simple_job` avec le nouveau nom`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Modifiez le modèle de tâche comme suit :

> **Note**
>
> Nous vous recommandons de l'utiliser nano pour ces étapes. Si vous préférez l'utiliserVim, vous devez définir son mode de collage à l'aide de`:set paste`.

a. Ouvrez le modèle dans un éditeur de texte.

```
nano simple_file_job/template.yaml
```

b. Ajoutez les éléments suivants `typeobjectType`, et `dataFlowparameterDefinitions`.

```
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
```

c. Ajoutez la commande de `bash` script suivante à la fin du fichier qui lit le fichier d'entrée et écrit dans le fichier de sortie.

```
        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

La mise à jour `template.yaml` doit correspondre exactement à ce qui suit :

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
```

```
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      runnable: true
      data: |
        #!/usr/bin/env bash
        echo "{{Param.Message}}"

        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

> **Note**
>
> Si vous souhaitez ajuster l'espacement dans le`template.yaml`, assurez-vous d'utiliser des espaces plutôt que des indentations.

d. Enregistrez le fichier et quittez l'éditeur de texte.

4. Fournissez des valeurs de paramètres pour les fichiers d'entrée et de sortie afin de soumettre le simple_file_job.

```
deadline bundle submit simple_file_job \
    -p "InFile=simple_job/template.yaml" \
    -p "OutFile=hash.txt"
```

5. Imprimez les informations relatives à la tâche.

```
deadline job get
```

- Vous verrez des résultats tels que les suivants :

```
parameters:
```

```
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/
template.yaml
  OutFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- Bien que vous n'ayez fourni que des chemins relatifs, le chemin complet est défini pour les paramètres. Le AWS CLI joint le répertoire de travail actuel à tous les chemins fournis en tant que paramètres lorsque les chemins ont le typePATH.
- L'agent de travail exécuté dans l'autre fenêtre du terminal prend en charge et exécute la tâche. Cette action crée le `hash.txt` fichier, que vous pouvez consulter à l'aide de la commande suivante.

  ```
  cat hash.txt
  ```

  Cette commande imprimera un résultat similaire à ce qui suit.

  ```
  eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/
  cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
  ```

Étapes suivantes

Après avoir appris à soumettre des tâches simples à l'aide de la CLI de Deadline Cloud, vous pouvez explorer les points suivants :

- Soumettez des offres d'emploi avec des pièces jointes dans Deadline Cloudpour savoir comment exécuter des tâches sur des hôtes exécutant différents systèmes d'exploitation.
- Ajoutez une flotte gérée par des services à votre parc de développeurs dans Deadline Cloudpour exécuter vos tâches sur des hôtes gérés par Deadline Cloud.
- Nettoyez les ressources de votre ferme dans Deadline Cloudpour arrêter les ressources que vous avez utilisées pour ce didacticiel.

Soumettez des offres d'emploi avec des pièces jointes dans Deadline Cloud

De nombreuses fermes utilisent des systèmes de fichiers partagés pour partager des fichiers entre les hôtes qui soumettent des tâches et ceux qui exécutent des tâches. Par exemple, dans l'`simple_file_job`exemple précédent, le système de fichiers local est partagé entre les fenêtres du AWS CloudShell terminal, qui s'exécutent dans l'onglet 1 où vous soumettez le travail, et dans l'onglet 2 où vous exécutez l'agent de travail.

Un système de fichiers partagé est avantageux lorsque le poste de travail émetteur et les hôtes de travail se trouvent sur le même réseau local. Si vous stockez vos données sur site à proximité des postes de travail qui y accèdent, l'utilisation d'une ferme basée sur le cloud signifie que vous devez partager vos systèmes de fichiers via un VPN à latence élevée ou synchroniser vos systèmes de fichiers dans le cloud. Aucune de ces options n'est facile à configurer ou à utiliser.

AWS Deadline Cloud propose une solution simple avec des pièces jointes à des tâches, similaires à des pièces jointes à des e-mails. Avec les pièces jointes à une tâche, vous associez des données à votre tâche. Deadline Cloud gère ensuite les détails du transfert et du stockage de vos données de travail dans des compartiments Amazon Simple Storage Service (Amazon S3).

Les processus de création de contenu sont souvent itératifs, ce qui signifie qu'un utilisateur soumet des tâches avec un petit sous-ensemble de fichiers modifiés. Comme les compartiments Amazon S3 stockent les pièces jointes aux tâches dans un espace de stockage adressable par le contenu, le nom de chaque objet est basé sur le hachage des données de l'objet et le contenu d'une arborescence de répertoires est stocké dans un format de fichier manifeste joint à une tâche.

Avant de pouvoir suivre les procédures décrites dans cette section, vous devez effectuer les opérations suivantes :

- Création d'un parc Deadline Cloud
- Exécutez l'agent de travail Deadline Cloud

Pour exécuter des tâches avec des pièces jointes, procédez comme suit.

Rubriques

- Ajoutez une configuration de pièces jointes aux tâches à votre file d'attente
- Soumettre simple_file_job avec des pièces jointes au poste

- Comprendre comment les pièces jointes aux tâches sont stockées dans Amazon S3
- Étapes suivantes

Ajoutez une configuration de pièces jointes aux tâches à votre file d'attente

Pour activer les pièces jointes aux tâches dans votre file d'attente, ajoutez une configuration de pièces jointes aux tâches à la ressource de file d'attente de votre compte.

Pour ajouter une configuration de pièces jointes aux tâches à votre file d'attente

1. Choisissez votre premier CloudShell onglet, puis entrez l'une des commandes suivantes pour utiliser un compartiment Amazon S3 pour les pièces jointes aux tâches.
   - Si vous n'avez pas de compartiment Amazon S3 privé existant, vous pouvez créer et utiliser un nouveau compartiment S3.

     ```
     DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
         | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
     if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
     else LOCATION_CONSTRAINT="--create-bucket-configuration \
         LocationConstraint=${AWS_REGION}"
     fi
     aws s3api create-bucket \
         $LOCATION_CONSTRAINT \
         --acl private \
         --bucket ${DEV_FARM_BUCKET}
     ```

   - Si vous possédez déjà un compartiment Amazon S3 privé, vous pouvez l'utiliser en le *MY_BUCKET_NAME* remplaçant par le nom de votre compartiment.

     ```
     DEV_FARM_BUCKET=MY_BUCKET_NAME
     ```

2. Après avoir créé ou choisi votre compartiment Amazon S3, ajoutez le nom du compartiment `~/.bashrc` pour le rendre disponible pour d'autres sessions de terminal.

   ```
   echo "DEV_FARM_BUCKET=$DEV_FARM_BUCKET" >> ~/.bashrc
    source ~/.bashrc
   ```

3. Créez un rôle Gestion des identités et des accès AWS (IAM) pour la file d'attente.

   ```
   aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
   ```

```
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess \
    --policy-document \
            '{
                "Version": "2012-10-17",
                "Statement": [
                {
                    "Action": [
                        "s3:GetObject*",
                        "s3:GetBucket*",
                        "s3:List*",
                        "s3:DeleteObject*",
                        "s3:PutObject",
                        "s3:PutObjectLegalHold",
                        "s3:PutObjectRetention",
                        "s3:PutObjectTagging",
                        "s3:PutObjectVersionTagging",
                        "s3:Abort*"
                    ],
                    "Resource": [
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'",
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
                    ],
                    "Effect": "Allow"
                }
            ]
            }'
```

4. Mettez à jour votre file d'attente pour inclure les paramètres des pièces jointes aux tâches et le rôle IAM.

```
QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --role-arn $QUEUE_ROLE_ARN \
    --job-attachment-settings \
        '{
            "s3BucketName": "'$DEV_FARM_BUCKET'",
            "rootPrefix": "JobAttachments"
        }'
```

5. Confirmez que vous avez mis à jour votre file d'attente.

```
deadline queue get
```

Des résultats tels que les suivants sont affichés :

```
...
jobAttachmentSettings:
  s3BucketName: DEV_FARM_BUCKET
  rootPrefix: JobAttachments
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole
...
```

Soumettre simple_file_job avec des pièces jointes au poste

Lorsque vous utilisez des pièces jointes à des tâches, les ensembles de tâches doivent fournir à Deadline Cloud suffisamment d'informations pour déterminer le flux de données de la tâche, par exemple en utilisant des PATH paramètres. Dans ce cassimple_file_job, vous avez modifié le `template.yaml` fichier pour indiquer à Deadline Cloud que le flux de données se trouve dans le fichier d'entrée et le fichier de sortie.

Après avoir ajouté la configuration des pièces jointes aux tâches à votre file d'attente, vous pouvez envoyer l'exemple simple_file_job avec les pièces jointes aux tâches. Ensuite, vous pouvez consulter le journal et le résultat de la tâche pour confirmer que les pièces jointes simple_file_job aux tâches fonctionnent.

Pour soumettre le bundle de tâches simple_file_job avec des pièces jointes

1. Choisissez votre premier CloudShell onglet, puis ouvrez le `JobBundle-Samples` répertoire.
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```
3. Soumettez simple_file_job à la file d'attente. Lorsque vous êtes invité à confirmer le téléchargement, entrez**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```
4. Pour afficher le résultat du journal de session de transfert de données des pièces jointes aux tâches, exécutez la commande suivante.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```
5. Répertoriez les actions de session exécutées au cours de la session.

```
aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID
```

Des résultats tels que les suivants sont affichés :

```
{
    "sessionactions": [
        {
            "sessionActionId": "sessionaction-123-0",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
```

```
            "progressPercent": 100.0,
            "definition": {
                "syncInputJobAttachments": {}
            }
        },
        {
            "sessionActionId": "sessionaction-123-1",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "taskRun": {
                    "taskId": "task-abc-0",
                    "stepId": "step-def"
                }
            }
        }
    ]
}
```

La première action de session a téléchargé les pièces jointes de tâche d'entrée, tandis que la seconde action exécute la tâche comme dans les étapes précédentes, puis a chargé les pièces jointes de tâche de sortie.

6. Répertoriez le répertoire de sortie.

```
ls *.txt
```

Une sortie telle qu'`hash.txt`elle existe dans le répertoire, mais `hash-jobattachments.txt` n'existe pas car le fichier de sortie de la tâche n'a pas encore été téléchargé.

7. Téléchargez le résultat de la tâche la plus récente.

```
deadline job download-output
```

8. Affichez le résultat du fichier téléchargé.

```
cat hash-jobattachments.txt
```

Des résultats tels que les suivants sont affichés :

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

Comprendre comment les pièces jointes aux tâches sont stockées dans Amazon S3

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour charger ou télécharger des données relatives aux pièces jointes aux tâches, qui sont stockées dans des compartiments Amazon S3. Comprendre comment Deadline Cloud stocke les pièces jointes aux tâches sur Amazon S3 vous aidera à développer des charges de travail et à intégrer des pipelines.

Pour vérifier comment les pièces jointes aux tâches de Deadline Cloud sont stockées dans Amazon S3

1. Choisissez votre premier CloudShell onglet, puis ouvrez le répertoire des exemples de lots de tâches.

   ```
   cd ~/deadline-cloud-samples/job_bundles/
   ```

2. Inspectez les propriétés de la tâche.

   ```
   deadline job get
   ```

   Des résultats tels que les suivants sont affichés :

   ```
   parameters:
     Message:
       string: Welcome to AWS Deadline Cloud!
     InFile:
       path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
   template.yaml
     OutFile:
       path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
   jobattachments.txt
   attachments:
     manifests:
     - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
       rootPathFormat: posix
   ```

```
    outputRelativeDirectories:
    - .
    inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
    inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
  fileSystem: COPIED
```

Le champ Pièces jointes contient une liste de structures de manifeste qui décrivent les chemins de données d'entrée et de sortie utilisés par la tâche lors de son exécution. Regardez `rootPath` le chemin du répertoire local sur la machine qui a soumis le travail. Pour consulter le suffixe d'objet Amazon S3 qui contient un fichier manifeste, consultez le`inputManifestFile`. Le fichier manifeste contient des métadonnées pour un instantané de l'arborescence des répertoires des données d'entrée de la tâche.

3. Imprimez joliment l'objet manifeste Amazon S3 pour voir la structure du répertoire d'entrée correspondant à la tâche.

```
MANIFEST_SUFFIX=$(aws deadline get-job \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --job-id $JOB_ID \
     --query "attachments.manifests[0].inputManifestPath" \
     --output text)
 aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .
```

Des résultats tels que les suivants sont affichés :

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "2ec297b04c59c4741ed97ac8fb83080c",
         "mtime": 1698186190000000,
         "path": "simple_job/template.yaml",
         "size": 445
     }
     ],
     "totalSize": 445
 }
```

4. Créez le préfixe Amazon S3 qui contient les manifestes pour les pièces jointes aux tâches de sortie et listez l'objet situé en dessous.

```
SESSION_ACTION=$(aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID \
    --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/
$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX
```

   Les pièces jointes aux tâches de sortie ne sont pas directement référencées à partir de la ressource de tâche, mais sont placées dans un compartiment Amazon S3 basé sur les ressources de la ferme IDs.

5. Obtenez la clé d'objet manifeste la plus récente pour l'identifiant d'action de session spécifique, puis imprimez joliment les objets du manifeste.

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionActionId)
 MANIFEST_KEY=$(aws s3api list-objects-v2 \
     --bucket $DEV_FARM_BUCKET \
     --prefix $TASK_OUTPUT_PREFIX \
     --query "Contents[*].Key" --output text \
     | grep $SESSION_ACTION_ID \
     | sort | tail -1)
 MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
 echo $MANIFEST_OBJECT | jq .
```

   Vous verrez les propriétés du fichier `hash-jobattachments.txt` dans la sortie, telles que les suivantes :

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
```

```
        "mtime": 1698785252554950,
        "path": "hash-jobattachments.txt",
        "size": 182
    }
    ],
    "totalSize": 182
}
```

Votre tâche ne comportera qu'un seul objet manifeste par exécution de tâche, mais en général, il est possible d'avoir plus d'objets par exécution de tâche.

6. Affichez la sortie de stockage Amazon S3 adressable au contenu sous le préfixe. `Data`

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

Des résultats tels que les suivants sont affichés :

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

Étapes suivantes

Après avoir appris à soumettre des tâches avec pièces jointes à l'aide de la CLI de Deadline Cloud, vous pouvez découvrir :

- Soumettre avec Deadline Cloudpour savoir comment exécuter des tâches à l'aide d'un bundle OpenJD sur vos hôtes de travail.
- Ajoutez une flotte gérée par des services à votre parc de développeurs dans Deadline Cloudpour exécuter vos tâches sur des hôtes gérés par Deadline Cloud.
- Nettoyez les ressources de votre ferme dans Deadline Cloudpour arrêter les ressources que vous avez utilisées pour ce didacticiel.

Ajoutez une flotte gérée par des services à votre parc de développeurs dans Deadline Cloud

AWS CloudShell ne fournit pas une capacité de calcul suffisante pour tester des charges de travail plus importantes. Il n'est pas non plus configuré pour fonctionner avec des tâches qui distribuent des tâches sur plusieurs hôtes de travail.

Au lieu de l'utiliser CloudShell, vous pouvez ajouter une flotte gérée par des services (SMF) Auto Scaling à votre parc de développeurs. Un SMF fournit une capacité de calcul suffisante pour des charges de travail plus importantes et peut gérer des tâches nécessitant de répartir les tâches entre plusieurs hôtes de travail.

Avant d'ajouter un SMF, vous devez configurer un parc, une file d'attente et un parc Deadline Cloud. Consultez Création d'un parc Deadline Cloud.

Pour ajouter une flotte gérée par des services à votre parc de développeurs

1. Choisissez votre premier AWS CloudShell onglet, puis créez le parc géré par le service et ajoutez-y son identifiant de flotte. `.bashrc` Cette action le rend disponible pour d'autres sessions de terminal.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
         --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
 aws deadline create-fleet \
     --farm-id $DEV_FARM_ID \
     --display-name "$DEV_FARM_NAME SMF" \
     --role-arn $FLEET_ROLE_ARN \
     --max-worker-count 5 \
     --configuration \
         '{
             "serviceManagedEc2": {
                 "instanceCapabilities": {
                     "vCpuCount": {
                         "min": 2,
                         "max": 4
                     },
                     "memoryMiB": {
                         "min": 512
                     },
                     "osFamily": "linux",
                     "cpuArchitectureType": "x86_64"
```

```
                },
                "instanceMarketOptions": {
                    "type": "spot"
                }
            }
        }'

 echo "DEV_SMF_ID=$(aws deadline list-fleets \
        --farm-id $DEV_FARM_ID \
        --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
        | [0]" --output text)" >> ~/.bashrc
 source ~/.bashrc
```

2. Associez le SMF à votre file d'attente.

```
aws deadline create-queue-fleet-association \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --fleet-id $DEV_SMF_ID
```

3. Soumettre simple_file_job à la file d'attente. Lorsque vous êtes invité à confirmer le téléchargement, entrez**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. Vérifiez que le SMF fonctionne correctement.

```
deadline fleet get
```

   - Le travailleur peut mettre quelques minutes à démarrer. Répétez la `deadline fleet get` commande jusqu'à ce que vous voyiez que la flotte fonctionne.
   - La flotte `queueFleetAssociationsStatus` destinée à la gestion des services sera. `ACTIVE`
   - Le SMF `autoScalingStatus` passera de `GROWING` à. `STEADY`

   Votre statut ressemblera à ce qui suit :

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
```

```
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. Consultez le journal de la tâche que vous avez soumise. Ce journal est stocké dans un journal dans Amazon CloudWatch Logs, et non dans le système de CloudShell fichiers.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
    --log-stream-names $SESSION_ID
```

Étapes suivantes

Après avoir créé et testé un parc géré par des services, vous devez supprimer les ressources que vous avez créées pour éviter des frais inutiles.

- Nettoyez les ressources de votre ferme dans Deadline Cloudpour arrêter les ressources que vous avez utilisées pour ce didacticiel.

Nettoyez les ressources de votre ferme dans Deadline Cloud

Pour développer et tester de nouvelles charges de travail et de nouvelles intégrations de pipeline, vous pouvez continuer à utiliser le parc de développeurs Deadline Cloud que vous avez créé pour ce didacticiel. Si vous n'avez plus besoin de votre parc de développeurs, vous pouvez supprimer ses ressources, notamment la ferme, le parc, la file d'attente, les rôles Gestion des identités et des accès AWS (IAM) et les journaux dans Amazon CloudWatch Logs. Après avoir supprimé ces ressources,

vous devrez recommencer le didacticiel pour pouvoir les utiliser. Pour de plus amples informations, veuillez consulter Commencer à utiliser les ressources de Deadline Cloud.

Pour assainir les ressources agricoles des développeurs

1. Choisissez votre premier CloudShell onglet, puis arrêtez toutes les associations de files d'attente et de flottes pour votre file d'attente.

```
FLEETS=$(aws deadline list-queue-fleet-associations \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --query "queueFleetAssociations[].fleetId" \
        --output text)
for FLEET_ID in $FLEETS; do
    aws deadline update-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID \
        --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

2. Répertoriez les associations de parcs de files d'attente.

```
aws deadline list-queue-fleet-associations \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

Vous devrez peut-être réexécuter la commande jusqu'à ce que le résultat soit `"status": "STOPPED"` affiché, puis vous pourrez passer à l'étape suivante. Ce processus peut prendre plusieurs minutes.

```
{
    "queueFleetAssociations": [
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:49:19+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:38+00:00",
```

```
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        },
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:32:06+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:39+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        }
    ]
}
```

3. Supprimez toutes les associations de files d'attente et de flottes associées à votre file d'attente.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-queue-fleet-association \
         --farm-id $DEV_FARM_ID \
         --queue-id $DEV_QUEUE_ID \
         --fleet-id $FLEET_ID
 done
```

4. Supprimez toutes les flottes associées à votre file d'attente.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-fleet \
         --farm-id $DEV_FARM_ID \
         --fleet-id $FLEET_ID
 done
```

5. Supprimez la file d'attente.

```
aws deadline delete-queue \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

6. Supprimez la ferme.

```
aws deadline delete-farm \
```

```
    --farm-id $DEV_FARM_ID
```

7. Supprimez les autres AWS ressources de votre ferme.

   a. Supprimez le rôle de flotte Gestion des identités et des accès AWS (IAM).

   ```
   aws iam delete-role-policy \
        --role-name "${DEV_FARM_NAME}FleetRole" \
        --policy-name WorkerPermissions
   aws iam delete-role \
        --role-name "${DEV_FARM_NAME}FleetRole"
   ```

   b. Supprimez le rôle IAM de la file d'attente.

   ```
   aws iam delete-role-policy \
        --role-name "${DEV_FARM_NAME}QueueRole" \
        --policy-name S3BucketsAccess
   aws iam delete-role \
        --role-name "${DEV_FARM_NAME}QueueRole"
   ```

   c. Supprimez les groupes de CloudWatch journaux Amazon Logs. Chaque file d'attente et flotte possède son propre groupe de journaux.

   ```
   aws logs delete-log-group \
        --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"
   aws logs delete-log-group \
        --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"
   aws logs delete-log-group \
        --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
   ```

Créez des jobs à soumettre à Deadline Cloud

Vous soumettez des offres d'emploi à Deadline Cloud à l'aide de lots de tâches. Un ensemble de tâches est un ensemble de fichiers, y compris un modèle de tâche Open Job Description (OpenJD) et tous les fichiers d'actifs nécessaires au rendu de la tâche.

Le modèle de tâche décrit la manière dont les travailleurs traitent les actifs et y accèdent, et fournit le script qu'ils exécutent. Les offres d'emploi permettent aux artistes, aux directeurs techniques et aux développeurs de pipelines de soumettre facilement des tâches complexes à Deadline Cloud depuis leur poste de travail local ou leur ferme de rendu sur site. Les offres d'emploi sont particulièrement utiles pour les équipes travaillant sur des projets d'effets visuels, d'animation ou de rendu multimédia à grande échelle qui nécessitent des ressources informatiques évolutives à la demande.

Vous pouvez créer le lot de tâches à l'aide du système de fichiers local pour stocker les fichiers et d'un éditeur de texte pour créer le modèle de tâche. Après avoir créé le bundle, soumettez la tâche à Deadline Cloud à l'aide de la CLI de Deadline Cloud ou d'un outil tel qu'un émetteur Deadline Cloud

Vous pouvez stocker vos actifs dans un système de fichiers partagé entre vos employés, ou vous pouvez utiliser les pièces jointes aux tâches de Deadline Cloud pour automatiser le transfert des actifs vers des compartiments S3 où vos employés peuvent y accéder. Les pièces jointes aux tâches permettent également de transférer les résultats de vos tâches vers vos postes de travail.

Les sections suivantes fournissent des instructions détaillées sur la création et la soumission de lots de tâches à Deadline Cloud.

Rubriques

Modèles Open Job Description (OpenJD) pour Deadline Cloud

Un ensemble de tâches est l'un des outils que vous utilisez pour définir des tâches pour AWS Deadline Cloud. Ils regroupent un modèle Open Job Description (OpenJD) contenant des informations supplémentaires telles que les fichiers et les répertoires que vos tâches utilisent avec les pièces jointes aux tâches. Vous utilisez l'interface de ligne de commande (CLI) de Deadline Cloud pour utiliser un ensemble de tâches afin de soumettre des tâches à exécuter dans une file d'attente.

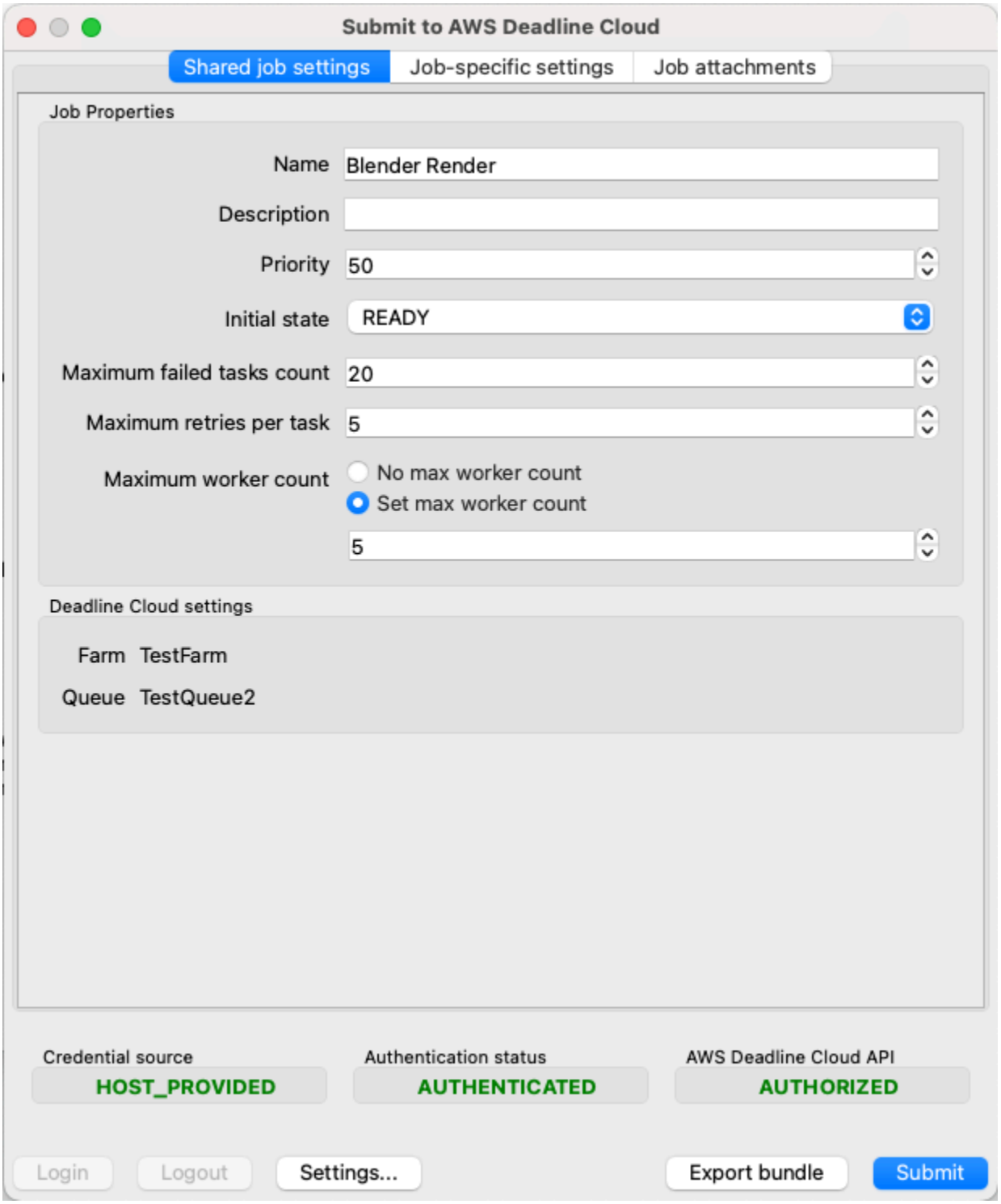
Un ensemble de tâches est une structure de répertoire qui contient un modèle de tâche OpenJD, d'autres fichiers qui définissent la tâche et les fichiers spécifiques à la tâche requis comme entrée pour votre tâche. Vous pouvez spécifier les fichiers qui définissent votre tâche sous forme de fichiers YAML ou JSON.

Le seul fichier requis est l'un `template.yaml` ou l'autre`template.json`. Vous pouvez également inclure les fichiers suivants :

```
/template.yaml (or template.json)
/asset_references.yaml (or asset_references.json)
/parameter_values.yaml (or parameter_values.json)
/other job-specific files and directories
```

Utilisez un ensemble de tâches pour les soumissions de tâches personnalisées à l'aide de la CLI de Deadline Cloud et d'une pièce jointe, ou vous pouvez utiliser une interface de soumission graphique. Par exemple, voici un exemple de Blender tiré de GitHub. Pour exécuter l'exemple à l'aide de la commande suivante dans le répertoire d'exemples de Blender :

```
deadline bundle gui-submit blender_render
```



Le panneau des paramètres spécifiques à la tâche est généré à partir des `userInterface` propriétés des paramètres de tâche définis dans le modèle de tâche.

Pour soumettre une tâche à l'aide de la ligne de commande, vous pouvez utiliser une commande similaire à la suivante

```
deadline bundle submit \
    --yes \
    --name Demo \
     -p BlenderSceneFile=location of scene file \
     -p OutputDir=file pathe for job output \
      blender_render/
```

Vous pouvez également utiliser la `deadline.client.api.create_job_from_job_bundle` fonction dans le package `deadline` Python.

Tous les plugins de soumission de tâches fournis avec Deadline Cloud, tels que le plug-in Autodesk Maya, génèrent un ensemble de tâches pour votre soumission, puis utilisent le package Python de Deadline Cloud pour soumettre votre travail à Deadline Cloud. Vous pouvez consulter les offres d'emploi soumises dans le répertoire de l'historique des tâches de votre poste de travail ou en utilisant un expéditeur. Vous pouvez trouver le répertoire de votre historique des tâches à l'aide de la commande suivante :

```
deadline config get settings.job_history_dir
```

Lorsque votre tâche est exécutée sur un worker Deadline Cloud, celui-ci a accès à des variables d'environnement qui lui fournissent des informations sur la tâche. Les variables d'environnement sont les suivantes :

Nom de la variable	Available
DEADLINE_FARM_ID	Toutes les actions
DATE_FLEET_ID	Toutes les actions
DEADLINE_WORKER_ID	Toutes les actions
DEADLINE_QUEUE_ID	Toutes les actions
DATE_JOB_ID	Toutes les actions
DEADLINE_STEP_ID	Actions relatives aux tâches
ID_DEADLINE_SESSION	Toutes les actions
ID_DEADLINE_TÂCHE	Actions relatives aux tâches

Nom de la variable	Available
DEADLINE_SESSION_ACTION_ID	Toutes les actions

Rubriques

Éléments de modèles de tâches pour les offres d'emploi

Le modèle de tâche définit l'environnement d'exécution et les processus exécutés dans le cadre d'une tâche Deadline Cloud. Vous pouvez créer des paramètres dans un modèle afin qu'il puisse être utilisé pour créer des tâches dont les valeurs d'entrée ne diffèrent que par les valeurs d'entrée, un peu comme une fonction dans un langage de programmation.

Lorsque vous soumettez une tâche à Deadline Cloud, elle s'exécute dans tous les environnements de file d'attente appliqués à la file d'attente. Les environnements de file d'attente sont créés à l'aide de la spécification des environnements externes Open Job Description (OpenJD). Pour plus de détails, consultez le modèle d'environnement dans le GitHub référentiel OpenJD.

Pour une introduction à la création d'une tâche à l'aide d'un modèle de tâche OpenJD, voir Présentation de la création d'une tâche dans le référentiel OpenJD GitHub . Vous trouverez des informations supplémentaires dans la section Comment les tâches sont exécutées. Vous trouverez des exemples de modèles de tâches dans le `samples` répertoire du GitHub référentiel OpenJD.

Vous pouvez définir le modèle de tâche au format YAML (`template.yaml`) ou au format JSON (`template.json`). Les exemples de cette section sont présentés au format YAML.

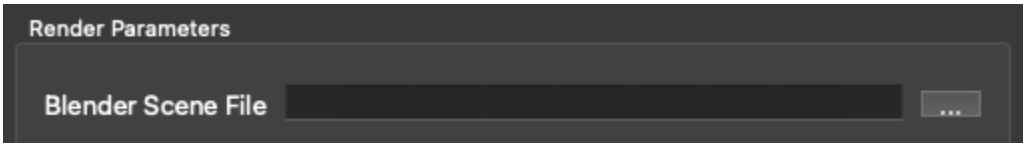
Par exemple, le modèle de tâche de l'`blender_render`exemple définit un paramètre d'entrée `BlenderSceneFile` sous la forme d'un chemin de fichier :

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
```

```
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
    - label: Blender Scene Files
      patterns: ["*.blend"]
    - label: All Files
      patterns: ["*"]
  description: >
    Choose the Blender scene file to render. Use the 'Job Attachments' tab
    to add textures and other files that the job needs.
```

La `userInterface` propriété définit le comportement des interfaces utilisateur générées automatiquement à la fois pour la ligne de commande à l'aide de la `deadline bundle gui-submit` commande et dans les plug-ins de soumission de tâches pour des applications telles qu'Autodesk Maya.

Dans cet exemple, le widget d'interface utilisateur permettant de saisir une valeur pour le `BlenderSceneFile` paramètre est une boîte de dialogue de sélection de fichiers qui affiche uniquement les fichiers. `.blend`



Pour d'autres exemples d'utilisation de l'`userIntefaceélément`, consultez l'exemple gui_control_showcase dans le référentiel sur. deadline-cloud-samples GitHub

Les `dataFlow` propriétés `objectType` et contrôlent le comportement des pièces jointes lorsque vous soumettez une tâche à partir d'un ensemble de tâches. Dans ce cas, `objectType: FILE dataFlow:IN` cela signifie que la valeur de `BlenderSceneFile` est un fichier d'entrée pour les pièces jointes aux tâches.

En revanche, la définition du `OutputDir` paramètre a `objectType: DIRECTORY` et `dataFlow: OUT` :

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
```

```
    label: Output Directory
    groupLabel: Render Parameters
  default: "./output"
  description: Choose the render output directory.
```

La valeur du `OutputDir` paramètre est utilisée par les pièces jointes aux tâches comme répertoire dans lequel la tâche écrit les fichiers de sortie.

Pour plus d'informations sur les `dataFlow` propriétés `objectType` et, voir JobPathParameterDefinitionla spécification Open Job Description

Le reste de l'exemple de modèle de `blender_render` tâche définit le flux de travail de la tâche comme une seule étape, chaque image de l'animation étant rendue comme une tâche distincte :

```
steps:
- name: RenderBlender
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"
  script:
    actions:
      onRun:
        command: bash
        # Note: {{Task.File.Run}} is a variable that expands to the filename on the
 worker host's
        # disk where the contents of the 'Run' embedded file, below, is written.
        args: ['{{Task.File.Run}}']
    embeddedFiles:
      - name: Run
        type: TEXT
        data: |
          # Configure the task to fail if any individual command fails.
          set -xeuo pipefail

          mkdir -p '{{Param.OutputDir}}'

          blender --background '{{Param.BlenderSceneFile}}' \
                  --render-output '{{Param.OutputDir}}/{{Param.OutputPattern}}' \
                  --render-format {{Param.Format}} \
                  --use-extension 1 \
                  --render-frame {{Task.Param.Frame}}
```

Par exemple, si la valeur du `Frames` paramètre est`1-10`, il définit 10 tâches. Chaque tâche possède une valeur différente pour le `Frame` paramètre. Pour exécuter une tâche, procédez comme suit :

1. Toutes les références de variables de la `data` propriété du fichier intégré sont étendues, par exemple`--render-frame 1`.
2. Le contenu de la `data` propriété est écrit dans un fichier du répertoire de travail de la session sur le disque.
3. La onRun commande de la tâche se résout en `bash` *`location of embedded file`* puis s'exécute.

Pour plus d'informations sur les fichiers intégrés, les sessions et les emplacements mappés par des chemins, consultez la section Comment les tâches sont exécutées dans la spécification Open Job Description.

Vous trouverez d'autres exemples de modèles de tâches dans le référentiel deadline-cloud-samples/job_bundles, ainsi que les exemples de modèles fournis avec la spécification Open Job Descriptions.

Découpage des tâches pour les modèles de tâches

Le découpage des tâches vous permet de regrouper plusieurs tâches en une seule unité de travail appelée partie. Dans une tâche de rendu, par exemple, cela signifie que Deadline Cloud peut distribuer plusieurs images ensemble au lieu d'une image par appel de commande. Cela réduit la charge de travail liée au démarrage des applications pour chaque tâche et réduit le temps d'exécution total des tâches. Pour plus de détails, voir Exécuter plusieurs images à la fois dans le wiki OpenJD.

OpenJD prend en charge les extensions qui ajoutent des fonctionnalités facultatives aux modèles de tâches. Le découpage des tâches est activé en ajoutant l'`TASK_CHUNKING`extension. Pour utiliser le découpage, ajoutez l'extension à votre modèle de tâche et utilisez le type de paramètre de `CHUNK[INT]` tâche. Soumettez des tâches segmentées à l'aide de la même `deadline bundle submit` commande. Par exemple, le modèle de tâche suivant affiche les cadres par blocs de 10 :

```
specificationVersion: 'jobtemplate-2023-09'
extensions:
  - TASK_CHUNKING
name: Blender Render with Contiguous Chunking
parameterDefinitions:
  - name: BlenderSceneFile
    type: PATH
    objectType: FILE
```

```
    dataFlow: IN
  - name: Frames
    type: STRING
    default: "1-100"
  - name: OutputDir
    type: PATH
    objectType: DIRECTORY
    dataFlow: OUT
    default: "./output"
steps:
  - name: RenderBlender
    parameterSpace:
      taskParameterDefinitions:
        - name: Frame
          type: CHUNK[INT]
          range: "{{Param.Frames}}"
          chunks:
            defaultTaskCount: 10
            rangeConstraint: CONTIGUOUS
    script:
      actions:
        onRun:
          command: bash
          args: ["{{Task.File.Run}}"]
      embeddedFiles:
        - name: Run
          type: TEXT
          data: |
            set -xeuo pipefail

            mkdir -p '{{Param.OutputDir}}'

            # Parse the chunk range (e.g., "1-10") into start and end frames
            START_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f1)"
            END_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f2)"

            blender --background '{{Param.BlenderSceneFile}}' \
                    --render-output '{{Param.OutputDir}}/output_####' \
                    --render-format PNG \
                    --use-extension 1 \
                    -s "$START_FRAME" \
                    -e "$END_FRAME" \
                    --render-anim
```

Dans cet exemple, Deadline Cloud divise les 100 images en morceaux tels que `1-10`11-20,, etc. La `{{Task.Param.Frame}}` variable s'étend jusqu'à une expression de plage telle que`1-10`. Comme elle `rangeConstraint` est définie sur`CONTIGUOUS`, la plage est toujours au `start-end` format. Le script analyse cette plage et transmet les images de début et de fin à Blender en utilisant les `-e` options `-s` et avec`--render-anim`.

La `chunks` propriété prend en charge les champs suivants :

- `defaultTaskCount`— (Obligatoire) Combien de tâches combiner en un seul morceau. La valeur maximale est de 150.
- `rangeConstraint`— (Obligatoire) Si`CONTIGUOUS`, un fragment est toujours une plage contiguë, par exemple. `1-10` Si`NONCONTIGUOUS`, un morceau peut être un ensemble arbitraire, comme`1,3,7-10`.
- `targetRuntimeSeconds`— (Facultatif) Le temps d'exécution cible en secondes pour chaque segment. Deadline Cloud peut ajuster dynamiquement la taille des fragments pour s'approcher de cette cible une fois que certains fragments sont terminés.

Pour d'autres exemples de découpage de tâches, y compris des exemples de base et des exemples de Blender avec des segments contigus et non contigus, consultez les exemples de découpage de tâches dans le référentiel d'exemples de Deadline Cloud sur. GitHub

> **Exigences relatives à la gestion du parc par le client**
>
> Le découpage des tâches nécessite une version d'agent de travail compatible. Si vous utilisez des flottes gérées par le client, assurez-vous que vos agents de travail sont à jour avant de soumettre des tâches par segmentation. Les flottes gérées par des services utilisent toujours une version d'agent de travail compatible.

> **Téléchargement du résultat pour les tâches segmentées**
>
> Lorsque vous téléchargez le résultat d'une tâche unique dans une tâche fragmentée, Deadline Cloud télécharge le résultat pour l'ensemble du fragment. Par exemple, si les images 1 à 10 ont été traitées ensemble, le téléchargement de la sortie pour l'image 3 inclut toutes les images 1 à 10. Cette fonctionnalité nécessite `deadline-cloud` la version 0.53.3 ou ultérieure.

Éléments de valeurs de paramètres pour les ensembles de tâches

Vous pouvez utiliser le fichier de paramètres pour définir les valeurs de certains paramètres de tâche dans le modèle de tâche ou les arguments de demande d'CreateJobopération dans le bundle de tâches afin de ne pas avoir à définir de valeurs lors de la soumission d'une tâche. L'interface utilisateur de soumission des tâches vous permet de modifier ces valeurs.

Vous pouvez définir le modèle de tâche au format YAML (`parameter_values.yaml`) ou au format JSON (`parameter_values.json`). Les exemples de cette section sont présentés au format YAML.

En YAML, le format du fichier est le suivant :

```
parameterValues:
- name: <string>
  value: <integer>, <float>, or <string>
- name: <string>
  value: <integer>, <float>, or <string>ab
... repeating as necessary
```

Chaque élément de la `parameterValues` liste doit être l'un des suivants :

- Paramètre de tâche défini dans le modèle de tâche.
- Paramètre de tâche défini dans un environnement de file d'attente pour la file d'attente à laquelle vous soumettez la tâche.
- Paramètre spécial transmis à l'`CreateJob`opération lors de la création d'une tâche.
  - `deadline:priority`— La valeur doit être un entier. Il est transmis à l'`CreateJob`opération en tant que paramètre de priorité.
  - `deadline:targetTaskRunStatus`— La valeur doit être une chaîne. Il est transmis à l'`CreateJob`opération en tant que paramètre targetTaskRunStatus.
  - `deadline:maxFailedTasksCount`— La valeur doit être un entier. Il est transmis à l'`CreateJob`opération en tant que paramètre maxFailedTasksCount.
  - `deadline:maxRetriesPerTask`— La valeur doit être un entier. Il est transmis à l'`CreateJob`opération en tant que paramètre de maxRetriesPertâche.
  - `deadline:maxWorkercount`— La valeur doit être un entier. Il est transmis à l'`CreateJob`opération en tant que maxWorkerCountparamètre.

Un modèle de tâche est toujours un modèle plutôt qu'une tâche spécifique à exécuter. Un fichier de valeurs de paramètres permet à un ensemble de tâches de servir de modèle si certains paramètres n'ont pas de valeurs définies dans ce fichier, ou de soumission de tâches spécifique si tous les paramètres ont des valeurs.

Par exemple, l'exemple blender_render ne possède pas de fichier de paramètres et son modèle de tâche définit des paramètres sans valeurs par défaut. Ce modèle doit être utilisé comme modèle pour créer des tâches. Une fois que vous avez créé une tâche à l'aide de cette offre de tâches, Deadline Cloud écrit une nouvelle série de tâches dans le répertoire de l'historique des tâches.

Par exemple, lorsque vous soumettez une tâche à l'aide de la commande suivante :

```
deadline bundle gui-submit blender_render/
```

Le nouveau lot de tâches contient un `parameter_values.yaml` fichier contenant les paramètres spécifiés :

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
parameter_values.yaml
parameterValues:
- name: deadline:targetTaskRunStatus
  value: READY
- name: deadline:maxFailedTasksCount
  value: 10
- name: deadline:maxRetriesPerTask
  value: 5
- name: deadline:priority
  value: 75
- name: BlenderSceneFile
  value: /private/tmp/bundle_demo/bmw27_cpu.blend
- name: Frames
  value: 1-10
- name: OutputDir
  value: /private/tmp/bundle_demo/output
- name: OutputPattern
  value: output_####
- name: Format
  value: PNG
- name: CondaPackages
  value: blender
- name: RezPackages
```

```
  value: blender
```

Vous pouvez créer la même tâche à l'aide de la commande suivante :

```
deadline bundle submit ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-
JobBundle-Demo/
```

> **Note**
>
> Le lot de tâches que vous soumettez est enregistré dans votre répertoire d'historique des tâches. Vous pouvez trouver l'emplacement de ce répertoire à l'aide de la commande suivante :
>
> ```
> deadline config get settings.job_history_dir
> ```

Éléments de référence aux actifs pour les offres d'emploi

Vous pouvez utiliser les pièces jointes aux tâches de Deadline Cloud pour transférer des fichiers entre votre poste de travail et Deadline Cloud. Le fichier de référence des actifs répertorie les fichiers et répertoires d'entrée, ainsi que les répertoires de sortie pour vos pièces jointes. Si vous ne listez pas tous les fichiers et répertoires de ce fichier, vous pouvez les sélectionner lorsque vous soumettez une tâche à l'aide de la `deadline bundle gui-submit` commande.

Ce fichier n'a aucun effet si vous n'utilisez pas de pièces jointes aux tâches.

Vous pouvez définir le modèle de tâche au format YAML (`asset_references.yaml`) ou au format JSON (`asset_references.json`). Les exemples de cette section sont présentés au format YAML.

En YAML, le format du fichier est le suivant :

```
assetReferences:
    inputs:
        # Filenames on the submitting workstation whose file contents are needed as
        # inputs to run the job.
        filenames:
        - list of file paths
        # Directories on the submitting workstation whose contents are needed as inputs
```

```
        # to run the job.
        directories:
        - list of directory paths

    outputs:
        # Directories on the submitting workstation where the job writes output files
        # if running locally.
        directories:
        - list of directory paths

    # Paths referenced by the job, but not necessarily input or output.
    # Use this if your job uses the name of a path in some way, but does not explicitly
need
    # the contents of that path.
    referencedPaths:
    - list of directory paths
```

Lorsque vous sélectionnez le fichier d'entrée ou de sortie à télécharger sur Amazon S3, Deadline Cloud compare le chemin du fichier aux chemins répertoriés dans vos profils de stockage. Chaque emplacement de système SHARED de fichiers de type -type dans un profil de stockage fait abstraction d'un partage de fichiers réseau monté sur vos postes de travail et vos hôtes de travail. Deadline Cloud télécharge uniquement les fichiers qui ne figurent pas sur l'un de ces partages de fichiers.

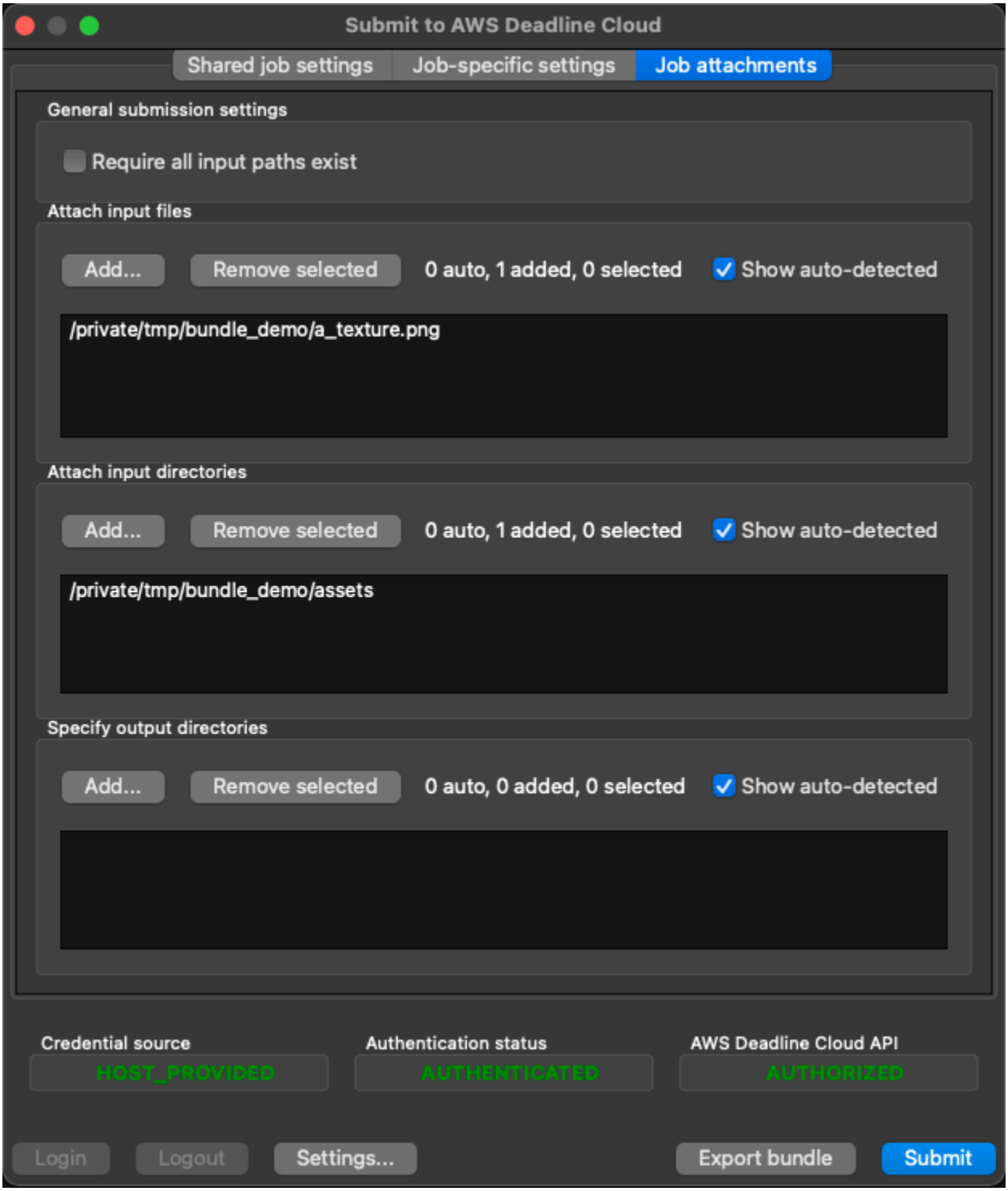
Pour plus d'informations sur la création et l'utilisation de profils de stockage, consultez la section Stockage partagé dans Deadline Cloud dans le guide de l'utilisateur de AWS Deadline Cloud.

Example- Le fichier de référence des actifs créé par l'interface graphique de Deadline Cloud

Utilisez la commande suivante pour soumettre une tâche à l'aide de l'exemple blender_render.

```
deadline bundle gui-submit blender_render/
```

Ajoutez des fichiers supplémentaires à la tâche dans l'onglet Pièces jointes à la tâche :



Après avoir soumis la tâche, vous pouvez consulter le `asset_references.yaml` fichier du bundle de tâches dans le répertoire de l'historique des tâches pour voir les actifs du fichier YAML :

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
asset_references.yaml
```

```
assetReferences:
  inputs:
    filenames:
    - /private/tmp/bundle_demo/a_texture.png
    directories:
    - /private/tmp/bundle_demo/assets
  outputs:
    directories: []
  referencedPaths: []
```

Utilisation de fichiers dans le cadre de vos tâches

La plupart des tâches que vous soumettez à AWS Deadline Cloud comportent des fichiers d'entrée et de sortie. Vos fichiers d'entrée et vos répertoires de sortie peuvent se trouver sur une combinaison de systèmes de fichiers partagés et de lecteurs locaux. Les offres d'emploi doivent localiser le contenu à ces emplacements. Deadline Cloud propose deux fonctionnalités, les pièces jointes aux tâches et les profils de stockage qui fonctionnent ensemble pour aider vos tâches à localiser les fichiers dont elles ont besoin.

Les offres d'emploi offrent plusieurs avantages

- Déplacer des fichiers entre hôtes à l'aide d'Amazon S3
- Transférez des fichiers de votre poste de travail vers des hôtes professionnels et vice versa
- Disponible pour les tâches dans les files d'attente pour lesquelles vous activez la fonctionnalité
- Principalement utilisé avec les flottes gérées par les services, mais également compatible avec les flottes gérées par le client.

Utilisez des profils de stockage pour cartographier la disposition des emplacements des systèmes de fichiers partagés sur votre poste de travail et sur les hôtes de travail. Ce mappage permet à vos tâches de localiser les fichiers et répertoires partagés lorsque leur emplacement diffère entre votre poste de travail et les hôtes de travail, comme dans le cas de configurations multiplateformes avec des postes de travail Windows basés et des hôtes de travail Linux basés. La carte de configuration de votre système de fichiers du profil de stockage est également utilisée par les pièces jointes aux tâches pour identifier les fichiers dont elles ont besoin pour être transférées entre les hôtes via Amazon S3.

Si vous n'utilisez pas de pièces jointes aux tâches et que vous n'avez pas besoin de remapper les emplacements des fichiers et des répertoires entre les postes de travail et les hôtes de travail, vous n'avez pas besoin de modéliser vos partages de fichiers à l'aide de profils de stockage.

Rubriques

Exemple d'infrastructure de projet

Pour démontrer l'utilisation des pièces jointes aux tâches et des profils de stockage, configurez un environnement de test avec deux projets distincts. Vous pouvez utiliser la console Deadline Cloud pour créer les ressources de test.

1. Si ce n'est pas déjà fait, créez un parc de tests. Pour créer une ferme, suivez la procédure décrite dans Créer une ferme.
2. Créez deux files d'attente pour les tâches dans chacun des deux projets. Pour créer des files d'attente, suivez la procédure décrite dans Créer une file d'attente.

    a. Créez la première file d'attente appelée**`Q1`**. Utilisez la configuration suivante, utilisez les valeurs par défaut pour tous les autres éléments.

    - Pour les pièces jointes aux tâches, choisissez Create a new Amazon S3 bucket.
    - Sélectionnez Activer l'association avec les flottes gérées par le client.
    - Pour exécuter en tant qu'utilisateur, entrez à **`jobuser`** la fois l'utilisateur et le groupe POSIX.
    - Pour le rôle de service de file d'attente, créez un nouveau rôle nommé **`AssetDemoFarm-Q1-Role`**
    - Décochez la case par défaut de l'environnement de file d'attente Conda.

    b. Créez la deuxième file d'attente appelée**`Q2`**. Utilisez la configuration suivante, utilisez les valeurs par défaut pour tous les autres éléments.

    - Pour les pièces jointes aux tâches, choisissez Create a new Amazon S3 bucket.
    - Sélectionnez Activer l'association avec les flottes gérées par le client.
    - Pour exécuter en tant qu'utilisateur, entrez à **`jobuser`** la fois l'utilisateur et le groupe POSIX.

    - Pour le rôle de service de file d'attente, créez un nouveau rôle nommé **`AssetDemoFarm-Q2-Role`**
    - Décochez la case par défaut de l'environnement de file d'attente Conda.

3. Créez une flotte unique gérée par le client qui exécute les tâches à partir des deux files d'attente. Pour créer le parc, suivez la procédure décrite dans Créer un parc géré par le client. Utilisez la configuration suivante :

    - Pour Nom, utilisez**`DemoFleet`**.
    - Pour le type de flotte, sélectionnez Géré par le client
    - Pour le rôle de service de flotte, créez un nouveau rôle nommé AssetDemoFarm-Fleet-Role.
    - N'associez la flotte à aucune file d'attente.

L'environnement de test suppose que trois systèmes de fichiers sont partagés entre les hôtes à l'aide de partages de fichiers réseau. Dans cet exemple, les emplacements portent les noms suivants :

- `FSCommon`- contient les actifs de travail d'entrée communs aux deux projets.
- `FS1`- contient les actifs de travail d'entrée et de sortie pour le projet 1.
- `FS2`- contient les actifs de travail d'entrée et de sortie pour le projet 2.

L'environnement de test suppose également qu'il existe trois postes de travail, comme suit :

- `WSAll`- Un poste Linux de travail basé utilisé par les développeurs pour tous les projets. Les emplacements des systèmes de fichiers partagés sont les suivants :
    - `FSCommon`: `/shared/common`
    - `FS1`: `/shared/projects/project1`
    - `FS2`: `/shared/projects/project2`
- `WS1`- Un poste Windows de travail basé utilisé pour le projet 1. Les emplacements des systèmes de fichiers partagés sont les suivants :
    - `FSCommon`: `S:\`
    - `FS1`: `Z:\`
    - `FS2`: Non disponible
- `WS1`- Un poste de travail macOS basé utilisé pour le projet 2. Les emplacements du système de fichiers partagé sont les suivants :

- FSCommon: /Volumes/common
- FS1: Non disponible
- FS2: /Volumes/projects/project2

Enfin, définissez les emplacements des systèmes de fichiers partagés pour les employés de votre flotte. Les exemples suivants font référence à cette configuration sous le nom de`WorkerConfig`. Les emplacements partagés sont les suivants :

- FSCommon: /mnt/common
- FS1: /mnt/projects/project1
- FS2: /mnt/projects/project2

Vous n'avez pas besoin de configurer de systèmes de fichiers partagés, de postes de travail ou de travailleurs correspondant à cette configuration. Les emplacements partagés n'ont pas besoin d'exister pour la démonstration.

Profils de stockage et mappage des chemins

Utilisez des profils de stockage pour modéliser les systèmes de fichiers de votre poste de travail et de vos hôtes de travail. Chaque profil de stockage décrit le système d'exploitation et la structure du système de fichiers de l'une de vos configurations système. Cette rubrique décrit comment utiliser les profils de stockage pour modéliser les configurations du système de fichiers de vos hôtes afin que Deadline Cloud puisse générer des règles de mappage de chemins pour vos tâches, et comment ces règles de mappage de chemins sont générées à partir de vos profils de stockage.

Lorsque vous soumettez une tâche à Deadline Cloud, vous pouvez fournir un identifiant de profil de stockage facultatif pour la tâche. Ce profil de stockage décrit le système de fichiers du poste de travail émetteur. Il décrit la configuration du système de fichiers d'origine utilisée par les chemins de fichiers du modèle de tâche.

Vous pouvez également associer un profil de stockage à une flotte. Le profil de stockage décrit la configuration du système de fichiers de tous les hôtes de travail du parc. Si vous avez des employés dont la configuration du système de fichiers est différente, ces travailleurs doivent être affectés à un parc différent de votre ferme.

Les règles de mappage de chemins décrivent comment les chemins doivent être remappés entre la façon dont ils sont spécifiés dans la tâche et l'emplacement réel du chemin sur un hôte de travail.

Deadline Cloud compare la configuration du système de fichiers décrite dans le profil de stockage d'une tâche avec le profil de stockage du parc qui exécute la tâche afin de dériver ces règles de mappage de chemins.

Rubriques

Modélisez les emplacements des systèmes de fichiers partagés à l'aide de profils de stockage

Un profil de stockage modélise la configuration du système de fichiers de l'une de vos configurations d'hôte. Il existe quatre configurations hôtes différentes dans l'exemple d'infrastructure de projet. Dans cet exemple, vous créez un profil de stockage distinct pour chacun d'entre eux. Vous pouvez créer un profil de stockage à l'aide de l'une des méthodes suivantes :

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation ressource
- Console AWS

Un profil de stockage est constitué d'une liste d'emplacements de systèmes de fichiers qui indiquent chacun à Deadline Cloud l'emplacement et le type d'emplacement du système de fichiers pertinent pour les tâches soumises ou exécutées sur un hôte. Un profil de stockage ne doit modéliser que les emplacements pertinents pour les tâches. Par exemple, l'`FSCommon`emplacement partagé est situé sur le poste de travail `WS1` à`S:\`, de sorte que l'emplacement du système de fichiers correspondant est le suivant :

```
{
    "name": "FSCommon",
    "path": "S:\\",
    "type": "SHARED"
}
```

Utilisez les commandes suivantes pour créer le profil de stockage pour les configurations de station de travail `WS1`WS2, `WS3` et la configuration de l'utilisateur à `WorkerConfig` l'aide de la commande AWS CLIin AWS CloudShell:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WSAll \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/shared/common"},
      {"name": "FS1", "type":"SHARED", "path":"/shared/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/shared/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS1 \
  --os-family WINDOWS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
      {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
   ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS2 \
  --os-family MACOS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/Volumes/common"},
      {"name": "FS2", "type":"SHARED", "path":"/Volumes/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WorkerCfg \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/mnt/common"},
      {"name": "FS1", "type":"SHARED", "path":"/mnt/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/mnt/projects/project2"}
```

```
  ]'
```

> **Note**
>
> Vous devez faire référence aux emplacements des systèmes de fichiers dans vos profils de stockage en utilisant les mêmes valeurs pour la `name` propriété dans tous les profils de stockage de votre parc de serveurs. Deadline Cloud compare les noms pour déterminer si les emplacements des systèmes de fichiers issus de différents profils de stockage font référence au même emplacement lors de la génération des règles de mappage de chemins.

Configuration des profils de stockage pour les flottes

Vous pouvez configurer un parc de manière à inclure un profil de stockage qui modélise les emplacements des systèmes de fichiers de tous les employés du parc. La configuration du système de fichiers hôte de tous les travailleurs d'une flotte doit correspondre au profil de stockage de cette flotte. Les travailleurs ayant des configurations de système de fichiers différentes doivent appartenir à des flottes distinctes.

Pour définir la configuration de votre flotte afin d'utiliser le profil `WorkerConfig` de stockage, utilisez le fichier AWS CLIdans AWS CloudShell:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerConfig
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

FLEET_WORKER_MODE=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.mode' \
)
FLEET_WORKER_CAPABILITIES=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.workerCapabilities' \
)

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
```

```
  "{
    \"customerManaged\": {
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

Configuration des profils de stockage pour les files d'attente

La configuration d'une file d'attente inclut une liste de noms distinguant majuscules et minuscules des emplacements du système de fichiers partagé auxquels les tâches soumises à la file d'attente doivent avoir accès. Par exemple, les tâches soumises à la file d'attente `Q1` nécessitent des emplacements `FSCommon` de système de fichiers et. `FS1` Les tâches soumises à la file d'attente `Q2` nécessitent l'emplacement du système de fichiers `FSCommon` et`FS2`.

Pour définir les configurations de la file d'attente afin d'exiger ces emplacements de système de fichiers, utilisez le script suivant :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of QUEUE2_ID to queue Q2's identifier
QUEUE2_ID=queue-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --required-file-system-location-names-to-add FSComm FS1

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --required-file-system-location-names-to-add FSComm FS2
```

La configuration d'une file d'attente inclut également une liste de profils de stockage autorisés qui s'applique aux tâches soumises à cette file d'attente et aux flottes associées à cette file d'attente. Seuls les profils de stockage qui définissent les emplacements du système de fichiers pour tous les emplacements de système de fichiers requis pour la file d'attente sont autorisés dans la liste des profils de stockage autorisés de la file d'attente.

Une tâche échoue si vous la soumettez avec un profil de stockage qui ne figure pas dans la liste des profils de stockage autorisés pour la file d'attente. Vous pouvez toujours soumettre une tâche sans

profil de stockage à une file d'attente. Les configurations de poste de travail sont étiquetées `WSAll` et `WS1` toutes deux possèdent les emplacements de système de fichiers requis (`FSCommonetFS1`) pour la file d'attente`Q1`. Ils doivent être autorisés à soumettre des tâches à la file d'attente. De même, les configurations des postes `WSAll` de travail `WS2` répondent aux exigences en matière de file d'attente`Q2`. Ils doivent être autorisés à soumettre des tâches à cette file d'attente. Mettez à jour les deux configurations de file d'attente pour autoriser les tâches à être soumises avec ces profils de stockage à l'aide du script suivant :

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS1 to the identifier of the WS1 storage profile
WS1_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS2 to the identifier of the WS2 storage profile
WS2_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS1_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS2_ID
```

Si vous ajoutez le profil `WS2` de stockage à la liste des profils de stockage autorisés pour la file d'attente, `Q1` il échoue :

```
$ aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WS2_ID

An error occurred (ValidationException) when calling the UpdateQueue operation: Storage
 profile id: sp-00112233445566778899aabbccddeeff does not have required file system
 location: FS1
```

Cela est dû au fait que le profil de `WS2` stockage ne contient pas de définition de l'emplacement du système de fichiers nommé `FS1` `Q1` requis par la file d'attente.

L'association d'un parc configuré à un profil de stockage ne figurant pas dans la liste des profils de stockage autorisés de la file d'attente échoue également. Par exemple :

```
$ aws deadline create-queue-fleet-association --farm-id $FARM_ID \
   --fleet-id $FLEET_ID \
   --queue-id $QUEUE1_ID
```

```
An error occurred (ValidationException) when calling the CreateQueueFleetAssociation
 operation: Mismatch between storage profile ids.
```

Pour corriger l'erreur, ajoutez le profil de stockage nommé `WorkerConfig` à la liste des profils de stockage autorisés pour la file d'attente `Q1` et la file d'attente`Q2`. Associez ensuite le parc à ces files d'attente afin que les employés du parc puissent exécuter des tâches à partir des deux files d'attente.

```
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerCfg
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

Dériver les règles de mappage des chemins à partir des profils de stockage

Les règles de mappage de chemins décrivent comment les chemins doivent être remappés entre la tâche et l'emplacement réel du chemin sur un hôte de travail. Lorsqu'une tâche est exécutée sur un travailleur, le profil de stockage de la tâche est comparé au profil de stockage du parc du travailleur afin de déterminer les règles de mappage des chemins pour la tâche.

Deadline Cloud crée une règle de mappage pour chacun des emplacements de système de fichiers requis dans la configuration de la file d'attente. Par exemple, une tâche soumise avec le profil de `WSAll` stockage à mettre en file d'attente `Q1` est soumise aux règles de mappage des chemins :

- `FSComm`: `/shared/common -> /mnt/common`
- `FS1`: `/shared/projects/project1 -> /mnt/projects/project1`

Deadline Cloud crée des règles pour les emplacements du système de FS1 fichiers FSComm et, mais pas pour l'emplacement du système de FS2 fichiers, même si les profils WSAll et WorkerConfig de stockage le définissent tous deuxFS2. Cela est dû au fait que Q1 la liste des emplacements de système de fichiers requis de la file d'attente est["FSComm", "FS1"].

Vous pouvez confirmer les règles de mappage de chemin disponibles pour les tâches soumises avec un profil de stockage particulier en soumettant une tâche qui imprime le fichier de règles de mappage de chemin d'Open Job Description, puis en lisant le journal de session une fois la tâche terminée :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSALL storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
            "onRun": {
              "command": "/bin/cat",
              "args": [ "{{Session.PathMappingRulesFile}}" ]
            }
          }
        }
      }
    ]
  }'
```

Si vous utilisez la CLI Deadline Cloud pour soumettre des tâches, ses paramètres de `settings.storage_profile_id` configuration définissent le profil de stockage que les tâches soumises avec la CLI auront. Pour soumettre des tâches avec le profil WSAll de stockage, définissez :

```
deadline config set settings.storage_profile_id $WSALL_ID
```

Pour exécuter un travailleur géré par le client comme s'il s'exécutait dans l'infrastructure d'exemple, suivez la procédure décrite dans la section Exécuter l'agent de travail dans le guide de l'utilisateur de Deadline Cloud pour exécuter un travailleur avec. AWS CloudShell Si vous avez déjà suivi ces instructions, supprimez d'abord `~/demoenv-persist` les répertoires `~/demoenv-logs` et. Définissez également les valeurs des variables d'`DEV_CMF_ID`Environnement `DEV_FARM_ID` et auxquelles les directions font référence comme suit avant de procéder :

```
DEV_FARM_ID=$FARM_ID
DEV_CMF_ID=$FLEET_ID
```

Une fois la tâche exécutée, vous pouvez consulter les règles de mappage des chemins dans le fichier journal de la tâche :

```
cat demoenv-logs/${QUEUE1_ID}/*.log
...
JJSON log results (see below)
...
```

Le journal contient le mappage des systèmes de `FSComm` fichiers `FS1` et. Reformatée pour plus de lisibilité, l'entrée du journal ressemble à ceci :

```
{
    "version": "pathmapping-1.0",
    "path_mapping_rules": [
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/projects/project1",
            "destination_path": "/mnt/projects/project1"
        },
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/common",
            "destination_path": "/mnt/common"
        }
    ]
```

Vous pouvez soumettre des tâches avec différents profils de stockage pour voir comment les règles de mappage des chemins changent.

Utiliser les pièces jointes aux tâches pour partager des fichiers

Utilisez les pièces jointes aux tâches pour rendre les fichiers ne figurant pas dans les répertoires partagés disponibles pour vos tâches et pour capturer les fichiers de sortie s'ils ne sont pas écrits dans des répertoires partagés. Job Attachments utilise Amazon S3 pour transférer les fichiers entre les hôtes. Les fichiers sont stockés dans des compartiments S3 et il n'est pas nécessaire de télécharger un fichier si son contenu n'a pas changé.

Vous devez utiliser des pièces jointes lorsque vous exécutez des tâches sur des flottes gérées par des services, car les hôtes ne partagent pas l'emplacement des systèmes de fichiers. Les pièces jointes aux tâches sont également utiles dans les flottes gérées par les clients lorsque les fichiers d'entrée ou de sortie d'une tâche sont stockés sur un système de fichiers réseau partagé, par exemple lorsque votre ensemble de tâches contient des scripts shell ou Python.

Lorsque vous soumettez un ensemble de tâches à l'aide de la CLI de Deadline Cloud ou d'un émetteur de Deadline Cloud, les pièces jointes aux tâches utilisent le profil de stockage de la tâche et les emplacements du système de fichiers requis dans la file d'attente pour identifier les fichiers d'entrée qui ne se trouvent pas sur un hôte de travail et qui doivent être téléchargés sur Amazon S3 dans le cadre de la soumission de la tâche. Ces profils de stockage aident également Deadline Cloud à identifier les fichiers de sortie sur les sites d'accueil des travailleurs qui doivent être téléchargés sur Amazon S3 afin d'être disponibles sur votre poste de travail.

Les exemples de pièces jointes aux tâches utilisent les configurations de ferme, de flotte, de files d'attente et de profils de stockage issues de Exemple d'infrastructure de projet etProfils de stockage et mappage des chemins. Vous devriez parcourir ces sections avant celle-ci.

Dans les exemples suivants, vous utilisez un exemple de série de tâches comme point de départ, puis vous le modifiez pour explorer les fonctionnalités de la pièce jointe aux tâches. Les offres d'emploi constituent le meilleur moyen pour vos offres d'emploi d'utiliser les pièces jointes.
Ils combinent un modèle de tâche Open Job Description dans un répertoire avec des fichiers supplémentaires qui répertorient les fichiers et les répertoires requis par les tâches utilisant le bundle de tâches. Pour plus d'informations sur les offres d'emploi, consultezModèles Open Job Description (OpenJD) pour Deadline Cloud.

Soumission de fichiers avec une tâche

Avec Deadline Cloud, vous pouvez permettre aux flux de travail d'accéder aux fichiers d'entrée qui ne sont pas disponibles dans les emplacements des systèmes de fichiers partagés sur les hôtes

de travail. Les pièces jointes aux tâches permettent aux tâches de rendu d'accéder à des fichiers résidant uniquement sur le disque d'un poste de travail local ou dans un environnement de parc géré par des services. Lorsque vous soumettez un ensemble de tâches, vous pouvez inclure des listes de fichiers d'entrée et de répertoires requis par la tâche. Deadline Cloud identifie ces fichiers non partagés, les télécharge depuis la machine locale vers Amazon S3 et les télécharge sur l'hôte du poste de travail. Il rationalise le processus de transfert des actifs d'entrée vers les nœuds de rendu, garantissant ainsi que tous les fichiers requis sont accessibles pour l'exécution distribuée des tâches.

Vous pouvez spécifier les fichiers des tâches directement dans le bundle de tâches, utiliser les paramètres du modèle de tâche que vous fournissez à l'aide de variables d'environnement ou d'un script, et utiliser le `assets_references` fichier de la tâche. Vous pouvez utiliser l'une de ces méthodes ou une combinaison des trois. Vous pouvez spécifier un profil de stockage pour le bundle de la tâche afin qu'il ne télécharge que les fichiers modifiés sur le poste de travail local.

Cette section utilise un exemple de bundle de tâches GitHub pour montrer comment Deadline Cloud identifie les fichiers de votre tâche à télécharger, comment ces fichiers sont organisés dans Amazon S3 et comment ils sont mis à la disposition des hôtes de travail traitant vos tâches.

Rubriques

- Comment Deadline Cloud télécharge des fichiers sur Amazon S3
- Comment Deadline Cloud choisit les fichiers à télécharger
- Comment les tâches trouvent-elles les fichiers d'entrée des pièces jointes

Comment Deadline Cloud télécharge des fichiers sur Amazon S3

Cet exemple montre comment Deadline Cloud télécharge des fichiers depuis votre poste de travail ou votre hôte de travail vers Amazon S3 afin qu'ils puissent être partagés. Il utilise un exemple de bundle de tâches GitHub et la CLI de Deadline Cloud pour soumettre des tâches.

Commencez par cloner le GitHubréférentiel d'échantillons de Deadline Cloud dans votre AWS CloudShellenvironnement, puis copiez le bundle de `job_attachments_devguide` tâches dans votre répertoire personnel :

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide ~/
```

Installez la CLI de Deadline Cloud pour soumettre des ensembles de tâches :

```
pip install deadline --upgrade
```

Le bundle de `job_attachments_devguide` tâches comporte une seule étape avec une tâche qui exécute un script shell bash dont l'emplacement du système de fichiers est transmis en tant que paramètre de tâche. La définition du paramètre de tâche est la suivante :

```
...
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
...
```

La `IN` valeur de la `dataFlow` propriété indique aux pièces jointes à la tâche que la valeur du `ScriptFile` paramètre est une entrée de la tâche. La valeur de la `default` propriété est un emplacement relatif par rapport au répertoire du bundle de tâches, mais il peut également s'agir d'un chemin absolu. Cette définition de paramètre déclare le `script.sh` fichier du répertoire du bundle de tâches en tant que fichier d'entrée requis pour l'exécution de la tâche.

Ensuite, assurez-vous qu'aucun profil de stockage n'est configuré sur la CLI de Deadline Cloud, puis soumettez la tâche à la file d'attente `Q1` :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Le résultat de la CLI de Deadline Cloud après l'exécution de cette commande est le suivant :

```
Submitting to Queue: Q1
...
Hashing Attachments  [####################################]  100%
Hashing Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
```

```
    Total processing time of 0.0327 seconds at 1.19 KB/s.

Uploading Attachments  [####################################]  100%
Upload Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.25639 seconds at 152.0 B/s.

Waiting for Job to be created...
Submitted job bundle:
   job_attachments_devguide/
Job creation completed successfully
job-74148c13342e4514b63c7a7518657005
```

Lorsque vous soumettez la tâche, Deadline Cloud hache d'abord le `script.sh` fichier, puis le télécharge sur Amazon S3.

Deadline Cloud traite le compartiment S3 comme un espace de stockage adressable par le contenu. Les fichiers sont téléchargés vers des objets S3. Le nom de l'objet est dérivé d'un hachage du contenu du fichier. Si deux fichiers ont un contenu identique, ils ont la même valeur de hachage, quel que soit leur emplacement ou leur nom. Ce stockage adressable par contenu permet à Deadline Cloud d'éviter de télécharger un fichier s'il est déjà disponible.

Vous pouvez utiliser l'AWS CLI pour voir les objets qui ont été chargés sur Amazon S3 :

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

aws s3 ls s3://$Q1_S3_BUCKET --recursive
```

Deux objets ont été chargés sur S3 :

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— Le contenu `descript.sh`. La valeur de `87cb19095dd5d78fcaf56384ef0e6241` la clé d'objet est le hachage du contenu du fichier, et l'extension `xxh128` indique que la valeur de hachage a été calculée sous la forme d'un xxhash de 128 bits.
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— L'objet manifeste pour la soumission

de la tâche. Les valeurs <farm-id><queue-id>, et <guid> sont l'identifiant de votre ferme, l'identifiant de la file d'attente et une valeur hexadécimale aléatoire. La valeur a1d221c7fd97b08175b3872a37428e8c dans cet exemple est une valeur de hachage calculée à partir de la chaîne/home/cloudshell-user/job_attachments_devguide, le répertoire dans lequel se script.sh trouve le répertoire.

L'objet manifeste contient les informations relatives aux fichiers d'entrée sur un chemin racine spécifique téléchargés vers S3 dans le cadre de la soumission de la tâche. Téléchargez ce fichier manifeste (aws s3 cp s3://$Q1_S3_BUCKET/<objectname>). Son contenu est similaire à :

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "script.sh",
            "size": 39
        }
    ],
    "totalSize": 39
}
```

Cela indique que le fichier script.sh a été chargé et que le hachage du contenu de ce fichier est 87cb19095dd5d78fcaf56384ef0e6241 le même. Cette valeur de hachage correspond à la valeur du nom DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128 de l'objet. Il est utilisé par Deadline Cloud pour savoir quel objet télécharger pour le contenu de ce fichier.

Le schéma complet de ce fichier est disponible dans GitHub.

Lorsque vous utilisez cette CreateJob opération, vous pouvez définir l'emplacement des objets du manifeste. Vous pouvez utiliser l'GetJobopération pour voir l'emplacement :

```
{
    "attachments": {
        "file system": "COPIED",
        "manifests": [
            {
                "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
```

```
                "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/
a1d221c7fd97b08175b3872a37428e8c_input",
                "rootPath": "/home/cloudshell-user/job_attachments_devguide",
                "rootPathFormat": "posix"
            }
        ]
    },
    ...
}
```

Comment Deadline Cloud choisit les fichiers à télécharger

Les fichiers et répertoires que les pièces jointes aux tâches considèrent comme des entrées pour votre tâche sont les suivants :

- Les valeurs de tous les paramètres de tâche de `PATH` type -type définis dans le modèle de tâches du bundle de tâches avec une `dataFlow` valeur de `IN` ou`INOUT`.
- Les fichiers et répertoires répertoriés en tant qu'entrées dans le fichier de référence des actifs du bundle de tâches.

Si vous soumettez une tâche sans profil de stockage, tous les fichiers considérés pour le téléchargement sont chargés. Si vous soumettez une tâche avec un profil de stockage, les fichiers ne sont pas chargés vers Amazon S3 s'ils se trouvent dans les emplacements du système de fichiers de `SHARED` type du profil de stockage, qui sont également des emplacements de système de fichiers obligatoires pour la file d'attente. Ces emplacements sont censés être disponibles sur les hôtes de travail qui exécutent la tâche. Il n'est donc pas nécessaire de les télécharger sur S3.

Dans cet exemple, vous créez des emplacements de système de `SHARED` fichiers `WSAll` dans votre CloudShell environnement AWS, puis vous ajoutez des fichiers à ces emplacements de système de fichiers. Utilisez la commande suivante :

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

sudo mkdir -p /shared/common /shared/projects/project1 /shared/projects/project2
sudo chown -R cloudshell-user:cloudshell-user /shared

for d in /shared/common /shared/projects/project1 /shared/projects/project2; do
  echo "File contents for $d" > ${d}/file.txt
done
```

Ajoutez ensuite un fichier de références d'actifs au bundle de tâches qui inclut tous les fichiers que vous avez créés en tant qu'entrées pour le travail. Utilisez la commande suivante :

```
cat > ${HOME}/job_attachments_devguide/asset_references.yaml << EOF
assetReferences:
  inputs:
    filenames:
    - /shared/common/file.txt
    directories:
    - /shared/projects/project1
    - /shared/projects/project2
EOF
```

Configurez ensuite la CLI de Deadline Cloud pour soumettre les tâches avec le profil de `WSAll` stockage, puis soumettez le bundle de tâches :

```

# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Deadline Cloud télécharge deux fichiers sur Amazon S3 lorsque vous soumettez la tâche. Vous pouvez télécharger les objets du manifeste de la tâche depuis S3 pour voir les fichiers téléchargés :

```
for manifest in $( \
  aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID \
    --query 'attachments.manifests[].inputManifestPath' \
    | jq -r '.[]'
); do
  echo "Manifest object: $manifest"
  aws s3 cp --quiet s3://$Q1_S3_BUCKET/DeadlineCloud/Manifests/$manifest /dev/stdout |
 jq .
done
```

Dans cet exemple, il existe un seul fichier manifeste dont le contenu est le suivant :

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "home/cloudshell-user/job_attachments_devguide/script.sh",
            "size": 39
        },
        {
            "hash": "af5a605a3a4e86ce7be7ac5237b51b79",
            "mtime": 1721163773582362,
            "path": "shared/projects/project2/file.txt",
            "size": 44
        }
    ],
    "totalSize": 83
}
```

Utilisez l'GetJob opération pour le manifeste pour vérifier qu'il s'`rootPath`agit de «/».

```
aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID --query
 'attachments.manifests[*]'
```

Le chemin racine d'un ensemble de fichiers d'entrée est toujours le sous-chemin commun le plus long de ces fichiers. Si votre tâche a été soumise à la Windows place et que certains fichiers d'entrée n'ont aucun sous-chemin commun parce qu'ils se trouvent sur des lecteurs différents, vous voyez un chemin racine distinct sur chaque lecteur. Les chemins d'un manifeste sont toujours relatifs au chemin racine du manifeste. Les fichiers d'entrée qui ont été téléchargés sont donc les suivants :

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— Le fichier de script contenu dans le bundle de tâches.
- `/shared/projects/project2/file.txt`— Le fichier situé dans un emplacement `SHARED` du système de fichiers dans le profil `WSAll` de stockage qui ne figure pas dans la liste des emplacements de système de fichiers requis pour la file d'attente`Q1`.

Les fichiers situés dans les emplacements du système de fichiers `FSCommon` (`/shared/common/file.txt`) et `FS1` (`/shared/projects/project1/file.txt`) ne figurent pas dans la liste. Cela est dû au fait que ces emplacements de système de fichiers se trouvent `SHARED` dans le profil `WSAll` de stockage et qu'ils figurent tous deux dans la liste des emplacements de système de fichiers requis dans la file d'attente`Q1`.

Vous pouvez voir les emplacements du système de fichiers pris en compte `SHARED` pour une tâche soumise avec un profil de stockage particulier lors de l'GetStorageProfileForQueue opération. Pour rechercher le profil de stockage `WSAll` pour la file d'attente, `Q1` utilisez la commande suivante :

```
aws deadline get-storage-profile --farm-id $FARM_ID --storage-profile-id $WSALL_ID

aws deadline get-storage-profile-for-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID --
storage-profile-id $WSALL_ID
```

Comment les tâches trouvent-elles les fichiers d'entrée des pièces jointes

Pour qu'une tâche utilise les fichiers que Deadline Cloud télécharge sur Amazon S3 à l'aide de pièces jointes, elle a besoin que ces fichiers soient disponibles via le système de fichiers sur les hôtes de travail. Lorsqu'une session pour votre tâche s'exécute sur un hôte de travail, Deadline Cloud télécharge les fichiers d'entrée de la tâche dans un répertoire temporaire sur le disque local de l'hôte de travail et ajoute des règles de mappage de chemin pour chacun des chemins racines de la tâche vers son emplacement dans le système de fichiers sur le disque local.

Pour cet exemple, lancez l'agent de travail Deadline Cloud dans un CloudShell onglet AWS. Laissez toutes les tâches déjà soumises terminer leur exécution, puis supprimez les journaux des tâches du répertoire des journaux :

```
rm -rf ~/devdemo-logs/queue-*
```

Le script suivant modifie le bundle de tâches pour afficher tous les fichiers du répertoire de travail temporaire de la session ainsi que le contenu du fichier de règles de mappage de chemins, puis soumet un job avec le bundle modifié :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
```

```
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

cat > ~/job_attachments_devguide/script.sh << EOF
#!/bin/bash

echo "Session working directory is: \$(pwd)"
echo
echo "Contents:"
find . -type f
echo
echo "Path mapping rules file: \$1"
jq . \$1
EOF

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/bash
        args:
        - "{{Param.ScriptFile}}"
        - "{{Session.PathMappingRulesFile}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Vous pouvez consulter le journal de l'exécution de la tâche une fois qu'elle a été exécutée par le travailleur dans votre AWS CloudShell environnement :

```
cat demoenv-logs/queue-*/session*.log
```

Le journal indique que la première chose qui se produit au cours de la session est que les deux fichiers d'entrée de la tâche sont téléchargés vers le travailleur :

```
2024-07-17 01:26:37,824 INFO ================================================
2024-07-17 01:26:37,825 INFO --------- Job Attachments Download for Job
2024-07-17 01:26:37,825 INFO ================================================
2024-07-17 01:26:37,825 INFO Syncing inputs using Job Attachments
2024-07-17 01:26:38,116 INFO Downloaded 142.0 B / 186.0 B of 2 files (Transfer rate:
 0.0 B/s)
2024-07-17 01:26:38,174 INFO Downloaded 186.0 B / 186.0 B of 2 files (Transfer rate:
 733.0 B/s)
2024-07-17 01:26:38,176 INFO Summary Statistics for file downloads:
Processed 2 files totaling 186.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.09752 seconds at 1.91 KB/s.
```

Vient ensuite le résultat de `script.sh` run by the job :

- Les fichiers d'entrée téléchargés lors de la soumission de la tâche se trouvent dans un répertoire dont le nom commence par « assetroot » dans le répertoire temporaire de la session.
- Les chemins des fichiers d'entrée ont été déplacés par rapport au répertoire « assetroot » plutôt que par rapport au chemin racine du manifeste d'entrée () de la tâche. "/"
- Le fichier de règles de mappage de chemins contient une règle supplémentaire qui "/" correspond au chemin absolu du répertoire « assetroot ».

Par exemple :

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
```

```
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }
```

> **Note**
>
> Si la tâche que vous soumettez comporte plusieurs manifestes avec des chemins racines différents, il existe un répertoire nommé « assetroot » différent pour chacun des chemins racines.

Si vous devez référencer l'emplacement du système de fichiers déplacé de l'un de vos fichiers d'entrée, répertoires ou emplacements de système de fichiers, vous pouvez soit traiter le fichier de règles de mappage de chemins dans votre tâche et effectuer le remappage vous-même, soit ajouter un paramètre de tâche de PATH type au modèle de tâche de votre ensemble de tâches et transmettre la valeur que vous devez remapper en tant que valeur de ce paramètre. Par exemple, l'exemple suivant modifie le lot de tâches pour qu'il comporte l'un de ces paramètres de tâche, puis soumet une tâche avec l'emplacement du système de fichiers `/shared/projects/project2` comme valeur :

```
cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: LocationToRemap
```

```
  type: PATH
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
        - "The location of {{RawParam.LocationToRemap}} in the session is
 {{Param.LocationToRemap}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/ \
  -p LocationToRemap=/shared/projects/project2
```

Le fichier journal de l'exécution de cette tâche contient sa sortie :

```
2024-07-17 01:40:35,283 INFO Output:
2024-07-17 01:40:35,284 INFO The location of /shared/projects/project2 in the session
 is /sessions/session-5b33f/assetroot-assetroot-3751a
```

Obtenir des fichiers de sortie à partir d'une tâche

Cet exemple montre comment Deadline Cloud identifie les fichiers de sortie générés par vos tâches, décide de télécharger ou non ces fichiers sur Amazon S3 et comment vous pouvez obtenir ces fichiers de sortie sur votre poste de travail.

Utilisez le lot de `job_attachments_devguide_output` tâches au lieu du lot de `job_attachments_devguide` tâches dans cet exemple. Commencez par créer une copie du bundle dans votre AWS CloudShell environnement à partir de votre clone du GitHub référentiel d'échantillons de Deadline Cloud :

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/
```

La différence importante entre cet ensemble de tâches et le lot de `job_attachments_devguide` tâches réside dans l'ajout d'un nouveau paramètre de tâche dans le modèle de tâche :

```
...
parameterDefinitions:
...
```

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

La `dataFlow` propriété du paramètre possède la valeur`OUT`. Deadline Cloud utilise la valeur des paramètres de `dataFlow` tâche avec une valeur égale `OUT` ou `INOUT` en tant que résultats de votre tâche. Si l'emplacement du système de fichiers transmis sous forme de valeur à ces types de paramètres de tâche est remappé à un emplacement du système de fichiers local sur le serveur de travail qui exécute le travail, Deadline Cloud recherchera de nouveaux fichiers à cet emplacement et les téléchargera sur Amazon S3 en tant que résultats de travail.

Pour voir comment cela fonctionne, lancez d'abord l'agent de travail Deadline Cloud dans un AWS CloudShell onglet. Laissez toutes les tâches déjà soumises terminer leur exécution. Supprimez ensuite les journaux des tâches du répertoire des journaux :

```
rm -rf ~/devdemo-logs/queue-*
```

Soumettez ensuite une tâche avec cet ensemble de tâches. Une fois que le worker a CloudShell exécuté vos courses, consultez les journaux :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output
```

Le journal indique qu'un fichier a été détecté en sortie et chargé sur Amazon S3 :

```
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Uploading output files to Job Attachments
```

```
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Started syncing outputs using Job Attachments
2024-07-17 02:13:10,955 INFO Found 1 file totaling 117.0 B in output directory: /
sessions/session-7efa/assetroot-assetroot-3751a/output_dir
2024-07-17 02:13:10,956 INFO Uploading output manifest to
 DeadlineCloud/Manifests/farm-0011/queue-2233/job-4455/step-6677/
task-6677-0/2024-07-17T02:13:10.835545Z_sessionaction-8899-1/
c6808439dfc59f86763aff5b07b9a76c_output
2024-07-17 02:13:10,988 INFO Uploading 1 output file to S3: s3BucketName/DeadlineCloud/
Data
2024-07-17 02:13:11,011 INFO Uploaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:13:11,011 INFO Summary Statistics for file uploads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.02281 seconds at 5.13 KB/s.
```

Le journal indique également que Deadline Cloud a créé un nouvel objet manifeste dans le compartiment Amazon S3 configuré pour être utilisé par les pièces jointes aux tâches en file d'attenteQ1. Le nom de l'objet manifeste est dérivé de la batterie de serveurs, de la file d'attente, de la tâche, de l'étape, de la tâche, de l'horodatage et des `sessionaction` identifiants de la tâche qui a généré le résultat. Téléchargez ce fichier manifeste pour voir où Deadline Cloud a placé les fichiers de sortie pour cette tâche :

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

# Fill this in with the object name from your log
OBJECT_KEY="DeadlineCloud/Manifests/..."

aws s3 cp --quiet s3://$Q1_S3_BUCKET/$OBJECT_KEY /dev/stdout | jq .
```

Le manifeste se présente comme suit :

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
```

```
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

Cela montre que le contenu du fichier de sortie est enregistré sur Amazon S3 de la même manière que les fichiers d'entrée des tâches. Comme pour les fichiers d'entrée, le fichier de sortie est stocké dans S3 avec un nom d'objet contenant le hachage du fichier et le préfixe`DeadlineCloud/Data`.

```
$ aws s3 ls --recursive s3://$Q1_S3_BUCKET | grep 34178940e1ef9956db8ea7f7c97ed842
2024-07-17 02:13:11        117 DeadlineCloud/
Data/34178940e1ef9956db8ea7f7c97ed842.xxh128
```

Vous pouvez télécharger le résultat d'une tâche sur votre poste de travail à l'aide du moniteur Deadline Cloud ou de la CLI de Deadline Cloud :

```
deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID
```

La valeur du paramètre de `OutputDir` tâche dans la tâche soumise est`./output_dir`, de sorte que la sortie est téléchargée dans un répertoire appelé `output_dir` dans le répertoire du bundle de tâches. Si vous avez spécifié un chemin absolu ou un emplacement relatif différent comme valeur pour`OutputDir`, les fichiers de sortie seront plutôt téléchargés vers cet emplacement.

```
$ deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id
 $JOB_ID
Downloading output from Job 'Job Attachments Explorer: Output'

Summary of files to download:
    /home/cloudshell-user/job_attachments_devguide_output/output_dir/output.txt (1
 file)

You are about to download files which may come from multiple root directories. Here are
 a list of the current root directories:
[0] /home/cloudshell-user/job_attachments_devguide_output
> Please enter the index of root directory to edit, y to proceed without changes, or n
 to cancel the download (0, y, n) [y]:
```

```
Downloading Outputs  [####################################]  100%
Download Summary:
    Downloaded 1 files totaling 117.0 B.
    Total download time of 0.14189 seconds at 824.0 B/s.
    Download locations (total file counts):
        /home/cloudshell-user/job_attachments_devguide_output (1 file)
```

Utilisation des fichiers d'une étape dans une étape dépendante

Cet exemple montre comment une étape d'une tâche peut accéder aux sorties d'une étape dont elle dépend dans la même tâche.

Pour rendre les résultats d'une étape accessibles à une autre, Deadline Cloud ajoute des actions supplémentaires à une session afin de télécharger ces résultats avant d'exécuter des tâches dans la session. Vous lui indiquez à partir de quelles étapes télécharger les sorties en déclarant ces étapes comme des dépendances de l'étape qui doit utiliser les sorties.

Utilisez le `job_attachments_devguide_output` job bundle pour cet exemple. Commencez par créer une copie dans votre AWS CloudShell environnement à partir de votre clone du GitHub référentiel d'échantillons de Deadline Cloud. Modifiez-le pour ajouter une étape dépendante qui ne s'exécute qu'après l'étape existante et utilise le résultat de cette étape :

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/

cat >> job_attachments_devguide_output/template.yaml << EOF
- name: DependentStep
  dependencies:
  - dependsOn: Step
  script:
    actions:
      onRun:
        command: /bin/cat
        args:
        - "{{Param.OutputDir}}/output.txt"
EOF
```

La tâche créée avec cet ensemble de tâches modifié s'exécute sous la forme de deux sessions distinctes, une pour la tâche de l'étape « Étape », puis une seconde pour la tâche de l'étape « DependentStep ».

Démarrez d'abord l'agent de travail de Deadline Cloud dans un CloudShell onglet. Laissez toutes les tâches déjà soumises terminer leur exécution, puis supprimez les journaux des tâches du répertoire des journaux :

```
rm -rf ~/devdemo-logs/queue-*
```

Soumettez ensuite une tâche à l'aide de l'ensemble de `job_attachments_devguide_output` tâches modifié. Attendez qu'il ait fini de s'exécuter sur le travailleur de votre CloudShell environnement. Consultez les journaux des deux sessions :

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output

# Wait for the job to finish running, and then:

cat demoenv-logs/queue-*/session-*
```

Dans le journal de session de la tâche de l'étape nomméeDependentStep, deux actions de téléchargement distinctes sont exécutées :

```
2024-07-17 02:52:05,666 INFO ================================================
2024-07-17 02:52:05,666 INFO --------- Job Attachments Download for Job
2024-07-17 02:52:05,667 INFO ================================================
2024-07-17 02:52:05,667 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:05,928 INFO Downloaded 207.0 B / 207.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:05,929 INFO Summary Statistics for file downloads:
Processed 1 file totaling 207.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03954 seconds at 5.23 KB/s.
```

```
2024-07-17 02:52:05,979 INFO
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,979 INFO --------- Job Attachments Download for Step
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,980 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:06,133 INFO Downloaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:06,134 INFO Summary Statistics for file downloads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03227 seconds at 3.62 KB/s.
```

La première action télécharge le `script.sh` fichier utilisé par l'étape nommée « Étape ». La deuxième action télécharge les résultats de cette étape. Deadline Cloud détermine les fichiers à télécharger en utilisant le manifeste de sortie généré par cette étape comme manifeste d'entrée.

À la fin du même journal, vous pouvez voir le résultat de l'étape nommée DependentStep « » :

```
2024-07-17 02:52:06,213 INFO Output:
2024-07-17 02:52:06,216 INFO Script location: /sessions/session-5b33f/
assetroot-assetroot-3751a/script.sh
```

Création de limites de ressources pour les tâches

Les tâches soumises à Deadline Cloud peuvent dépendre de ressources partagées entre plusieurs tâches. Par exemple, une ferme peut avoir plus de travailleurs que de permis flottants pour une ressource spécifique. Il se peut également qu'un serveur de fichiers partagé ne soit en mesure de fournir des données qu'à un nombre limité de travailleurs en même temps. Dans certains cas, une ou plusieurs tâches peuvent exiger toutes ces ressources, ce qui entraîne des erreurs dues à l'indisponibilité des ressources lorsque de nouveaux travailleurs commencent à travailler.

Pour résoudre ce problème, vous pouvez utiliser des limites pour ces ressources limitées. Deadline Cloud tient compte de la disponibilité de ressources limitées et utilise ces informations pour garantir que les ressources sont disponibles au fur et à mesure que les nouveaux travailleurs démarrent, afin que les emplois soient moins susceptibles d'échouer en raison de l'indisponibilité des ressources.

Des limites sont créées pour l'ensemble de la ferme. Les tâches soumises à une file d'attente ne peuvent être soumises qu'aux limites associées à la file d'attente. Si vous spécifiez une limite pour une tâche qui n'est pas associée à la file d'attente, la tâche n'est pas compatible et ne sera pas exécutée.

Pour utiliser une limite, vous

- Création d'une limite
- Associer une limite et une file d'attente
- Soumettre une offre d'emploi nécessitant des limites

> **Note**
>
> Si vous exécutez une tâche dont les ressources sont limitées dans une file d'attente non associée à une limite, cette tâche peut consommer toutes les ressources. Si votre ressource est limitée, assurez-vous que toutes les étapes des tâches dans les files d'attente qui utilisent cette ressource sont associées à une limite.

Pour les limites définies dans un parc de serveurs, associées à une file d'attente et spécifiées dans une tâche, l'une des quatre situations suivantes peut se produire :

- Si vous créez une limite, que vous l'associez à une file d'attente et que vous spécifiez la limite dans le modèle d'une tâche, la tâche s'exécute et n'utilise que les ressources définies dans la limite.
- Si vous créez une limite, que vous la spécifiez dans un modèle de tâche, mais que vous n'associez pas la limite à une file d'attente, la tâche est marquée comme incompatible et ne sera pas exécutée.
- Si vous créez une limite, ne l'associez pas à une file d'attente et ne spécifiez pas la limite dans le modèle d'une tâche, la tâche s'exécute mais n'utilise pas la limite.
- Si vous n'utilisez aucune limite, la tâche s'exécute.

Si vous associez une limite à plusieurs files d'attente, celles-ci partagent les ressources limitées par cette limite. Par exemple, si vous créez une limite de 100 et qu'une file d'attente utilise 60 ressources, les autres files d'attente ne peuvent utiliser que 40 ressources. Lorsqu'une ressource est publiée, elle peut être utilisée par une tâche de n'importe quelle file d'attente.

Deadline Cloud fournit deux AWS CloudFormation indicateurs pour vous aider à surveiller les ressources fournies par une limite. Vous pouvez surveiller le nombre actuel de ressources utilisées et le nombre maximum de ressources disponibles dans la limite. Pour plus d'informations, consultez la section Mesures relatives aux limites de ressources dans le guide du développeur de Deadline Cloud.

Vous appliquez une limite à une étape de tâche dans un modèle de tâche. Lorsque vous spécifiez le nom du montant requis pour une limite dans la `amounts` section `hostRequirements` d'une étape et qu'une limite associée à ce montant `amountRequirementName` est associée à la file d'attente du travail, les tâches planifiées pour cette étape sont limitées par la limite de la ressource.

Si une étape nécessite une ressource limitée par une limite atteinte, les tâches de cette étape ne seront pas prises en charge par des travailleurs supplémentaires.

Vous pouvez appliquer plusieurs limites à une étape de travail. Par exemple, si l'étape utilise deux licences logicielles différentes, vous pouvez appliquer une limite distincte pour chaque licence. Si une étape nécessite deux limites et que la limite de l'une des ressources est atteinte, les tâches de cette étape ne seront pas prises en charge par d'autres travailleurs tant que les ressources ne seront pas disponibles.

Arrêter et supprimer des limites

Lorsque vous arrêtez ou supprimez l'association entre une file d'attente et une limite, une tâche utilisant la limite arrête de planifier les tâches à partir des étapes qui nécessitent cette limite et bloque la création de nouvelles sessions pour une étape.

Les tâches à l'état PRÊT restent prêtes, et les tâches reprennent automatiquement lorsque l'association entre la file d'attente et la limite redevient active. Vous n'avez pas besoin de demander des offres d'emploi.

Lorsque vous arrêtez ou supprimez l'association entre une file d'attente et une limite, deux options s'offrent à vous pour arrêter l'exécution des tâches :

- Arrêter et annuler des tâches : les travailleurs dont les sessions ont atteint la limite annulent toutes les tâches.
- Arrêtez et terminez l'exécution des tâches : les travailleurs dont les sessions ont atteint la limite terminent leurs tâches.

Lorsque vous supprimez une limite à l'aide de la console, les collaborateurs arrêtent d'abord d'exécuter les tâches immédiatement ou éventuellement lorsqu'elles sont terminées. Lorsque l'association est supprimée, les événements suivants se produisent :

- Les étapes nécessitant cette limite sont signalées comme non compatibles.
- L'intégralité de la tâche contenant ces étapes est annulée, y compris les étapes qui ne nécessitent pas de limite.

- La tâche est marquée comme non compatible.

Si la file d'attente associée à la limite est associée à un parc dont la capacité correspond au nom du montant requis pour la limite, ce parc continuera à traiter les tâches avec la limite spécifiée.

Création d'une limite

Vous créez une limite à l'aide de la console Deadline Cloud ou de l'CreateLimit opération de l'API Deadline Cloud. Les limites sont définies pour un parc, mais associées à des files d'attente. Après avoir créé une limite, vous pouvez l'associer à une ou plusieurs files d'attente.

Pour créer une limite

1. Dans le tableau de bord de la console Deadline Cloud (console Deadline Cloud), sélectionnez la ferme pour laquelle vous souhaitez créer une file d'attente.
2. Choisissez la ferme à laquelle ajouter la limite, cliquez sur l'onglet Limites, puis choisissez Créer une limite.
3. Fournissez les détails de la limite. Le nom de l'exigence de montant est le nom utilisé dans le modèle de tâche pour identifier la limite. Il doit commencer par le préfixe **`amount.`** suivi du nom du montant. Le nom du montant requis doit être unique dans les files d'attente associées à la limite.
4. Si vous choisissez Définir un montant maximum, il s'agit du nombre total de ressources autorisées par cette limite. Si vous choisissez Aucun montant maximum, l'utilisation des ressources n'est pas limitée. Même lorsque l'utilisation des ressources n'est pas limitée, la CloudWatch métrique `CurrentCount` Amazon est émise afin que vous puissiez suivre l'utilisation. Pour plus d'informations, consultez les CloudWatchstatistiques dans le guide du développeur de Deadline Cloud.
5. Si vous connaissez déjà les files d'attente qui devraient utiliser cette limite, vous pouvez les choisir dès maintenant. Il n'est pas nécessaire d'associer une file d'attente pour créer une limite.
6. Choisissez Créer une limite.

Associer une limite et une file d'attente

Après avoir créé une limite, vous pouvez associer une ou plusieurs files d'attente à la limite. Seules les files d'attente associées à une limite utilisent les valeurs spécifiées dans la limite.

Vous créez une association avec une file d'attente à l'aide de la console Deadline Cloud ou de l'CreateQueueLimitAssociation opération de l'API Deadline Cloud.

Pour associer une file d'attente à une limite

1. Dans le tableau de bord de la console Deadline Cloud (console Deadline Cloud), sélectionnez la ferme dans laquelle vous souhaitez associer une limite à une file d'attente.
2. Cliquez sur l'onglet Limites, choisissez la limite à laquelle associer une file d'attente, puis choisissez Modifier la limite.
3. Dans la section Associer les files d'attente, choisissez les files d'attente à associer à la limite.
4. Sélectionnez Enregistrer les modifications.

Soumettre une offre d'emploi nécessitant des limites

Vous appliquez une limite en la spécifiant comme exigence d'hôte pour le travail ou l'étape du travail. Si vous ne spécifiez pas de limite dans une étape et que cette étape utilise une ressource associée, l'utilisation de l'étape n'est pas prise en compte dans la limite lorsque les tâches sont planifiées.

Certains émetteurs de Deadline Cloud vous permettent de définir une exigence en matière d'hôte. Vous pouvez spécifier le nom du montant requis pour la limite dans l'expéditeur pour appliquer la limite.

Si votre auteur ne prend pas en charge l'ajout d'exigences relatives à l'hôte, vous pouvez également appliquer une limite en modifiant le modèle de tâche correspondant à la tâche.

Pour appliquer une limite à une étape d'une tâche dans le lot de tâches

1. Ouvrez le modèle de tâche correspondant à la tâche à l'aide d'un éditeur de texte. Le modèle de tâche se trouve dans le répertoire des ensembles de tâches correspondant à la tâche. Pour plus d'informations, consultez la section Job bundles dans le guide du développeur de Deadline Cloud.
2. Trouvez la définition de l'étape à laquelle appliquer la limite.
3. Ajoutez ce qui suit à la définition de l'étape. Remplacez *amount.name* le nom de votre limite par le montant requis. Pour une utilisation normale, vous devez définir la `min` valeur sur 1.

   YAML

   ```
   hostRequirements:
   ```

```
    amounts:
    - name: amount.name
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name",
            "min": "1"
        }
    ]
}
```

Vous pouvez ajouter plusieurs limites à une étape de travail comme suit. Remplacez *amount.name_1* et *amount.name_2* par les noms des exigences relatives au montant de vos limites.

YAML

```
  hostRequirements:
    amounts:
    - name: amount.name_1
      min: 1
    - name: amount.name_2
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name_1",
            "min": "1"
        },
        {
            "name": "amount.name_2",
            "min": "1"
        }
    ]
```

```
}
```

4. Enregistrez les modifications apportées au modèle de tâche.

Comment soumettre une offre d'emploi à Deadline Cloud

Il existe de nombreuses manières de soumettre des offres d'emploi à AWS Deadline Cloud. Cette section décrit certaines des manières dont vous pouvez soumettre des tâches à l'aide des outils fournis par Deadline Cloud ou en créant vos propres outils personnalisés pour vos charges de travail.

- Depuis un terminal : lorsque vous développez un ensemble de tâches pour la première fois ou lorsque les utilisateurs qui soumettent une tâche sont à l'aise avec la ligne de commande
- À partir d'un script, pour personnaliser et automatiser les charges de travail
- À partir d'une application : lorsque le travail de l'utilisateur est effectué dans une application ou lorsque le contexte d'une application est important.

Les exemples suivants utilisent la bibliothèque `deadline` Python et l'outil de ligne de `deadline` commande. Les deux sont disponibles PyPiet hébergés sur GitHub.

Rubriques

- Soumettre une offre d'emploi à Deadline Cloud depuis un terminal
- Soumettre une tâche à Deadline Cloud à l'aide d'un script
- Soumettre une offre d'emploi dans le cadre d'une candidature

Soumettre une offre d'emploi à Deadline Cloud depuis un terminal

En utilisant uniquement un ensemble de tâches et la CLI de Deadline Cloud, vous ou vos utilisateurs plus techniques pouvez rapidement itérer sur la rédaction de lots de tâches pour tester la soumission d'une tâche. Utilisez la commande suivante pour soumettre un ensemble de tâches :

```
deadline bundle submit <path-to-job-bundle>
```

Si vous soumettez un ensemble de tâches dont les paramètres ne sont pas définis par défaut, vous pouvez les spécifier à l'aide de l'`--parameter`option `-p`/.

```
deadline bundle submit <path-to-job-bundle> -p <parameter-name>=<parameter-value> -
p ...
```

Pour obtenir la liste complète des options disponibles, exécutez la commande d'aide :

```
deadline bundle submit --help
```

Soumettre une tâche à Deadline Cloud à l'aide d'une interface graphique

La CLI de Deadline Cloud est également dotée d'une interface utilisateur graphique qui permet aux utilisateurs de voir les paramètres qu'ils doivent fournir avant de soumettre une tâche. Si vos utilisateurs préfèrent ne pas interagir avec la ligne de commande, vous pouvez écrire un raccourci sur le bureau qui ouvre une boîte de dialogue pour soumettre un ensemble de tâches spécifique :

```
deadline bundle gui-submit <path-to-job-bundle>
```

Utilisez l'`--browse`option can afin que l'utilisateur puisse sélectionner un ensemble de tâches :

```
deadline bundle gui-submit --browse
```

Pour obtenir la liste complète des options disponibles, exécutez la commande d'aide :

```
deadline bundle gui-submit --help
```

Soumettre une tâche à Deadline Cloud à l'aide d'un script

Pour automatiser la soumission de tâches à Deadline Cloud, vous pouvez les scripter à l'aide d'outils tels que bash, Powershell et de fichiers batch.

Vous pouvez ajouter des fonctionnalités telles que le remplissage des paramètres de tâche à partir de variables d'environnement ou d'autres applications. Vous pouvez également soumettre plusieurs tâches d'affilée ou créer un script pour créer un ensemble de tâches à soumettre.

Soumettre une offre d'emploi en Python

Deadline Cloud dispose également d'une bibliothèque Python open source pour interagir avec le service. Le code source est disponible sur GitHub.

La bibliothèque est disponible sur pypi via pip (). `pip install deadline` Il s'agit de la même bibliothèque que celle utilisée par l'outil Deadline Cloud CLI :

```
from deadline.client import api

job_bundle_path = "/path/to/job/bundle"
job_parameters = [
    {
        "name": "parameter_name",
        "value": "parameter_value"
    },
]

job_id = api.create_job_from_job_bundle(
    job_bundle_path,
    job_parameters
)
print(job_id)
```

Pour créer une boîte de dialogue comme la `deadline bundle gui-submit` commande, vous pouvez utiliser la `show_job_bundle_submitter` fonction du `deadline.client.ui.job_bundle_submitter`.

L'exemple suivant démarre une application Qt et montre l'expéditeur du bundle de tâches :

```
# The GUI components must be installed with pip install "deadline[gui]"
import sys
from qtpy.QtWidgets import QApplication
from deadline.client.ui.job_bundle_submitter import show_job_bundle_submitter

app = QApplication(sys.argv)
submitter = show_job_bundle_submitter(browse=True)
submitter.show()
app.exec()
print(submitter.create_job_response)
```

Pour créer votre propre boîte de dialogue, vous pouvez utiliser la `SubmitJobToDeadlineDialog` classe dans `deadline.client.ui.dialogs.submit_job_to_deadline_dialog`. Vous pouvez transmettre des valeurs, intégrer votre propre onglet spécifique à la tâche et déterminer comment le lot de tâches est créé (ou transmis).

Soumettre une offre d'emploi dans le cadre d'une candidature

Pour permettre aux utilisateurs de soumettre facilement des tâches, vous pouvez utiliser les environnements d'exécution de script ou les systèmes de plugins fournis par une application. Les utilisateurs disposent d'une interface familière et vous pouvez créer de puissants outils qui les aident lors de la soumission d'une charge de travail.

Intégrer des offres d'emploi dans une candidature

Cet exemple montre comment soumettre des offres d'emploi que vous mettez à disposition dans l'application.

Pour permettre à un utilisateur d'accéder à ces ensembles de tâches, créez un script intégré dans un élément de menu qui lance la CLI de Deadline Cloud.

Le script suivant permet à l'utilisateur de sélectionner le lot de tâches :

```
deadline bundle gui-submit --install-gui
```

Pour utiliser un ensemble de tâches spécifique dans un élément de menu à la place, utilisez ce qui suit :

```
deadline bundle gui-submit </path/to/job/bundle> --install-gui
```

Cela ouvre une boîte de dialogue dans laquelle l'utilisateur peut modifier les paramètres, les entrées et les sorties de la tâche, puis soumettre la tâche. Vous pouvez avoir différents éléments de menu pour différents ensembles de tâches à soumettre par un utilisateur dans le cadre d'une candidature.

Si le travail que vous soumettez avec un ensemble de tâches contient des paramètres et des références d'actifs similaires dans toutes les soumissions, vous pouvez renseigner les valeurs par défaut dans le lot de tâches sous-jacent.

Obtenir des informations à partir d'une application

Pour extraire des informations d'une application afin que les utilisateurs n'aient pas à les ajouter manuellement à la soumission, vous pouvez intégrer Deadline Cloud à l'application afin que vos utilisateurs puissent soumettre des tâches via une interface familière sans avoir à quitter l'application ou à utiliser des outils de ligne de commande.

Si votre application dispose d'un environnement d'exécution de script qui prend en charge Python et pyside/pyqt, vous pouvez utiliser les composants de l'interface graphique de la bibliothèque cliente de Deadline Cloud pour créer une interface utilisateur. Pour un exemple, consultez Deadline Cloud pour l'intégration de Maya sur GitHub.

La bibliothèque cliente de Deadline Cloud propose les opérations suivantes pour vous aider à fournir une expérience utilisateur intégrée solide :

- Extrayez les paramètres d'environnement de la file d'attente, les paramètres de tâche et les références aux actifs à partir de variables d'environnement et en appelant le SDK de l'application.
- Définissez les paramètres dans le bundle de tâches. Pour éviter de modifier le lot d'origine, vous devez en faire une copie et envoyer la copie.

Si vous utilisez la `deadline bundle gui-submit` commande pour soumettre le bundle de tâches, vous devez programmer les `asset_references.yaml` fichiers `parameter_values.yaml` et pour transmettre les informations de l'application. Pour plus d'informations sur ces fichiers, consultezModèles Open Job Description (OpenJD) pour Deadline Cloud.

Si vous avez besoin de contrôles plus complexes que ceux proposés par OpenJD, si vous souhaitez soustraire le travail à l'utilisateur ou si vous souhaitez que l'intégration corresponde au style visuel de l'application, vous pouvez écrire votre propre boîte de dialogue qui appelle la bibliothèque cliente de Deadline Cloud pour soumettre le travail.

Planifier des tâches dans Deadline Cloud

Une fois que vous avez créé une tâche, AWS Deadline Cloud planifie son traitement sur une ou plusieurs flottes associées à une file d'attente. La flotte qui traite une tâche particulière est choisie en fonction de la configuration de planification, des capacités configurées pour la flotte et des exigences de l'hôte pour une étape spécifique.

Les sections suivantes fournissent des informations détaillées sur le processus de planification d'une tâche.

Configurations de planification

Vous pouvez configurer la façon dont Deadline Cloud planifie les tâches dans une file d'attente en définissant une configuration de planification dans la file d'attente. La configuration de planification contrôle la façon dont les travailleurs sont répartis entre les tâches.

Vous pouvez définir la configuration de planification à l'aide de la console Deadline Cloud ou en appelant le CreateQueueou UpdateQueue APIs.

Trois configurations de planification sont disponibles :

- Priorité, first-in-first-out (`priorityFifo`) — Planifie d'abord la tâche la plus prioritaire et la plus ancienne soumise (par défaut).
- Priorité équilibrée (`priorityBalanced`) — Répartit les travailleurs de manière égale entre les tâches les plus prioritaires.
- Pondéré, équilibré (`weightedBalanced`) — Utilise une formule pondérée pour déterminer comment les travailleurs sont répartis entre les tâches.

Dans toutes les configurations de planification, les tâches en cours s'exécutent jusqu'à leur fin avant qu'une nouvelle décision de planification ne soit prise. Si vous modifiez la configuration de planification pendant que les tâches sont en cours d'exécution, la modification ne s'applique que lorsque les travailleurs seront affectés ensuite. Les tâches en cours ne sont ni interrompues ni réaffectées.

Priorité, first-in-first-out

La priorité first-in-first-out (`priorityFifo`) est la configuration de planification par défaut pour les nouvelles files d'attente. Deadline Cloud assigne d'abord les collaborateurs à la tâche la plus prioritaire. Lorsque plusieurs tâches partagent la même priorité, la tâche la plus ancienne (la plus ancienne soumise) reçoit en premier tous les travailleurs disponibles.

Utilisez le FIFO prioritaire lorsque vous souhaitez un ordre strict des tâches. Cette configuration est appropriée lorsque les tâches doivent être terminées une par une dans l'ordre dans lequel elles ont été soumises, comme les étapes séquentielles du pipeline ou le traitement par lots où chaque tâche doit se terminer avant le début de la suivante.

Cette configuration ne comporte aucun paramètre supplémentaire.

Priorité, équilibrée

Priority, balanced (`priorityBalanced`) répartit les travailleurs de manière égale entre tous les emplois au niveau de priorité le plus élevé. Lorsqu'une seule tâche a la priorité la plus élevée, Deadline Cloud affecte tous les collaborateurs à cette tâche. Lorsque plusieurs tâches ont la plus haute priorité, les travailleurs sont répartis équitablement entre eux. Si les travailleurs ne peuvent pas

être répartis équitablement, les travailleurs supplémentaires sont répartis entre les emplois les plus prioritaires.

Utilisez l'équilibre des priorités lorsque plusieurs artistes ou utilisateurs soumettent des travaux avec la même priorité et que chaque utilisateur a besoin d'un feedback immédiat. Cette configuration garantit qu'aucune tâche ne monopolise tous les travailleurs disponibles, de sorte que tous les utilisateurs se voient attribuer des travailleurs peu de temps après leur soumission.

Si un emploi a moins de tâches restantes que sa part de travailleurs, les travailleurs excédentaires sont redistribués vers d'autres emplois ayant le même niveau de priorité. Si tous les emplois les plus prioritaires sont entièrement attribués, les travailleurs excédentaires accèdent aux emplois du niveau de priorité le plus élevé suivant.

Cette configuration possède le paramètre suivant :

`renderingTaskBuffer`

Contrôle l'adhérence des travailleurs. Un travailleur passe de sa tâche actuelle à une autre tâche ayant la même priorité uniquement si la différence entre les tâches de rendu dépasse cette `renderingTaskBuffer` valeur. Une valeur plus élevée permet aux travailleurs de conserver leur emploi actuel plus longtemps, ce qui réduit les changements de contexte. La valeur par défaut est 1.

Pondéré, équilibré

Weighted, balanced (`weightedBalanced`) utilise une formule pour calculer le poids de chaque tâche. Deadline Cloud assigne d'abord les travailleurs au poste le plus important. Si plusieurs emplois ont le même poids, les travailleurs sont répartis entre eux.

Utilisez l'équilibre pondéré lorsque vous avez besoin d'un contrôle précis sur la façon dont les employés sont répartis entre les tâches, avec des priorités, des taux d'erreur et des délais de soumission variables. Cette configuration convient aux environnements de parc de rendu complexes dans lesquels vous souhaitez trouver l'équilibre entre la priorité des tâches, l'âge de la tâche, la gestion des erreurs et la rigidité des travailleurs.

Le poids de chaque tâche est calculé comme suit :

```
weight = (job.Priority * priorityWeight) +
         (job.Errors * errorWeight) +
         ((currentTimeInSeconds - job.SubmissionTime) * submissionTimeWeight) +
```

```
        ((job.RenderingTasks - renderingTaskBuffer) * renderingTaskWeight)
```

Le `renderingTaskBuffer` composant n'est appliqué que si le travailleur travaille actuellement sur le poste. `renderingTaskWeight`Il est généralement défini sur une valeur négative afin que les tâches auxquelles des travailleurs ont été affectés reçoivent une pondération inférieure, plaçant ainsi les autres tâches en tête de la file d'attente. `errorWeight`Il est également généralement négatif, de sorte que les tâches comportant des erreurs ne sont pas hiérarchisées. Vous pouvez utiliser les remplacements de planification pour les tâches à priorité minimale et maximale.

Cette configuration comporte les paramètres suivants :

`priorityWeight`

Le poids appliqué à la priorité d'une tâche. Une valeur positive signifie que les tâches les plus prioritaires sont planifiées en premier. La valeur par défaut est `100.0`. Gamme : `0` jusqu'à`10000`.

`errorWeight`

Le poids appliqué au nombre d'erreurs d'une tâche. Une valeur négative signifie que les tâches sans erreur sont planifiées en premier. La valeur par défaut est `-10.0`. Gamme : `-10000` jusqu'à`10000`.

`submissionTimeWeight`

Le poids appliqué au temps de soumission d'une tâche (en secondes). Une valeur positive signifie que les tâches soumises précédemment sont planifiées en premier. La valeur par défaut est `3.0`. Gamme : `0` jusqu'à`10000`.

`renderingTaskWeight`

Le poids appliqué au nombre de tâches actuellement affichées pour une tâche. Une valeur négative signifie que les prochaines tâches impliquant moins de travailleurs sont planifiées. La valeur par défaut est `-100.0`. Gamme : `-10000` jusqu'à`10000`.

`renderingTaskBuffer`

Nombre de tâches de rendu avant que le poids des tâches de rendu ne prenne effet. Une valeur positive permet aux travailleurs de conserver leur emploi actuel. La valeur par défaut est 1. Gamme : `0` jusqu'à`1000`.

`maxPriorityOverride`

Facultatif. Lorsqu'elle est définie sur`alwaysScheduleFirst`, les tâches ayant la priorité maximale (100) sont toujours planifiées avant les autres tâches, quelle que soit la formule

pondérée. Lorsque plusieurs tâches ont la priorité maximale, les liens sont rompus à l'aide de la formule pondérée standard. Lorsque la dérogation est absente, les tâches prioritaires maximales utilisent la formule pondérée standard sans traitement spécial.

`minPriority0verride`

Facultatif. Lorsqu'elle est définie sur`alwaysScheduleLast`, les tâches ayant la priorité minimale (0) sont toujours planifiées après les autres tâches, quelle que soit la formule pondérée. Lorsque plusieurs tâches ont la priorité minimale, les liens sont rompus à l'aide de la formule pondérée standard. Lorsque la dérogation est absente, les tâches prioritaires minimales utilisent la formule pondérée standard sans traitement spécial.

Déterminer la compatibilité de la flotte

Après avoir créé une tâche, Deadline Cloud vérifie les exigences en matière d'hôte pour chaque étape de la tâche par rapport aux capacités des flottes associées à la file d'attente à laquelle la tâche a été soumise. Si une flotte répond aux exigences de l'hôte, le poste est confié à l'`READY`État.

Si une étape de la tâche comporte des exigences qui ne peuvent pas être satisfaites par une flotte associée à la file d'attente, le statut de l'étape est défini sur`NOT_COMPATIBLE`. De plus, les autres étapes de la tâche sont annulées.

Les capacités d'une flotte sont définies au niveau de la flotte. Même si un travailleur d'un parc répond aux exigences du poste, aucune tâche ne lui sera affectée si son parc ne répond pas aux exigences du poste.

Le modèle de tâche suivant comporte une étape qui spécifie les exigences de l'hôte pour l'étape :

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
        - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
```

```
    # Capabilities starting with "amount." are amount capabilities. If they start with
 "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
 usage.
    - name: amount.worker.vcpu
      min: 4
      max: 8
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Cette tâche peut être planifiée pour une flotte dotée des fonctionnalités suivantes :

```
{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
```

Cette tâche ne peut pas être planifiée pour une flotte dotée de l'une des fonctionnalités suivantes :

```
{
    "vCpuCount": {"min": 4},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

{
    "vCpuCount": {"max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host
 requirement.

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
```

```
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
    The osFamily doesn't match.
```

Dimensionnement du parc

Lorsqu'une tâche est attribuée à un parc géré par des services compatibles, le parc est redimensionné automatiquement. Le nombre de travailleurs de la flotte varie en fonction du nombre de tâches pouvant être exécutées par la flotte.

Lorsqu'une tâche est attribuée à un parc géré par le client, il se peut que des employés existent déjà ou qu'ils puissent être créés à l'aide de la mise à l'échelle automatique basée sur les événements. Pour plus d'informations, consultez la section Utiliser EventBridge pour gérer les événements de dimensionnement automatique dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.

Séances

Les tâches d'une tâche sont divisées en une ou plusieurs sessions. Les travailleurs exécutent les sessions pour configurer l'environnement, exécuter les tâches, puis détruire l'environnement. Chaque session est composée d'une ou de plusieurs actions que le travailleur doit effectuer.

Au fur et à mesure qu'un collaborateur exécute des actions de section, des actions de session supplémentaires peuvent lui être envoyées. Le travailleur réutilise les environnements existants et les pièces jointes aux tâches au cours de la session pour effectuer les tâches de manière plus efficace.

Pour les employés de flotte gérés par des services, les répertoires de sessions sont supprimés à la fin de la session, mais les autres répertoires sont conservés entre les sessions. Ce comportement vous permet de mettre en œuvre des stratégies de mise en cache pour les données qui peuvent être réutilisées au cours de plusieurs sessions. Pour mettre en cache les données entre les sessions, stockez-les dans le répertoire personnel de l'utilisateur exécutant la tâche. Par exemple, les packages conda sont mis en cache dans le répertoire personnel de l'utilisateur à l'adresse `C:\Users\job-user\.conda-pkgs` on Windows workers et `/home/job-user/.conda-pkgs` on Linux workers. Ces données restent disponibles jusqu'à ce que le travailleur arrête ses activités.

Les pièces jointes aux tâches sont créées par l'émetteur que vous utilisez dans le cadre de votre offre de tâches Deadline Cloud CLI. Vous pouvez également créer des pièces jointes à des tâches à l'aide de l'`--attachments`option de `create-job` AWS CLI commande. Les environnements sont définis à deux endroits : les environnements de file d'attente attachés à une file d'attente spécifique et les environnements de travail et d'étapes définis dans le modèle de travail.

Il existe quatre types d'actions de session :

- `syncInputJobAttachments`— Télécharge les pièces jointes aux tâches saisies vers le travailleur.
- `envEnter`— Exécute les `onEnter` actions pour un environnement.
- `taskRun`— Exécute les onRun actions correspondant à une tâche.
- `envExit`— Exécute les `onExit` actions pour un environnement.

Le modèle de tâche suivant comporte un environnement par étapes. Il contient une `onEnter` définition pour configurer l'environnement des étapes, une onRun définition qui définit la tâche à exécuter et une `onExit` définition pour démonter l'environnement des étapes. Les sessions créées pour cette tâche incluront une `envEnter` action, une ou plusieurs `taskRun` actions, puis une `envExit` action.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
      actions:
        onEnter:
          command: MayaAdaptor
          args:
          - daemon
          - start
          - --init-data
          - file://{{Env.File.initData}}
        onExit:
          command: MayaAdaptor
```

```
        args:
        - daemon
        - stop
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      range: 1-5
      type: INT
  script:
    embeddedFiles:
    - name: runData
      filename: run-data.yaml
      type: TEXT
      data: |
        frame: {{Task.Param.Frame}}
    actions:
      onRun:
        command: MayaAdaptor
        args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

Pipelining des actions de session

Le pipeline des actions de session permet à un planificateur de préattribuer plusieurs actions de session à un travailleur. Le travailleur peut ensuite exécuter ces actions de manière séquentielle, réduisant ou éliminant le temps d'inactivité entre les tâches.

Pour créer une affectation initiale, le planificateur crée une session avec une tâche, le collaborateur exécute la tâche, puis le planificateur analyse la durée de la tâche pour déterminer les futures affectations.

Pour que le planificateur soit efficace, il existe des règles relatives à la durée des tâches. Pour les tâches de moins d'une minute, le planificateur utilise un schéma de croissance basé sur la puissance de 2. Par exemple, pour une tâche d'une seconde, le planificateur affecte 2 nouvelles tâches, puis 4, puis 8. Pour les tâches de plus d'une minute, le planificateur n'assigne qu'une seule nouvelle tâche et le pipeline reste désactivé.

Pour calculer la taille du pipeline, le planificateur effectue les opérations suivantes :

- Utilise la durée moyenne des tâches par rapport aux tâches terminées
- Vise à occuper le travailleur pendant une minute
- Ne prend en compte que les tâches au cours de la même session
- Ne partage pas les données relatives à la durée entre les travailleurs

Grâce au pipeline des actions de session, les employés commencent immédiatement de nouvelles tâches et il n'y a aucun temps d'attente entre les demandes du planificateur. Il améliore également l'efficacité des travailleurs et une meilleure répartition des tâches pour les processus de longue durée.

En outre, si une nouvelle tâche plus prioritaire est disponible, le travailleur terminera toutes les tâches précédemment assignées avant la fin de sa session en cours et une nouvelle session provenant d'une tâche plus prioritaire sera attribuée.

Dépendances des étapes

Deadline Cloud prend en charge la définition des dépendances entre les étapes afin qu'une étape attende la fin d'une autre étape avant de commencer. Vous pouvez définir plusieurs interdépendances pour une étape. Une étape comportant une dépendance n'est planifiée que lorsque toutes ses dépendances sont terminées.

Si le modèle de tâche définit une dépendance circulaire, la tâche est rejetée et son statut est défini sur`CREATE_FAILED`.

Le modèle de tâche suivant crée une tâche en deux étapes. `StepB`dépend de`StepA`. `StepB`ne s'exécute qu'une fois `StepA` terminé avec succès.

Une fois le travail créé, `StepA` il est dans l'`READY`état et `StepB` est dans l'`PENDING`état. Après avoir `StepA` terminé, `StepB` passe à l'`READY`état. En cas d'`StepA`échec ou `StepA` d'annulation, `StepB` passe à l'`CANCELED`état.

Vous pouvez définir une dépendance pour plusieurs étapes. Par exemple, si `StepC` cela dépend des deux `StepA` et`StepB`, `StepC` ne démarrera pas tant que les deux autres étapes ne seront pas terminées.

Les dépendances entre les étapes sont soumises aux restrictions suivantes :

- Dépendances par étape — Une étape peut dépendre d'un maximum de 128 autres étapes.
- Consommateurs par étape — Un maximum de 32 autres étapes peut dépendre d'une seule étape.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task A Done!
- name: B
  dependencies:
  - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task B Done!
```

Modifier une tâche dans Deadline Cloud

Vous pouvez utiliser les `update` commandes suivantes AWS Command Line Interface (AWS CLI) pour modifier la configuration d'une tâche ou pour définir le statut cible d'une tâche, d'une étape ou d'une tâche :

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

Dans les exemples de `update` commandes suivants, remplacez chacune *`user input placeholder`* par vos propres informations.

Example— Demander une offre d'emploi

Toutes les tâches de la tâche passent au `READY` statut, sauf s'il existe des dépendances entre les étapes. Les étapes comportant des dépendances passent à l'une `READY` ou l'autre à `PENDING` mesure qu'elles sont restaurées.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status PENDING
```

Example— Annuler une offre d'emploi

Toutes les tâches de la tâche qui n'ont pas le statut requis `SUCCEEDED` ou qui `FAILED` sont marquées`CANCELED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example— Marquer l'échec d'une tâche

Toutes les tâches du poste dont le statut est défini `SUCCEEDED` restent inchangées. Toutes les autres tâches sont marquées`FAILED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example— Marquer un travail réussi

Toutes les tâches du poste sont transférées à l'SUCCEEDEDétat.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example— Suspendre une tâche

Les tâches du poste dans l'FAILEDétat SUCCEEDEDCANCELED, ou ne changent pas. Toutes les autres tâches sont marquéesSUSPENDED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUSPENDED
```

Example— Modifier la priorité d'une tâche

Met à jour la priorité d'une tâche dans une file d'attente pour modifier l'ordre dans lequel elle est planifiée. Les tâches les plus prioritaires sont généralement planifiées en premier.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example— Modifie le nombre de tâches échouées autorisées

Actualise le nombre maximum de tâches échouées que la tâche peut avoir avant que les tâches restantes ne soient annulées.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example— Modifie le nombre de tentatives de tâches autorisées

Actualise le nombre maximal de tentatives pour une tâche avant que celle-ci n'échoue. Une tâche qui a atteint le nombre maximum de tentatives ne peut pas être mise en attente tant que cette valeur n'est pas augmentée.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example— Archiver une tâche

Met à jour l'état du cycle de vie de la tâche surARCHIVED. Les tâches archivées ne peuvent être ni planifiées ni modifiées. Vous ne pouvez archiver qu'une tâche dont l'SUSPENDEDétat est FAILED CANCELEDSUCCEEDED,, ou.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example— Changer le nom d'une tâche

Met à jour le nom d'affichage d'une tâche. Le nom de la tâche peut comporter jusqu'à 128 caractères.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example— Modifier la description d'un poste

Met à jour la description d'une tâche. La description peut comporter jusqu'à 2 048 caractères. Pour supprimer la description existante, transmettez une chaîne vide.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example— Demander une étape

Toutes les tâches de l'étape passent à l'`READY`état, sauf s'il existe des dépendances entre les étapes. Les tâches des étapes comportant des dépendances passent à l'une `READY` ou l'autre`PENDING`, et la tâche est restaurée.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example— Annuler une étape

Toutes les tâches de l'étape qui n'ont pas le statut `SUCCEEDED` ou qui `FAILED` sont marquées`CANCELED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status CANCELED
```

Example— Marquer l'échec d'une étape

Toutes les tâches de l'étape dont le statut est `SUCCEEDED` défini restent inchangées. Toutes les autres tâches sont marquées`FAILED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example— Marquer une étape comme réussie

Toutes les tâches de l'étape sont marquéesSUCCEEDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example— Suspendre une étape

Les tâches de l'étape à l'FAILEDétat SUCCEEDEDCANCELED, ou ne changent pas. Toutes les autres tâches sont marquéesSUSPENDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example— Modifier le statut d'une tâche

Lorsque vous utilisez la commande `update-task` Deadline Cloud CLI, la tâche passe à l'état spécifié.

```
aws deadline update-task \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--task-id taskID \
```

```
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

Créez et utilisez des flottes gérées par les clients de Deadline Cloud

Lorsque vous créez un parc géré par le client (CMF), vous avez le contrôle total de votre pipeline de traitement. Vous définissez l'environnement réseau et logiciel de chaque collaborateur. Deadline Cloud fait office de référentiel et de planificateur pour vos tâches.

Un travailleur peut être une instance Amazon Elastic Compute Cloud (Amazon EC2), un travailleur travaillant dans une installation de colocation ou un travailleur sur site. Chaque collaborateur doit exécuter l'agent de travail Deadline Cloud. Tous les employés doivent avoir accès au point de terminaison du service Deadline Cloud.

Les rubriques suivantes expliquent comment créer un CMF de base à l'aide d'instances Amazon EC2.

Rubriques

Créez une flotte gérée par le client

Pour créer une flotte gérée par le client (CMF), procédez comme suit.

Deadline Cloud console

Pour utiliser la console Deadline Cloud pour créer une flotte gérée par le client

1. Ouvrez la console Deadline Cloud.

2. Sélectionnez Fermes. La liste des fermes disponibles s'affiche.
3. Sélectionnez le nom de la ferme dans laquelle vous souhaitez travailler.
4. Sélectionnez l'onglet Flottes, puis choisissez Create fleet.
5. Entrez le nom de votre flotte.
6. (Facultatif) Entrez une description pour votre flotte.
7. Sélectionnez Géré par le client pour le type de flotte.
8. Sélectionnez l'accès aux services de votre flotte.

    a. Nous vous recommandons d'utiliser l'option Créer et utiliser un nouveau rôle de service pour chaque flotte pour un contrôle des autorisations plus précis. Cette option est sélectionnée par défaut.

    b. Vous pouvez également utiliser un rôle de service existant en sélectionnant Choisir un rôle de service.
9. Passez en revue vos sélections, puis choisissez Next.
10. Sélectionnez un système d'exploitation pour votre flotte. Tous les employés d'une flotte doivent disposer d'un système d'exploitation commun.
11. Sélectionnez l'architecture du processeur hôte.
12. Sélectionnez les capacités minimales et maximales du vCPU et du matériel de mémoire pour répondre aux exigences de charge de travail de vos flottes.
13. Sélectionnez un type d'Auto Scaling. Pour plus d'informations, consultez Utiliser EventBridge pour gérer les événements Auto Scaling.

    - Aucune mise à l'échelle : vous créez une flotte sur site et vous souhaitez vous désinscrire de Deadline Cloud Auto Scaling.
    - Recommandations de dimensionnement : vous êtes en train de créer une flotte Amazon Elastic Compute Cloud (Amazon EC2).
14. (Facultatif) Sélectionnez la flèche pour développer la section Ajouter des fonctionnalités.
15. (Facultatif) Cochez la case Ajouter une capacité GPU - Facultatif, puis entrez le minimum et le maximum GPUs ainsi que la mémoire.
16. Passez en revue vos sélections, puis choisissez Next.
17. (Facultatif) Définissez les capacités de travail personnalisées, puis choisissez Next.
18. À l'aide de la liste déroulante, sélectionnez une ou plusieurs files d'attente à associer à la flotte.

> **Note**
>
> Nous recommandons d'associer une flotte uniquement aux files d'attente situées toutes dans la même limite de confiance. Cette recommandation garantit une limite de sécurité solide entre l'exécution de tâches sur le même travailleur.

19. Passez en revue les associations de files d'attente, puis choisissez Next.
20. (Facultatif) Pour l'environnement de file d'attente Conda par défaut, nous créerons un environnement pour votre file d'attente qui installera les packages conda demandés par les jobs.

> **Note**
>
> L'environnement de file d'attente conda est utilisé pour installer les packages conda demandés par les jobs. En général, vous devez décocher l'environnement de file d'attente conda sur les files d'attente associées, CMFs car les commandes conda requises CMFs ne seront pas installées par défaut.

21. (Facultatif) Ajoutez des balises à votre CMF. Pour plus d'informations, consultez la section Marquage de vos AWS ressources.
22. Passez en revue la configuration de votre flotte et apportez les modifications nécessaires, puis choisissez Créer une flotte.
23. Sélectionnez l'onglet Flottes, puis notez l'ID de flotte.

AWS CLI

Pour utiliser le AWS CLI pour créer une flotte gérée par le client

1. Ouvrez un terminal .
2. Créez `fleet-trust-policy.json` dans un nouvel éditeur.
    a. Ajoutez la politique IAM suivante, en remplaçant le *`ITALICIZED`* texte par votre identifiant de AWS compte et l'identifiant de ferme Deadline Cloud.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:deadline:*:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

b. Enregistrez vos modifications.

3. Créer `fleet-policy.json`.

   a. Ajoutez la politique IAM suivante.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:AssumeFleetRoleForWorker",
                "deadline:UpdateWorker",
                "deadline:DeleteWorker",
```

```
                "deadline:UpdateWorkerSchedule",
                "deadline:BatchGetJobEntity",
                "deadline:AssumeQueueRoleForWorker"
            ],
            "Resource": "arn:aws:deadline:*:111122223333:*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:*://deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        }
    ]
}
```

b. Enregistrez vos modifications.

4. Ajoutez un rôle IAM que les employés de votre flotte pourront utiliser.

```
aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
 FleetWorkerPolicy --policy-document file://fleet-policy.json
```

5. Créer `create-fleet-request.json`.

   a. Ajoutez la politique IAM suivante, en remplaçant le texte en ITALIQUE par les valeurs de votre CMF.

> **Note**
>
> Vous pouvez le trouver *ROLE_ARN* dans le`create-cmf-fleet.json`. Pour le*OS_FAMILY*, vous devez choisir l'une des options `linux` suivantes : `macos` ou`windows`.

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
```

```
}
```

b. Enregistrez vos modifications.

6. Créez votre flotte.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

Configuration et configuration de l'hôte de travail

Un hôte de travail fait référence à une machine hôte qui exécute un serveur de travail Deadline Cloud. Cette section explique comment configurer l'hôte de travail et le configurer en fonction de vos besoins spécifiques. Chaque hôte de travail exécute un programme appelé agent de travail. L'agent des travailleurs est chargé de :

- Gérer le cycle de vie des travailleurs.
- Synchronisation du travail assigné, de son avancement et de ses résultats.
- Surveillance du travail en cours.
- Transfert des journaux vers des destinations configurées.

Nous vous recommandons d'utiliser l'agent de travail Deadline Cloud fourni. L'agent de travail est open source et nous encourageons les demandes de fonctionnalités, mais vous pouvez également les développer et les personnaliser en fonction de vos besoins.

Pour effectuer les tâches décrites dans les sections suivantes, vous avez besoin des éléments suivants :

Linux

- Une Linux instance basée sur Amazon Elastic Compute Cloud (Amazon EC2). Nous recommandons Amazon Linux 2023.
- `sudo`privilèges
- Python 3.9 ou supérieur

Windows

- Une Windows instance basée sur Amazon Elastic Compute Cloud (Amazon EC2). Nous recommandons Windows Server 2022
- Accès administrateur à l'hôte du travailleur
- Python 3.9 ou supérieur installé pour tous les utilisateurs

Création et configuration d'un environnement virtuel Python

Vous pouvez créer un environnement virtuel Python Linux si vous avez installé Python 3.9 ou supérieur et que vous l'avez placé dans votre`PATH`.

> **Note**
>
> ActivéWindows, les fichiers d'agent doivent être installés dans le répertoire global site-packages de Python. Les environnements virtuels Python ne sont actuellement pas pris en charge.

Pour créer et activer un environnement virtuel Python

1. Ouvrez un terminal en tant qu'`root`utilisateur (ou utilisez`sudo`/`su`).
2. Créez et activez un environnement virtuel Python.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Installer l'agent de travail de Deadline Cloud

Après avoir configuré votre Python et créé un environnement virtuel sur celui-ciLinux, installez les packages Python de l'agent de travail Deadline Cloud.

Pour installer les packages Python de l'agent de travail

Linux

1. Ouvrez un terminal en tant qu'`root`utilisateur (ou utilisez`sudo`/`su`).
2. Téléchargez et installez les packages d'agents de travail de Deadline Cloud depuis PyPI :

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. Ouvrez une invite de commande ou un PowerShell terminal d'administrateur.
2. Téléchargez et installez les packages d'agents de travail de Deadline Cloud depuis PyPI :

```
python -m pip install deadline-cloud-worker-agent
```

Lorsque votre hôte Windows de travail nécessite des noms de chemin longs (supérieurs à 250 caractères), vous devez activer les noms de chemin longs comme suit :

Pour activer les longs chemins pour les hôtes Windows de travail

1. Assurez-vous que la clé de registre à long chemin est activée. Pour plus d'informations, consultez la section Configuration du registre pour activer les chemins des journaux sur le site Web de Microsoft.
2. Installez le Windows SDK pour les applications de bureau C++ x86. Pour plus d'informations, consultez la section WindowsSDK dans le centre de Windows développement.
3. Ouvrez l'emplacement d'installation de Python dans votre environnement où l'agent de travail est installé. La valeur par défaut est `C:\Program Files\Python311`. Il existe un fichier exécutable nommé`pythonservice.exe`.
4. Créez un nouveau fichier appelé `pythonservice.exe.manifest` au même endroit. Ajoutez ce qui suit :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
 version="1.0.0.0"/>
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
```

```
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. Ouvrez une invite de commande et exécutez la commande suivante à l'emplacement du fichier manifeste que vous avez créé :

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

   Vous devez voir des résultats similaires à ce qui suit :

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

Le travailleur peut désormais accéder à de longs trajets. Pour nettoyer, supprimez le `pythonservice.exe.manifest` fichier et désinstallez le SDK.

Configuration de l'agent de travail Deadline Cloud

Vous pouvez configurer les paramètres de l'agent de travail de Deadline Cloud de trois manières. Nous vous recommandons d'utiliser la configuration du système d'exploitation en exécutant l'`install-deadline-worker`outil.

L'agent Worker ne prend pas en charge l'exécution en tant qu'utilisateur de domaine sous Windows. Pour exécuter une tâche en tant qu'utilisateur de domaine, vous pouvez spécifier un compte d'utilisateur de domaine lorsque vous configurez un utilisateur de file d'attente pour exécuter des tâches. Pour plus d'informations, consultez l'étape 7 de la section Files d'attente de Deadline Cloud dans le guide de l'utilisateur de AWS Deadline Cloud.

Arguments de ligne de commande — Vous pouvez spécifier des arguments lorsque vous exécutez l'agent de travail Deadline Cloud depuis la ligne de commande. Certains paramètres de configuration ne sont pas disponibles via les arguments de ligne de commande. Pour voir tous les arguments de ligne de commande disponibles, entrez`deadline-worker-agent --help`.

Variables d'environnement — Vous pouvez configurer l'agent de travail de Deadline Cloud en définissant une variable d'environnement commençant par`DEADLINE_WORKER_`. Par exemple, pour voir tous les arguments de ligne de commande disponibles, vous pouvez utiliser `export DEADLINE_WORKER_VERBOSE=true` pour définir la sortie de l'agent de travail sur détaillée. Pour plus d'exemples et d'informations, voir `/etc/amazon/deadline/worker.toml.example` sur Linux ou `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` surWindows.

Fichier de configuration : lorsque vous installez l'agent de travail, celui-ci crée un fichier de configuration situé à l'emplacement `/etc/amazon/deadline/worker.toml` activé Linux ou `C:\ProgramData\Amazon\Deadline\Config\worker.toml` activéWindows. L'agent de travail charge ce fichier de configuration au démarrage. Vous pouvez utiliser l'exemple de fichier de configuration (`/etc/amazon/deadline/worker.toml.example`activé Linux ou `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` activéWindows) pour adapter le fichier de configuration de l'agent de travail par défaut à vos besoins spécifiques.

Enfin, nous vous recommandons d'activer l'arrêt automatique de l'agent de travail une fois que votre logiciel est déployé et fonctionne comme prévu. Cela permet au parc de travailleurs d'augmenter en cas de besoin et de s'arrêter une fois le travail terminé. La mise à l'échelle automatique permet de s'assurer que vous n'utilisez que les ressources nécessaires. Pour permettre à une instance démarrée par le groupe auto scaling de s'arrêter, vous devez l'ajouter `shutdown_on_stop=true` au fichier `worker.toml` de configuration.

Pour activer l'arrêt automatique

En tant qu'**`root`**utilisateur :

- Installez l'agent de travail avec des paramètres**`--allow-shutdown`**.

  Linux

  Entrez :

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

Windows

Entrez :

```
install-deadline-worker ^
  --farm-id FARM_ID ^
  --fleet-id FLEET_ID ^
  --region REGION ^
  --allow-shutdown
```

Création de groupes et d'utilisateurs de tâches

Cette section décrit la relation utilisateur/groupe requise entre l'utilisateur agent et les utilisateurs `jobRunAsUser` définis dans vos files d'attente.

L'agent de travail de Deadline Cloud doit s'exécuter en tant qu'utilisateur dédié spécifique à l'agent sur l'hôte. Vous devez configurer la `jobRunAsUser` propriété des files d'attente de Deadline Cloud afin que les utilisateurs exécutent les tâches de file d'attente en tant qu'utilisateur et groupe de système d'exploitation spécifiques. Cette configuration signifie que vous pouvez contrôler les autorisations de système de fichiers partagés dont disposent vos tâches. Il constitue également une limite de sécurité importante entre vos tâches et l'utilisateur de l'agent de travail.

Linuxutilisateurs et groupes d'emplois

Pour configurer un utilisateur d'agent de travail local et `jobRunAsUser` vérifier que vous répondez aux exigences suivantes. Si vous utilisez un module d'authentification enfichable (PAM) Linux tel qu'Active Directory ou LDAP, votre procédure peut être différente.

L'utilisateur de l'agent de travail et le `jobRunAsUser` groupe partagé sont définis lorsque vous installez l'agent de travail. Les valeurs par défaut sont `deadline-worker-agent` et`deadline-job-users`, mais vous pouvez les modifier lorsque vous installez le Worker Agent.

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

Les commandes doivent être exécutées en tant qu'utilisateur root.

- Chacun `jobRunAsUser` doit avoir un groupe principal correspondant. La création d'un utilisateur à l'aide de la `adduser` commande crée généralement un groupe principal correspondant.

```
adduser -r -m jobRunAsUser
```

- Le groupe principal de `jobRunAsUser` est un groupe secondaire pour l'utilisateur de l'agent de travail. Le groupe partagé permet à l'agent de travail de mettre des fichiers à la disposition de la tâche pendant son exécution.

```
usermod -a -G jobRunAsUser deadline-worker-agent
```

- `jobRunAsUser`Il doit être membre du groupe de tâches partagé.

```
usermod -a -G deadline-job-users jobRunAsUser
```

- Le ne `jobRunAsUser` doit pas appartenir au groupe principal de l'utilisateur de l'agent de travail. Les fichiers sensibles écrits par l'agent de travail appartiennent au groupe principal de l'agent. Si a `jobRunAsUser` fait partie de ce groupe, les fichiers de l'agent de travail peuvent être accessibles aux tâches exécutées sur le travailleur.
- La valeur par défaut Région AWS doit correspondre à la région de la ferme à laquelle appartient le travailleur. Cela doit être appliqué à tous les `jobRunAsUser` comptes du travailleur.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

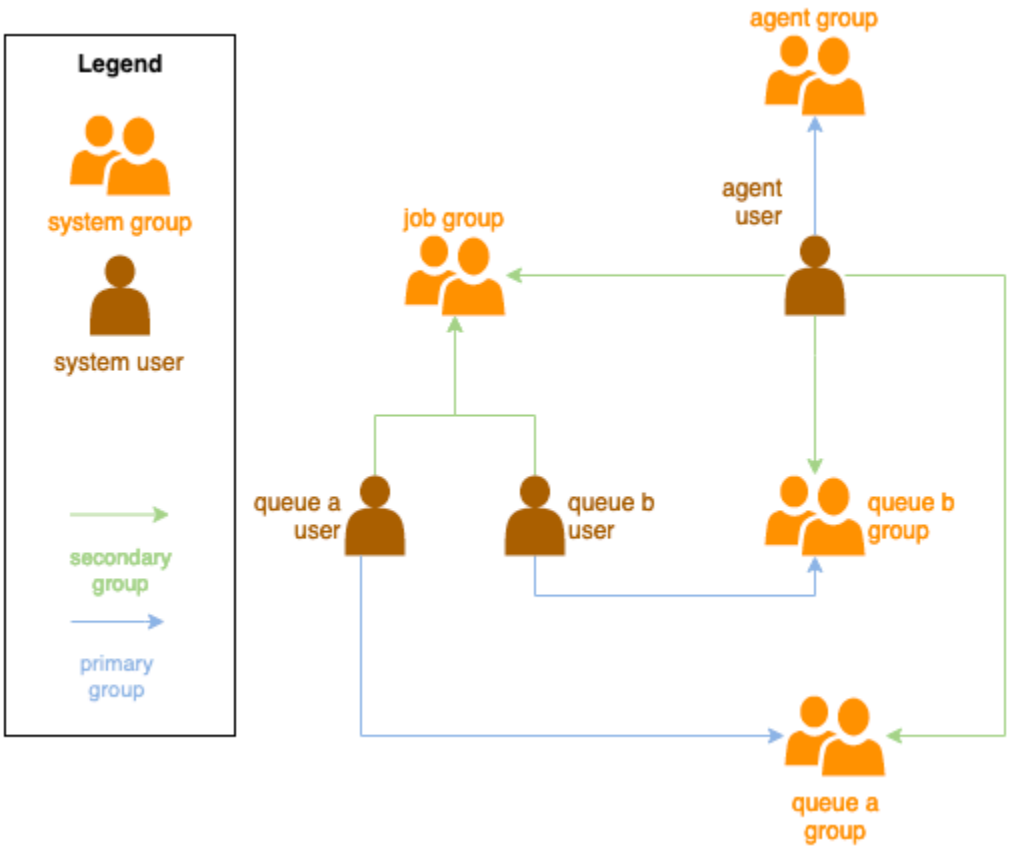
- L'utilisateur de l'agent de travail doit être capable d'exécuter `sudo` des commandes en tant que`jobRunAsUser`. Exécutez la commande suivante pour ouvrir un éditeur afin de créer une nouvelle règle sudoers :

```
visudo -f /etc/sudoers.d/deadline-worker-job-user
```

  Ajoutez ce qui suit au fichier :

```
# Allows the Deadline Cloud worker agent OS user to run commands
 # as the queue OS user without requiring a password.
deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
```

Le schéma suivant illustre la relation entre l'utilisateur de l'agent et les `jobRunAsUser` utilisateurs et groupes pour les files d'attente associées à la flotte.



Utilisateurs Windows

Pour utiliser un Windows utilisateur en tant que`jobRunAsUser`, celui-ci doit répondre aux exigences suivantes :

- Tous les `jobRunAsUser` utilisateurs de la file d'attente doivent exister.
- Leurs mots de passe doivent correspondre à la valeur du secret spécifiée dans le `JobRunAsUser` champ de leur file d'attente. Pour obtenir des instructions, reportez-vous à l'étape 7 de la section Files d'attente de Deadline Cloud dans le guide de l'utilisateur de AWS Deadline Cloud.
- L'agent-utilisateur doit être en mesure de se connecter sous le nom de ces utilisateurs.

Sécurisation de l'hôte de votre travailleur

Lorsque vous configurez votre hôte professionnel, suivez les meilleures pratiques de sécurité pour protéger les informations sensibles et maintenir des contrôles d'accès appropriés.

Configuration des autorisations relatives aux dossiers journaux

L'agent de travail écrit des fichiers journaux qui peuvent contenir des informations sensibles provenant des scripts de configuration de l'hôte et de l'exécution des tâches. La `install-deadline-worker` commande crée le répertoire des journaux avec des autorisations sécurisées. Si vous devez créer le répertoire manuellement avant l'installation, suivez les procédures suivantes pour faire correspondre les autorisations utilisées par les flottes gérées par les services :

Linux

Pour configurer les autorisations du répertoire des journaux sur Linux

1. Créez le répertoire des journaux :

```
sudo mkdir -p /var/log/amazon/deadline
```

2. Définissez le propriétaire et le groupe comme utilisateur de l'agent de travail :

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. Définissez les autorisations sur 750 :

```
sudo chmod -R 750 /var/log/amazon/deadline
```

Ces autorisations garantissent que seuls l'utilisateur et le groupe de l'agent de travail peuvent accéder aux fichiers journaux, empêchant ainsi les utilisateurs des tâches et autres utilisateurs non autorisés de lire des informations potentiellement sensibles.

Windows

Pour configurer les autorisations du répertoire des journaux sur Windows

1. Ouvrez un PowerShell terminal d'administration.
2. Créez le répertoire des journaux :

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. Configuration restreinte ACLs pour autoriser uniquement l'utilisateur de l'agent de travail et les administrateurs :

```
$acl = Get-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs"
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
$agentRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("deadline-worker",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$adminRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("Administrators",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$acl.AddAccessRule($agentRule)
$acl.AddAccessRule($adminRule)
Set-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs" $acl
```

Ces commandes limitent l'accès au répertoire des journaux uniquement à l'utilisateur de l'agent de travail et au groupe des administrateurs, empêchant ainsi les utilisateurs des tâches et autres utilisateurs non autorisés de lire des informations potentiellement sensibles.

Gérez l'accès aux secrets des utilisateurs des Windows offres d'emploi

Lorsque vous configurez une file d'attente avec un Windows`jobRunAsUser`, vous devez spécifier un secret AWS Secrets Manager. La valeur de ce secret est censée être un objet codé en JSON de la forme suivante :

```
{
    "password": "JOB_USER_PASSWORD"
}
```

Pour que les travailleurs puissent exécuter des tâches conformément à la configuration de la file d'attente`jobRunAsUser`, le rôle IAM de la flotte doit disposer des autorisations nécessaires pour obtenir la valeur du secret. Si le secret est chiffré à l'aide d'une clé KMS gérée par le client, le rôle IAM de la flotte doit également être autorisé à le déchiffrer à l'aide de la clé KMS.

Il est fortement recommandé de suivre le principe du moindre privilège pour ces secrets. Cela signifie que l'accès pour récupérer la valeur secrète du `jobRunAsUser` → `windows` → d'une file d'attente `passwordArn` doit être :

- attribué à un rôle de flotte lorsqu'une association de flotte de files d'attente est créée entre la flotte et la file d'attente
- révoqué d'un rôle de flotte lorsqu'une association de file d'attente et de flotte est supprimée entre le parc et la file d'attente

De plus, le secret du AWS Secrets Manager contenant le `jobRunAsUser` mot de passe doit être supprimé lorsqu'il n'est plus utilisé.

Autoriser l'accès à un mot de passe secret

Les flottes Deadline Cloud ont besoin d'accéder au `jobRunAsUser` mot de passe stocké dans le secret de mot de passe de la file d'attente lorsque la file d'attente et la flotte sont associées. Nous vous recommandons d'utiliser la politique de ressources de AWS Secrets Manager pour accorder l'accès aux rôles de la flotte. Si vous respectez strictement cette directive, il est plus facile de déterminer quels rôles de flotte ont accès au secret.

Pour autoriser l'accès au secret

1. Ouvrez la console AWS Secret Manager pour accéder au secret.
2. Dans la section Autorisations relatives aux ressources, ajoutez une déclaration de politique sous la forme suivante :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
```

```
}
```

Révoquer l'accès à un mot de passe secret

Lorsqu'une flotte n'a plus besoin d'accéder à une file d'attente, supprimez l'accès au mot de passe secret de la file d'attente`jobRunAsUser`. Nous vous recommandons d'utiliser la politique de ressources de AWS Secrets Manager pour accorder l'accès aux rôles de la flotte. Si vous respectez strictement cette directive, il est plus facile de déterminer quels rôles de flotte ont accès au secret.

Pour révoquer l'accès au secret

1. Ouvrez la console AWS Secret Manager pour accéder au secret.
2. Dans la section Autorisations relatives aux ressources, supprimez la déclaration de politique du formulaire :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action" : [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

Installation et configuration du logiciel requis pour les tâches

Après avoir configuré l'agent de travail Deadline Cloud, vous pouvez préparer l'hôte de travail avec tous les logiciels nécessaires à l'exécution des tâches.

Lorsque vous soumettez une tâche à une file d'attente associée`jobRunAsUser`, la tâche s'exécute sous le nom de cet utilisateur. Lorsqu'une tâche est soumise avec des commandes qui ne sont pas un chemin absolu, cette commande doit être trouvée dans le fichier PATH de cet utilisateur.

Sous Linux, vous pouvez spécifier le PATH pour un utilisateur dans l'une des options suivantes :

- leur `~/.bashrc` ou `~/.bash_profile`
- fichiers de configuration système tels que `/etc/profile.d/*` et `/etc/profile`
- scripts de démarrage du shell :`/etc/bashrc`.

Sous Windows, vous pouvez spécifier le PATH pour un utilisateur dans l'une des options suivantes :

- leurs variables d'environnement spécifiques à l'utilisateur
- les variables d'environnement à l'échelle du système

Installation d'adaptateurs d'outils de création de contenu numérique

Deadline Cloud fournit des OpenJobDescription adaptateurs pour utiliser les applications populaires de création de contenu numérique (DCC). Pour utiliser ces adaptateurs dans un parc géré par le client, vous devez installer le logiciel DCC et les adaptateurs d'application. Assurez-vous ensuite que les programmes exécutables du logiciel sont disponibles sur le chemin de recherche du système (par exemple, dans la variable d'PATHenvironnement).

Pour installer des adaptateurs DCC sur un parc géré par le client

1. Ouvrez le terminal A.
    a. Sous Linux, ouvrez un terminal en tant qu'`root`utilisateur (ou utilisez`sudo`/`su`)
    b. ActivéWindows, ouvrez une invite de commande ou un PowerShell terminal d'administrateur.
2. Installez les packages d'adaptateurs Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

Configuration des AWS informations d'identification

La phase initiale du cycle de vie des travailleurs est le démarrage. Au cours de cette phase, le logiciel d'agent des travailleurs crée un travailleur dans votre flotte et obtient les AWS informations d'identification du rôle de votre flotte pour une exploitation ultérieure.

AWS credentials for Amazon EC2

Pour créer un rôle IAM pour Amazon EC2 avec les autorisations d'hôte de Deadline Cloud

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles dans le volet de navigation, puis choisissez Créer un rôle.
3. Sélectionnez le AWS service.
4. Sélectionnez EC2 comme service ou cas d'utilisation, puis sélectionnez Suivant.
5. Pour accorder les autorisations nécessaires, joignez la politique `AWSDeadlineCloud-WorkerHost` AWS gérée.

On-premises AWS credentials

Vos employés sur site utilisent des informations d'identification pour accéder à Deadline Cloud. Pour un accès plus sécurisé, nous vous recommandons d'utiliser IAM Roles Anywhere pour authentifier vos employés. Pour plus d'informations, consultez IAM Roles Anywhere.

Pour les tests, vous pouvez utiliser les clés d'accès utilisateur IAM pour les AWS informations d'identification. Nous vous recommandons de définir une date d'expiration pour l'utilisateur IAM en incluant une politique intégrée restrictive.

> **Important**
>
> Tenez compte des avertissements suivants :
>
> - N'utilisez PAS les informations d'identification root de votre compte pour accéder aux AWS ressources. Ces informations d'identification offrent un accès illimité au compte et sont difficiles à révoquer.
> - N'incluez PAS de clés d'accès littérales ou d'informations d'identification dans vos fichiers d'application. Vous risqueriez en effet d'exposer accidentellement vos

informations d'identification si, par exemple, vous chargiez le projet sur un référentiel public.

- N'incluez PAS de fichiers contenant des informations d'identification dans votre zone de projet.
- Sécurisez vos clés d'accès. Ne communiquez pas vos clés d'accès à des tiers non autorisés, même pour vous aider à trouver les identifiants de votre compte. Ce faisant, vous pouvez donner à quelqu'un un accès permanent à votre compte.
- Sachez que toutes les informations d'identification stockées dans le fichier AWS d'informations d'identification partagé sont stockées en texte brut.

Pour plus de détails, consultez la section Meilleures pratiques en matière de gestion des clés AWS d'accès dans le manuel de référence AWS général.

Créer un utilisateur IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Utilisateurs, puis sélectionnez Créer un utilisateur.
3. Nommez l'utilisateur. Décochez la case Fournir un accès utilisateur au AWS Management Console, puis choisissez Next.
4. Choisissez Joindre directement les politiques.
5. Dans la liste des politiques d'autorisation, choisissez la AWSDeadlineCloud-WorkerHostpolitique, puis cliquez sur Suivant.
6. Vérifiez les informations de l'utilisateur, puis choisissez Créer un utilisateur.

Restreindre l'accès utilisateur à une fenêtre de temps limitée

Toutes les clés d'accès utilisateur IAM que vous créez sont des informations d'identification à long terme. Pour garantir que ces informations d'identification expirent en cas de mauvaise gestion, vous pouvez fixer une durée limitée en créant une politique en ligne qui spécifie une date après laquelle les clés ne seront plus valides.

1. Ouvrez l'utilisateur IAM que vous venez de créer. Dans l'onglet Autorisations, choisissez Ajouter des autorisations, puis choisissez Créer une politique intégrée.

2. Dans l’éditeur JSON, spécifiez les autorisations suivantes. Pour utiliser cette politique, remplacez la valeur d'`aws:CurrentTime`horodatage dans l'exemple de politique par votre propre heure et date.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Créer une clé d’accès

1. Sur la page des détails de l'utilisateur, sélectionnez l'onglet Informations d'identification de sécurité. Dans la section Clés d’accès, choisissez Créer une clé d’accès.
2. Indiquez que vous souhaitez utiliser la clé pour Autre, puis cliquez sur Suivant, puis sur Créer une clé d'accès.
3. Sur la page Récupérer les clés d'accès, choisissez Afficher pour révéler la valeur de la clé d'accès secrète de votre utilisateur. Vous pouvez copier les informations d’identification ou télécharger un fichier .csv.

Stocker les clés d'accès des utilisateurs

- Stockez les clés d'accès utilisateur dans le fichier d'informations d' AWS identification de l'utilisateur agent sur le système hôte du poste de travail :
  - LinuxActivé, le fichier se trouve dans `~/.aws/credentials`

- WindowsActivé, le fichier se trouve dans `%USERPROFILE\.aws\credentials`

Remplacez les clés suivantes :

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> Lorsque vous n'aurez plus besoin de cet utilisateur IAM, nous vous recommandons de le supprimer afin de respecter les meilleures pratiques en AWS matière de sécurité. Nous vous recommandons de demander à vos utilisateurs humains d'utiliser des informations d'identification temporaires AWS IAM Identity Centerlors de l'accès AWS.

Flux de données hôte par le personnel pour les flottes gérées par le client

Cette rubrique décrit les connexions réseau que les hôtes de AWS Deadline Cloud (Deadline Cloud) établissent en cours de fonctionnement, y compris les points de terminaison contactés, les protocoles utilisés et les données transmises. Ces informations s'appliquent aux employés du parc géré par le client (CMF), y compris les instances Amazon Elastic Compute Cloud (Amazon EC2) et les employés sur site. Utilisez ces informations pour configurer les règles de pare-feu, créer des points de terminaison VPC, effectuer des audits de sécurité ou planifier des politiques réseau pour vos hôtes de travail. Pour plus d'informations sur le réseau de flotte géré par des services, consultez. Inter-network confidentialité du trafic

Toutes les communications avec les employés sont uniquement sortantes. Les hôtes de travail initient toutes les connexions ; il n'est pas nécessaire d'autoriser les connexions entrantes. Toutes les connexions utilisent le protocole HTTPS (TLS 1.2 ou version ultérieure) sur le port 443.

Cette rubrique inclut les sections suivantes :

- the section called “Points de terminaison et protocoles”
- the section called “Opérations d'API utilisées par les travailleurs”

- the section called “Autres données transmises”
- the section called “Options de connectivité privées”

Points de terminaison et protocoles

Le tableau suivant répertorie les points de terminaison AWS de service auxquels les hôtes de travail se connectent en cours de fonctionnement. Pour obtenir la liste complète des points de terminaison régionaux pour chaque service, consultez les points de terminaison et quotas du service dans la référence AWS générale.

Référence du point de terminaison hôte du travailleur

AWS service	Endpoint	Port/Protocole	Objectif	Obligatoire
Deadline Cloud (planification)	`scheduling.deadline. [Region].amazonaws.com`	443//HTTPS	Enregistrement des travailleurs, interrogation des tâches, mises à jour du statut, échange d'informations d'identification, récupération des entités de travail. Consultez the section called “Opérations d'API utilisées par les travailleurs”.	Toujours
Amazon CloudWatch Logs (CloudWatch journaux)	`logs.[Region].amazona s.com`	443//HTTPS	Livraison de l'agent de travail et du journal de session.	Toujours
Amazon Simple Storage Service	`s3.[Region].amazonaws.com`	443//HTTPS	Chargement et téléchargement des pièces jointes au job.	Si vous utilisez des pièces jointes

AWS service	Endpoint	Port/Protocole	Objectif	Obligatoire
(Amazon S3)				

Si vos tâches utilisent d'autres AWS services, vous devrez peut-être également autoriser les connexions sortantes vers ces points de terminaison de service.

Opérations d'API utilisées par les travailleurs

Toutes les opérations d'API suivantes utilisent le `scheduling.deadline.[Region].amazonaws.com` point de terminaison. Pour connaître les schémas complets de demande et de réponse de chaque opération, consultez le document de référence de l'API Deadline Cloud.

Phase Bootstrap

Lorsqu'un hôte de travailleurs démarre, l'agent de travail s'enregistre auprès de la flotte. Les informations d'identification bootstrap nécessitent les autorisations de la politique `AWSDeadlineCloud-WorkerHost` AWS gérée, ou des autorisations personnalisées équivalentes. La phase de démarrage utilise les opérations d'API suivantes :

- `CreateWorker`— Inscrit le travailleur dans la flotte. Envoie le nom d'hôte et les adresses IP. Reçoit un identifiant de travailleur.
- `AssumeFleetRoleForWorker`— Obtient les informations d'identification relatives aux rôles de la flotte. Reçoit AWS les informations d'identification temporaires que l'agent de travail utilise pour les opérations suivantes.

Phase opérationnelle

Après le bootstrap, l'agent de travail interroge les sessions de travail et traite les sessions. Le rôle de flotte nécessite les autorisations de la politique `AWSDeadlineCloud-FleetWorker` AWS gérée, ou des autorisations personnalisées équivalentes, et utilise les opérations d'API suivantes :

- `UpdateWorker`— Met à jour le statut du travailleur, par exemple `STOPPED` pendant l'arrêt.
- `UpdateWorkerSchedule`— Sondages pour les missions de travail. Envoie des mises à jour sur l'état des actions de session, y compris l'état d'achèvement, le pourcentage de progression, le

message de progression et les hachages du manifeste de sortie. Reçoit les sessions assignées (ID de tâche, ID de file d'attente, actions de session, configuration du journal), les demandes d'annulation, le statut du travailleur souhaité et l'intervalle de mise à jour.

- `BatchGetJobEntity`— Récupère les détails de la tâche assignée. Envoie les identifiants des entités de travail. Reçoit les détails de la tâche, les détails de l'environnement et les détails des pièces jointes à la tâche.
- `AssumeFleetRoleForWorker`— Actualise régulièrement les informations d'identification des rôles de la flotte.
- `AssumeQueueRoleForWorker`— Obtient les informations d'identification du rôle de file d'attente associées à une file d'attente spécifique. Le travailleur utilise ces informations d'identification pour accéder aux pièces jointes aux tâches dans Amazon S3.

Autres données transmises

Outre les opérations de l'API de planification de Deadline Cloud, les hôtes de travail transmettent les données suivantes à d'autres AWS services :

Données du journal

L'agent de travail envoie les journaux de l'agent de travail et les journaux de session (stdout et stderr issus des processus de travail) à CloudWatch Logs à l'aide de l'opération API. `PutLogEvents`

Pièces jointes aux offres d'emploi

Les travailleurs transfèrent les fichiers d'entrée et de sortie via Amazon S3 à l'aide `GetObject` d'opérations `PutObject` d'API. Le travailleur utilise les informations d'identification du rôle de file d'attente obtenues par `AssumeQueueRoleForWorker` le biais de cet accès.

Télémétrie (facultatif)

L'agent de travail envoie des mesures opérationnelles telles que des rapports d'accident. Vous pouvez désactiver la collecte de données télémétriques. Pour de plus amples informations, veuillez consulter Refus.

Options de connectivité privées

Vous pouvez l'utiliser AWS PrivateLink pour maintenir le trafic entre les hôtes de travail CMF et Deadline Cloud au sein de votre VPC, sans passer par l'Internet public. Pour les employés sur

site, vous pouvez combiner AWS PrivateLink avec AWS Direct Connect (Direct Connect) ou une connexion VPN. Pour de plus amples informations, veuillez consulter Accès AWS Deadline Cloud en utilisant un point de terminaison d'interface (AWS PrivateLink).

Testez la configuration de votre hôte de travail

Après avoir installé l'agent de travail, installé le logiciel nécessaire au traitement de vos tâches et configuré les AWS informations d'identification de l'agent de travail, vous devez vérifier que l'installation peut traiter vos tâches avant de créer un AMI pour votre flotte. Vous devez tester les éléments suivants :

- L'agent de travail Deadline Cloud est correctement configuré pour s'exécuter en tant que service système.
- Que le travailleur interroge la file d'attente associée pour le travail.
- Que le travailleur traite avec succès les tâches envoyées à la file d'attente associée au parc.

Une fois que vous avez testé la configuration et que vous avez réussi à traiter des tâches représentatives, vous pouvez utiliser le travailleur configuré pour créer un AMI pour les EC2 employés d'Amazon, ou comme modèle pour vos employés sur site.

> **Note**
>
> Si vous testez la configuration de l'hôte de travail d'une flotte à dimensionnement automatique, vous pouvez avoir des difficultés à tester votre travailleur dans les situations suivantes :
>
> - S'il n'y a aucun travail dans la file d'attente, Deadline Cloud arrête l'agent de travail peu de temps après le démarrage du travailleur.
> - Si l'agent de travail est configuré pour arrêter l'hôte lors de l'arrêt, il arrête la machine s'il n'y a aucun travail dans la file d'attente.
>
> Pour éviter ces problèmes, utilisez une flotte intermédiaire qui ne s'adapte pas automatiquement pour configurer et tester vos employés. Après avoir testé l'hôte du travailleur, assurez-vous de définir le bon identifiant de flotte avant de préparer un AMI.

Pour tester la configuration de votre hôte de travail

1. Exécutez l'agent de travail en démarrant le service du système d'exploitation.

   Linux

   À partir d'un shell root, exécutez la commande suivante :

   ```
   systemctl start deadline-worker
   ```

   Windows

   À partir d'une invite de commande de l'administrateur ou PowerShell terminal, entrez la commande suivante :

   ```
   sc.exe start DeadlineWorker
   ```

2. Surveillez le travailleur pour vous assurer qu'il démarre et interrogez-le pour obtenir du travail.

   Linux

   À partir d'un shell root, exécutez la commande suivante :

   ```
   systemctl status deadline-worker
   ```

   La commande doit renvoyer une réponse du type :

   ```
   Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
   ```

   Si la réponse ne ressemble pas à cela, inspectez le fichier journal à l'aide de la commande suivante :

   ```
   tail -n 25 /var/log/amazon/deadline/worker-agent.log
   ```

   Windows

   À partir d'une invite de commande de l'administrateur ou PowerShell terminal, entrez la commande suivante :

```
sc.exe query DeadlineWorker
```

La commande doit renvoyer une réponse du type :

```
STATE   : 4 RUNNING
```

Si la réponse n'en contient pas`RUNNING`, inspectez le fichier journal du travail. Ouvert et administrateur PowerShell demandez et exécutez la commande suivante :

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. Soumettez les tâches à la file d'attente associée à votre flotte. Les tâches doivent être représentatives des tâches traitées par la flotte.
4. Surveillez la progression de la tâche à l'aide du moniteur ou de la CLI de Deadline Cloud. En cas d'échec d'une tâche, consultez les journaux de session et de travail.
5. Mettez à jour la configuration de l'hôte de travail selon les besoins jusqu'à ce que les tâches soient terminées avec succès.
6. Lorsque les tâches de test sont réussies, vous pouvez arrêter le travailleur :

   Linux

   À partir d'un shell root, exécutez la commande suivante :

   ```
   systemctl stop deadline-worker
   ```

   Windows

   À partir d'une invite de commande de l'administrateur ou PowerShell terminal, entrez la commande suivante :

   ```
   sc.exe stop DeadlineWorker
   ```

Créez un Amazon Machine Image

Pour créer un Amazon Machine Image (AMI) à utiliser dans une flotte gérée par le client (CMF EC2) Amazon Elastic Compute Cloud (Amazon), effectuez les tâches décrites dans cette section. Vous devez créer une EC2 instance Amazon avant de continuer. Pour plus d'informations, consultez Lancer votre instance dans le Guide de EC2 l'utilisateur Amazon pour les instances Linux.

> **Important**
>
> Création d'un AMI crée un instantané des volumes attachés à l' EC2 instance Amazon. Tous les logiciels installés sur l'instance sont conservés, de sorte que les instances, qui sont réutilisées lorsque vous lancez des instances depuis AMI. Nous vous recommandons d'adopter une stratégie de correction et de mettre régulièrement à jour les nouvelles AMI avec un logiciel mis à jour avant de l'appliquer à votre flotte.

Préparez l' EC2 instance Amazon

Avant de construire un AMI, vous devez supprimer l'état du travailleur. L'état du travailleur persiste entre les lancements de l'agent de travail. Si cet état persiste sur AMI, alors toutes les instances lancées à partir de celui-ci partageront le même état.

Nous vous recommandons également de supprimer tous les fichiers journaux existants. Les fichiers journaux peuvent rester sur une EC2 instance Amazon lorsque vous préparez l'AMI. La suppression de ces fichiers permet de réduire la confusion lors du diagnostic d'un éventuel problème dans les flottes de travailleurs qui utilisent l'AMI.

Vous devez également activer le service système d'agent de travail afin que l'agent de travail Deadline Cloud EC2 soit lancé au démarrage d'Amazon.

Enfin, nous vous recommandons d'activer l'arrêt automatique de l'agent de travail. Cela permet au parc de travailleurs d'augmenter en cas de besoin et de s'arrêter une fois le travail de rendu terminé. Cette mise à l'échelle automatique permet de s'assurer que vous n'utilisez les ressources que lorsque vous en avez besoin.

Pour préparer l' EC2 instance Amazon

1. Ouvrez la EC2 console Amazon.
2. Lancez une EC2 instance Amazon. Pour plus d'informations, consultez Lancer votre instance.

3. Configurez l'hôte pour qu'il se connecte à votre fournisseur d'identité (IdP), puis montez le système de fichiers partagé dont il a besoin.
4. Suivez les didacticiels pourInstaller l'agent de travail de Deadline Cloud, puisConfiguration de l'agent de travail, etCréation de groupes et d'utilisateurs de tâches.
5. Si vous préparez un AMI basé sur Amazon Linux 2023 pour exécuter un logiciel compatible avec la plate-forme de référence VFX, vous devez mettre à jour plusieurs exigences. Pour plus d'informations, consultez la section Compatibilité de la plate-forme de référence VFX dans le guide de l'utilisateur de AWS Deadline Cloud.
6. Ouvrez un terminal .
   a. Sous Linux, ouvrez un terminal en tant qu'`root`utilisateur (ou utilisez`sudo`/`su`)
   b. Activé Windows, ouvrez une invite de commande ou un PowerShell terminal d'administrateur.
7. Assurez-vous que le service de travail n'est pas en cours d'exécution et qu'il est configuré pour démarrer au démarrage :
   a. Sous Linux, exécutez

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. Activé Windows, courez

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. Supprimez l'état du travailleur.
   a. Sous Linux, exécutez

```
rm -rf /var/lib/deadline/*
```

   b. Activé Windows, courez

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Supprimez les fichiers journaux.
   a. Sous Linux, exécutez

```
rm -rf /var/log/amazon/deadline/*
```

b. Activé Windows, courez

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. Activé Windows, il est recommandé d'exécuter l'application Amazon EC2 Launch Settings située dans le menu Démarrer pour terminer la préparation finale de l'hôte et arrêter l'instance.

> **Note**
>
> Vous DEVEZ choisir Shutdown without Sysprep et ne jamais choisir Shutdown with Sysprep. L'arrêt de Sysprep rendra tous les utilisateurs locaux inutilisables. Pour plus d'informations, consultez la section Avant de commencer de la rubrique Création d'une AMI personnalisée du Guide de l'utilisateur pour les instances Windows.

Construisez le AMI

Pour construire le AMI

1. Ouvrez la EC2 console Amazon.
2. Sélectionnez Instances dans le volet de navigation, puis sélectionnez votre instance.
3. Choisissez État de l'instance, puis Arrêter l'instance.
4. Une fois l'instance arrêtée, choisissez Actions.
5. Choisissez Image et modèles, puis Créer une image.
6. Entrez le nom de l'image.
7. (Facultatif) Entrez une description pour votre image.
8. Choisissez Create image (Créer une image).

Créez une infrastructure de flotte avec un groupe Amazon EC2 Auto Scaling

Cette section explique comment créer une flotte Amazon EC2 Auto Scaling.

Utilisez le modèle CloudFormation YAML ci-dessous pour créer un groupe Amazon EC2 Auto Scaling (Auto Scaling), un Amazon Virtual Private Cloud (Amazon VPC) avec deux sous-réseaux, un profil d'instance et un rôle d'accès à l'instance. Ils sont nécessaires pour lancer une instance à l'aide d'Auto Scaling dans les sous-réseaux.

Vous devez revoir et mettre à jour la liste des types d'instances en fonction de vos besoins de rendu.

Pour une explication complète des ressources et des paramètres utilisés dans le modèle CloudFormation YAML, consultez la référence au type de ressource Deadline Cloud dans le guide de l'AWS CloudFormation utilisateur.

Pour créer une flotte Amazon EC2 Auto Scaling

1. Utilisez l'exemple suivant pour créer un CloudFormation modèle qui définit les `AMIId` paramètres `FarmIDFleetID`, et. Enregistrez le modèle dans un `.YAML` fichier sur votre ordinateur local.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
```

```
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
    DependsOn:
      - deadlineIGW
      - deadlineVPCGatewayAttachment
  deadlinePublicSubnet0:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.16.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - a
  deadlineSubnetRouteTableAssociation0:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
  deadlinePublicSubnet1:
```

```
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.20.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - c
  deadlineSubnetRouteTableAssociation1:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet1
  deadlineInstanceAccessAccessRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
        - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
```

```
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
        MetadataOptions:
          HttpTokens: required
          HttpEndpoint: enabled

  deadlineAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      AutoScalingGroupName: !Join
        - ''
        - - deadline-ASG-autoscalable-
          - !Ref FleetId
      MinSize: 0
      MaxSize: 10
      VPCZoneIdentifier:
        - !Ref deadlinePublicSubnet0
        - !Ref deadlinePublicSubnet1
      NewInstancesProtectedFromScaleIn: true
      MixedInstancesPolicy:
        InstancesDistribution:
          OnDemandBaseCapacity: 0
          OnDemandPercentageAboveBaseCapacity: 0
          SpotAllocationStrategy: capacity-optimized
          OnDemandAllocationStrategy: lowest-price
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref deadlineLaunchTemplate
            Version: !GetAtt
              - deadlineLaunchTemplate
              - LatestVersionNumber
```

```
        Overrides:
          - InstanceType: m5.large
          - InstanceType: m5d.large
          - InstanceType: m5a.large
          - InstanceType: m5ad.large
          - InstanceType: m5n.large
          - InstanceType: m5dn.large
          - InstanceType: m4.large
          - InstanceType: m3.large
          - InstanceType: r5.large
          - InstanceType: r5d.large
          - InstanceType: r5a.large
          - InstanceType: r5ad.large
          - InstanceType: r5n.large
          - InstanceType: r5dn.large
          - InstanceType: r4.large
    MetricsCollection:
      - Granularity: 1Minute
        Metrics:
          - GroupMinSize
          - GroupMaxSize
          - GroupDesiredCapacity
          - GroupInServiceInstances
          - GroupTotalInstances
          - GroupInServiceCapacity
          - GroupTotalCapacity
```

2. Ouvrez la CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation.

   Utilisez la CloudFormation console pour créer une pile en suivant les instructions de téléchargement du fichier modèle que vous avez créé. Pour plus d'informations, consultez la section Création d'une pile sur la CloudFormation console dans le Guide de AWS CloudFormation l'utilisateur.

> **Note**
>
> - Les informations d'identification du rôle IAM associées à l'instance Amazon EC2 de votre travailleur sont disponibles pour tous les processus exécutés sur ce travailleur, y compris les tâches. Le travailleur doit avoir le moins de privilèges pour opérer : `deadline:CreateWorker` et `deadline:AssumeFleetRoleForWorker`.

- L'agent de travail obtient les informations d'identification pour le rôle de file d'attente et les configure pour les utiliser lors de l'exécution de tâches. Le rôle de profil d'instance Amazon EC2 ne doit pas inclure les autorisations nécessaires à vos tâches.

Faites évoluer automatiquement votre flotte Amazon EC2 grâce à la fonction de recommandation de dimensionnement de Deadline Cloud

Deadline Cloud s'appuie sur un groupe Amazon EC2 Auto Scaling (Auto Scaling) pour dimensionner automatiquement le parc géré par le client (CMF) Amazon EC2. Vous devez configurer le mode flotte et déployer l'infrastructure requise dans votre compte pour que votre flotte évolue automatiquement. L'infrastructure que vous avez déployée fonctionnera pour toutes les flottes, vous ne devez donc la configurer qu'une seule fois.

Le flux de travail de base est le suivant : vous configurez le mode flotte pour qu'il évolue automatiquement, puis Deadline Cloud envoie un EventBridge événement pour cette flotte chaque fois que la taille recommandée change (un événement contient l'identifiant de la flotte, la taille de flotte recommandée et d'autres métadonnées). Vous aurez une EventBridge règle pour filtrer les événements pertinents et disposerez d'un Lambda pour les consommer. Le Lambda s'intégrera à Amazon EC2 Auto Scaling `AutoScalingGroup` pour dimensionner automatiquement le parc Amazon EC2.

Réglez le mode flotte sur **`EVENT_BASED_AUTO_SCALING`**

Configurez votre mode flotte pour`EVENT_BASED_AUTO_SCALING`. Pour ce faire, vous pouvez utiliser la console ou utiliser le AWS CLI pour appeler directement l'`UpdateFleet`API `CreateFleet` or. Une fois le mode configuré, Deadline Cloud commence à envoyer EventBridge des événements chaque fois que la taille de flotte recommandée change.

- Exemple de `UpdateFleet` commande :

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- Exemple de `CreateFleet` commande :

```
aws deadline create-fleet \
```

```
    --farm-id FARM_ID \
    --display-name "Fleet name" \
    --max-worker-count 10 \
    --configuration file://configuration.json
```

Voici un exemple d'`configuration.json`utilisation dans les commandes CLI ci-dessus (`--configuration file://configuration.json`).

- Pour activer Auto Scaling sur votre flotte, vous devez régler le mode sur`EVENT_BASED_AUTO_SCALING`.
- `workerCapabilities`Il s'agit des valeurs par défaut attribuées au CMF lorsque vous l'avez créé. Vous pouvez modifier ces valeurs si vous avez besoin d'augmenter les ressources disponibles pour votre CMF.

Une fois que vous avez configuré le mode flotte, Deadline Cloud commence à émettre des événements de recommandation de taille de flotte pour cette flotte.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

Déployez la pile Auto Scaling à l'aide du CloudFormation modèle

Vous pouvez configurer une EventBridge règle pour filtrer les événements, un Lambda pour consommer les événements et contrôler Auto Scaling, et une file d'attente SQS pour stocker les événements non traités. Utilisez le CloudFormation modèle suivant pour déployer tous les éléments

d'une pile. Une fois les ressources déployées avec succès, vous pouvez soumettre une tâche et la flotte augmentera automatiquement.

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
              "source": "aws.deadline",
              "account": "111122223333",
              "time": "2017-12-22T18:43:48Z",
              "region": "us-west-1",
              "resources": [],
              "detail": {
                  "farmId": "farm-12345678900000000000000000000000",
                  "fleetId": "fleet-12345678900000000000000000000000",
                  "oldFleetSize": 1,
                  "newFleetSize": 5,
              }
          }
          """

          import json
          import boto3
          import logging

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          auto_scaling_client = boto3.client("autoscaling")
```

```
          def lambda_handler(event, context):
              logger.info(event)
              event_detail = event["detail"]
              fleet_id = event_detail["fleetId"]
              desired_capacity = event_detail["newFleetSize"]

              asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
              auto_scaling_client.set_desired_capacity(
                  AutoScalingGroupName=asg_name,
                  DesiredCapacity=desired_capacity,
                  HonorCooldown=False,
              )

              return {
                  'statusCode': 200,
                  'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
              }
      Handler: index.lambda_handler
      Role: !GetAtt
        - AutoScalingLambdaServiceRole
        - Arn
      Runtime: python3.11
    DependsOn:
      - AutoScalingLambdaServiceRoleDefaultPolicy
      - AutoScalingLambdaServiceRole
  AutoScalingEventRule:
    Type: 'AWS::Events::Rule'
    Properties:
      EventPattern:
        source:
          - aws.deadline
        detail-type:
          - Fleet Size Recommendation Change
      State: ENABLED
      Targets:
        - Arn: !GetAtt
            - AutoScalingLambda
            - Arn
          DeadLetterConfig:
            Arn: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
          Id: Target0
```

```
        RetryPolicy:
          MaximumRetryAttempts: 15
  AutoScalingEventRuleTargetPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !GetAtt
        - AutoScalingLambda
        - Arn
      Principal: events.amazonaws.com
      SourceArn: !GetAtt
        - AutoScalingEventRule
        - Arn
  AutoScalingLambdaServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - !Join
          - ''
          - - 'arn:'
            - !Ref 'AWS::Partition'
            - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  AutoScalingLambdaServiceRoleDefaultPolicy:
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
      Roles:
        - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
```

```
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

Procéder à un bilan de santé de la flotte

Après avoir créé votre flotte, vous devez établir un bilan de santé personnalisé pour vous assurer que votre flotte reste saine et exempte d'instances bloquées afin d'éviter des coûts inutiles. Consultez la section Déploiement d'un bilan de santé de la flotte Deadline Cloud GitHub. Un bilan de santé peut réduire le risque qu'une modification accidentelle de votre configurationAmazon Machine Image, de votre modèle de lancement ou de votre réseau passe inaperçue.

Configuration et utilisation des flottes gérées par le service Deadline Cloud

Un parc géré par des services (SMF) est un ensemble de travailleurs géré par Deadline Cloud. Un SMF élimine le besoin de gérer le dimensionnement du parc pour traiter les demandes ou de réduire la taille du parc une fois les tâches terminées.

Lorsqu'un SMF est associé à une file d'attente à l'aide de l'environnement de file d'attente conda par défaut, Deadline Cloud configure les travailleurs de la flotte avec le progiciel approprié. Pour les applications partenaires prises en charge, consultez la section Environnement de file d'attente Conda par défaut dans le guide de l'utilisateur de AWS Deadline Cloud.

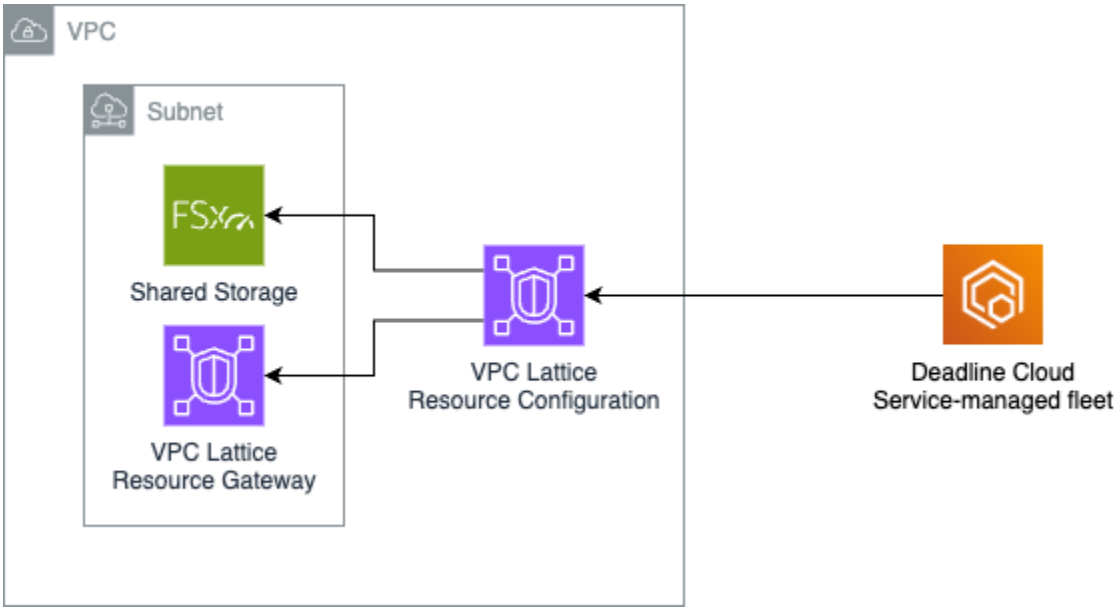
Dans la plupart des cas, il n'est pas nécessaire de changer de SMF pour traiter vos charges de travail. Toutefois, certaines situations peuvent nécessiter que vous apportiez des modifications à vos flottes.

> **Note**
>
> Pour installer un logiciel personnalisé sur les travailleurs à l'aide de scripts de configuration de l'hôte, voirExécuter des scripts de configuration de l'hôte avec des privilèges d'administrateur.

Rubriques

Connectez les ressources VPC à votre SMF avec des points de terminaison de ressources VPC

Avec les points de terminaison de ressources Amazon VPC pour les flottes gérées par les services (SMF) de Deadline Cloud, vous pouvez connecter vos ressources VPC telles que les systèmes de fichiers réseau (NFS), les serveurs de licences et les bases de données à vos employés de Deadline Cloud. Cette fonctionnalité vous permet de tirer parti de la plateforme entièrement gérée de Deadline Cloud tout en l'intégrant à votre infrastructure existante au sein d'un VPC.



> **Tip**
>
> Pour un CloudFormation modèle de référence qui configure un FSx cluster Amazon et le connecte à un parc géré par des services, consultez smf_vpc_fsx dans le référentiel d'exemples de Deadline Cloud sur. GitHub

Comment fonctionnent les points de terminaison des ressources VPC

Les points de terminaison des ressources VPC utilisent VPC Lattice pour créer une connexion sécurisée entre vos employés SMF de Deadline Cloud et les ressources de votre VPC. La connexion est unidirectionnelle, ce qui signifie que les travailleurs peuvent établir une connexion aux ressources de votre VPC et transférer des données dans les deux sens, mais les ressources de votre VPC ne peuvent pas établir de connexion avec un travailleur.

Lorsque vous connectez une ressource VPC à un parc géré par le service Deadline Cloud, vos employés peuvent accéder aux ressources de votre VPC à l'aide d'un nom de domaine privé. En outre, le trafic entre les travailleurs et les ressources de votre VPC passe par le biais de VPC Lattice, et les ressources de votre VPC voient le trafic provenant de la passerelle de ressources VPC Lattice.

Pour en savoir plus, consultez le guide de l'utilisateur du VPC Lattice.

Conditions préalables

Avant de connecter les ressources VPC à votre flotte gérée par le service Deadline Cloud, assurez-vous de disposer des éléments suivants :

- Un AWS compte auprès d'un VPC contenant les ressources que vous souhaitez connecter.
- Autorisations IAM pour créer et gérer les ressources VPC Lattice.
- Un parc Deadline Cloud avec au moins un parc géré par des services.
- Ressources VPC que vous souhaitez rendre accessibles (NFSFSx, serveurs de licences, etc.).

Configuration d'un point de terminaison de ressource VPC

Pour configurer un point de terminaison de ressources VPC, vous devez créer des ressources dans VPC Lattice AWS RAM, puis connecter ces ressources à votre flotte dans Deadline Cloud. Pour configurer un point de terminaison de ressource VPC pour votre SMF, procédez comme suit.

1. Pour créer une passerelle de ressources dans VPC Lattice, consultez la section Créer une passerelle de ressources dans le guide de l'utilisateur de VPC Lattice.
2. Pour créer une configuration de ressources dans VPC Lattice, consultez la section Créer une configuration de ressources dans le guide de l'utilisateur de VPC Lattice.
3. Pour partager la ressource avec votre flotte Deadline Cloud, créez un partage de ressources dans AWS RAM. Consultez la section Création d'un partage de ressources pour obtenir des instructions.

   Lors de la création d'un partage de ressources, pour Principaux, sélectionnez Service principal dans la liste déroulante, puis entrez. **`fleets.deadline.amazonaws.com`**
4. Pour connecter la configuration des ressources à votre flotte Deadline Cloud, procédez comme suit.

   a. Si ce n'est pas déjà fait, ouvrez la console Deadline Cloud.
   b. Dans le volet de navigation, choisissez Fermes, puis sélectionnez votre ferme.
   c. Cliquez sur l'onglet Flottes, puis sélectionnez votre flotte.
   d. Sélectionnez l'onglet Configurations .
   e. Sous Points de terminaison des ressources VPC, choisissez Modifier.
   f. Sélectionnez la configuration de ressource que vous avez créée, puis choisissez Enregistrer les modifications.

Accès à vos ressources VPC

Après avoir connecté votre ressource VPC à votre flotte, les employés peuvent y accéder à l'aide d'un nom de domaine privé au format suivant : `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Ce domaine est privé et uniquement accessible par les employés (pas depuis Internet ou votre poste de travail).

Pour monter ou configurer l'accès à la ressource VPC sur vos serveurs de travail, utilisez un script de configuration d'hôte. Les scripts de configuration de l'hôte s'exécutent avec des privilèges d'administrateur lorsque les utilisateurs démarrent, ce qui vous permet de monter des systèmes de fichiers, de configurer les paramètres réseau ou d'effectuer d'autres tâches de configuration.

Authentification et sécurité

Pour les ressources nécessitant une authentification, stockez les informations d'identification de manière sécurisée dans AWS Secrets Manager, accédez aux secrets issus des scripts de configuration de votre hôte ou des scripts de travail, et implémentez les autorisations de système de fichiers appropriées pour contrôler l'accès. Tenez compte des implications en matière de sécurité lorsque vous partagez des ressources entre plusieurs flottes. Par exemple, si deux flottes sont connectées au même stockage partagé, les tâches exécutées sur une flotte peuvent accéder aux actifs créés à partir de l'autre flotte.

Considérations techniques

Lorsque vous utilisez des points de terminaison de ressources VPC, tenez compte des points suivants :

- Les connexions ne peuvent être établies que depuis les travailleurs vers les ressources VPC, et non depuis les ressources VPC vers les travailleurs.
- Une fois établie, une connexion persiste jusqu'à ce qu'elle soit réinitialisée, même si la configuration des ressources est déconnectée.
- La connexion VPC Lattice gère automatiquement les connexions entre les zones de disponibilité sans frais supplémentaires. Votre passerelle de ressources doit partager une zone de disponibilité avec votre ressource VPC. Nous vous recommandons donc de configurer la passerelle de ressources pour couvrir toutes les zones de disponibilité.

- Le trafic passant par le point de terminaison de la ressource VPC utilise la traduction d'adresses réseau (NAT), qui n'est pas compatible avec tous les cas d'utilisation. Par exemple, Microsoft Active Directory (AD) ne peut pas se connecter via NAT.

Pour plus d'informations sur les quotas VPC Lattice, consultez Quotas pour VPC Lattice.

Résolution des problèmes

Si vous rencontrez des problèmes avec les points de terminaison des ressources VPC, vérifiez les points suivants.

- Si vous recevez un message d'erreur tel que « mount.nfs : accès refusé par le serveur lors du montage », vous devrez peut-être mettre à jour la configuration client de votre volume NFS.
- Vérifiez la configuration de vos ressources en effectuant des tests à partir d'une instance Amazon EC2 ou AWS CloudShell dans votre VPC.
- Testez votre connexion à Deadline Cloud à l'aide de tâches CLI simples. Pour plus d'informations, consultez les exemples de Deadline Cloud sur GitHub.
- Vérifiez les paramètres du groupe de sécurité de la passerelle de ressources en cas d'échec de connexion.
- Activez les journaux d'accès VPC pour surveiller les connexions.

Utilisez des pièces jointes à des tâches avec des flottes gérées par des services

Les pièces jointes aux tâches transfèrent des fichiers entre votre poste de travail et les employés de Deadline Cloud à l'aide d'Amazon Simple Storage Service (Amazon S3). Vous pouvez utiliser les pièces jointes aux tâches seules ou conjointement avec le stockage partagé pour associer des données auxiliaires aux tâches qui ne sont pas partagées avec d'autres tâches, telles que des scripts de tâche, des fichiers de configuration ou des actifs de projet stockés localement.

Pour plus d'informations sur le fonctionnement des pièces jointes aux tâches, consultez la section Pièces jointes du guide de l'utilisateur de Deadline Cloud. Pour plus de détails sur la spécification des fichiers d'entrée et de sortie dans les ensembles de tâches, consultezUtiliser les pièces jointes aux tâches pour partager des fichiers.

Choisissez un mode de système de fichiers

Lorsque vous soumettez une tâche avec des pièces jointes, vous pouvez choisir la manière dont les collaborateurs chargent les fichiers depuis Amazon S3 en définissant la `fileSystem` propriété :

- COPIÉ (par défaut) : télécharge tous les fichiers sur le disque local avant le début des tâches. Idéal lorsque chaque tâche nécessite le plus grand nombre de fichiers d'entrée.
- VIRTUEL — Monte un système de fichiers virtuel qui télécharge des fichiers à la demande. Il est préférable que les tâches n'aient besoin que d'un sous-ensemble de fichiers d'entrée. Disponible uniquement sur les serveurs SMF sous Linux.

> **Important**
>
> La mise en cache en mode VIRTUEL peut augmenter la consommation de mémoire et n'est pas optimisée pour toutes les charges de travail. Nous vous recommandons de tester votre charge de travail avant d'exécuter des tâches de production.

Pour des informations détaillées sur la configuration du mode système de fichiers, consultez la section Système de fichiers virtuel du guide de l'utilisateur de Deadline Cloud.

Optimisez les performances de transfert

La vitesse de synchronisation des fichiers entre Amazon S3 et les opérateurs SMF dépend de la configuration du volume Amazon Elastic Block Store (Amazon EBS) de votre flotte. Par défaut, les opérateurs SMF utilisent des volumes Amazon EBS gp3 avec des paramètres de performance de base. Pour les charges de travail comportant de gros fichiers d'entrée ou de nombreux petits fichiers, vous pouvez améliorer les vitesses de transfert en augmentant le débit et les paramètres d'IOPS d'Amazon EBS. Vous pouvez mettre à jour ces paramètres à l'aide du AWS Command Line Interface (AWS CLI).

Débit (Mio/s)

Vitesse à laquelle les données peuvent être lues ou écrites sur le volume. La valeur par défaut est 125 MiB/s, maximum is 1,000 MiB/s pour les volumes gp3. Augmentation pour les transferts de fichiers séquentiels volumineux.

IOPS

Opérations d'entrée/sortie par seconde. La valeur par défaut est de 3 000 IOPS, le maximum est de 16 000 IOPS pour les volumes gp3. Augmentez lors du transfert de nombreux petits fichiers.

> **Note**
>
> L'augmentation du débit et des IOPS d'Amazon EBS augmente le coût de la flotte. Pour plus d'informations sur les tarifs, consultez la section Tarification de Deadline Cloud.

Pour mettre à jour les paramètres Amazon EBS sur un parc existant à l'aide du AWS CLI

- Exécutez la commande suivante :

```
aws deadline update-fleet \
  --farm-id farm-0123456789abcdef0 \
  --fleet-id fleet-0123456789abcdef0 \
  --configuration '{
    "serviceManagedEc2": {
      "instanceCapabilities": {
        "vCpuCount": {"min": 4},
        "memoryMiB": {"min": 8192},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64",
        "rootEbsVolume": {
          "sizeGiB": 250,
          "iops": 6000,
          "throughputMiB": 500
        }
      },
      "instanceMarketOptions": {"type": "spot"}
    }
  }'
```

Télécharger les résultats des tâches

Une fois votre tâche terminée, téléchargez les fichiers de sortie à l'aide de la CLI de AWS Deadline Cloud ou du moniteur Deadline Cloud (moniteur Deadline Cloud) :

```
deadline job download-output --job-id job-0123456789abcdef0
```

Pour le téléchargement automatique des résultats au fur et à mesure que les tâches sont terminées, consultez la section Téléchargements automatiques du guide de l'utilisateur de Deadline Cloud.

Déployez et configurez des logiciels personnalisés sur les travailleurs

AWS Deadline Cloud propose plusieurs méthodes pour déployer et configurer des logiciels, des plugins et des outils personnalisés pour vos employés. La méthode que vous choisissez dépend de vos besoins, par exemple si vous avez besoin de privilèges d'administrateur, de la fréquence à laquelle le logiciel change et si le logiciel doit être disponible pour toutes les tâches ou uniquement pour des tâches spécifiques.

Choisissez une méthode de déploiement

Utilisez le tableau suivant pour choisir la méthode de déploiement adaptée à votre cas d'utilisation.

Critères	Environnement de file d'attente	Script de configuration de l'hôte	Forfait Conda personnalisé
Privilèges d'administrateur requis	Non	Oui	Non
Quand il fonctionne	Début de session	Démarrage par un travailleur	Début de session
Scope	Par file d'attente ou tâche	Tous les travailleurs de la flotte	Par file d'attente ou tâche
Peut être contrôlé par la soumission de tâches	Oui	Non	Oui
Complexité de configuration	Faible	Medium	Élevée
Idéal pour	Plugins, scripts, variables d'environnement simples	Pilotes système, Docker, supports de stockage	Applications complexes avec dépendances

Guide de décision rapide :

- Vous avez besoin de privilèges d'administrateur ou de root ? Utilisez un script de configuration de l'hôte.
- Plugin ou script simple sans droits d'administrateur ? Utilisez un environnement de file d'attente.
- Une application complexe nécessitant un contrôle de version ? Créez un package Conda personnalisé.

Configuration des tâches à l'aide d'environnements de file d'

AWS Deadline Cloud utilise des environnements de file d'attente pour configurer le logiciel sur vos employés. Un environnement vous permet d'effectuer des tâches chronophages, telles que la configuration et le démontage, une fois pour toutes les tâches d'une session. Il définit les actions à exécuter sur un travailleur lors du démarrage ou de l'arrêt d'une session. Vous pouvez configurer un environnement pour une file d'attente, les tâches exécutées dans la file d'attente et les étapes individuelles d'une tâche.

Vous définissez les environnements comme des environnements de file d'attente ou des environnements de travail. Créez des environnements de file d'attente avec la console Deadline Cloud ou avec l'CreateQueueEnvironmentopération deadline : et définissez des environnements de travail dans les modèles de tâches des tâches que vous soumettez. Ils suivent la spécification Open Job Description (OpenJD) pour les environnements. Pour plus de détails, reportez-voushttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment <Environment>à la spécification OpenJD sur GitHub.

Outre un `name` et`description`, chaque environnement contient deux champs qui définissent l'environnement sur l'hôte. Il s'agit des options suivantes :

- `script`— L'action entreprise lorsque cet environnement est exécuté sur un travailleur.
- `variables`— Ensemble de name/value paires de variables d'environnement définies lors de l'entrée dans l'environnement.

Vous devez définir au moins l'une des valeurs suivantes : `script` ou`variables`.

Vous pouvez définir plusieurs environnements dans votre modèle de tâche. Chaque environnement est appliqué dans l'ordre dans lequel il est répertorié dans le modèle. Vous pouvez l'utiliser pour gérer la complexité de vos environnements.

L'environnement de file d'attente par défaut pour Deadline Cloud utilise le gestionnaire de packages conda pour charger le logiciel dans l'environnement, mais vous pouvez utiliser d'autres gestionnaires de packages. L'environnement par défaut définit deux paramètres pour spécifier le logiciel à charger. Ces variables sont définies par les émetteurs fournis par Deadline Cloud, mais vous pouvez les définir dans vos propres scripts et applications utilisant l'environnement par défaut. Il s'agit des options suivantes :

- `CondaPackages`— Une liste séparée par des espaces des packages conda correspondant aux spécifications à installer pour le travail. Par exemple, l'émetteur de Blender ajouterait des images `blender=3.6` au rendu dans Blender 3.6.
- `CondaChannels`— Liste séparée par des espaces de canaux conda à partir desquels installer les packages. Pour les flottes gérées par des services, les packages sont installés depuis le canal. `deadline-cloud` Vous pouvez ajouter d'autres chaînes.

Contrôlez l'environnement de travail avec les environnements de file d'attente OpenJD

Vous pouvez définir des environnements personnalisés pour vos tâches de rendu à l'aide d'environnements de file d'attente. Un environnement de file d'attente est un modèle qui contrôle les variables d'environnement, les mappages de fichiers et les autres paramètres des tâches exécutées dans une file d'attente spécifique. Il vous permet d'adapter l'environnement d'exécution des tâches soumises à une file d'attente aux exigences de vos charges de travail. AWS Deadline Cloud propose trois niveaux imbriqués dans lesquels vous pouvez appliquer les environnements Open Job Description (OpenJD) : file d'attente, job et step. En définissant des environnements de file d'attente, vous pouvez garantir des performances cohérentes et optimisées pour différents types de tâches, rationaliser l'allocation des ressources et simplifier la gestion des files d'attente.

L'environnement de file d'attente est un modèle que vous attachez à une file d'attente de votre AWS compte depuis la console de AWS gestion ou à l'aide du AWS CLI. Vous pouvez créer un environnement pour une file d'attente, ou vous pouvez créer plusieurs environnements de file d'attente qui s'appliquent afin de créer l'environnement d'exécution. Cette approche vous permet de créer et de tester un environnement par étapes afin de vous assurer qu'il fonctionne correctement pour vos tâches.

Les environnements des tâches et des étapes sont définis dans le modèle de tâche que vous utilisez pour créer une tâche dans votre file d'attente. La syntaxe OpenJD est la même dans ces différentes

formes d'environnements. Dans cette section, nous les montrerons à l'intérieur des modèles de tâches.

Rubriques

Définir des variables d'environnement dans un environnement de file d'attente

De nombreuses applications et frameworks utilisent des variables d'environnement pour contrôler les paramètres des fonctionnalités, les niveaux de journalisation et la configuration de l'affichage. Vous pouvez utiliser les environnements Open Job Description (OpenJD) pour définir des variables d'environnement dont héritera chaque commande de tâche relevant de leur champ d'application.

Portée de la variable d'environnement

AWS Deadline Cloud applique des variables d'environnement issues des environnements de file d'attente que vous attachez à une file d'attente. Dans un modèle de tâche, vous pouvez également définir des variables d'environnement au niveau de la tâche et de l'étape à l'aide des environnements OpenJD. Les variables définies dans une portée plus étroite remplacent les variables portant le même nom dans une portée plus large.

- Environnement de file d'attente : modèle que vous attachez à une file d'attente dans Deadline Cloud. Les variables s'appliquent à toutes les tâches soumises à la file d'attente. Vous pouvez définir des variables à l'aide d'une `variables` carte pour les valeurs fixes ou utiliser des scripts pour les valeurs dynamiques.
- Environnement de travail : défini ci-dessous `jobEnvironments` dans un modèle de travail. Les variables s'appliquent à toutes les étapes et tâches du travail. Une variable au niveau de la tâche remplace une variable au niveau de la file d'attente portant le même nom.
- Environnement des étapes : défini ci-dessous `stepEnvironments` dans un modèle de tâche. Les variables s'appliquent uniquement aux tâches de cette étape. Une variable de niveau étape remplace une variable de niveau tâche ou de file d'attente portant le même nom.

Définition de variables dans un environnement de file d'attente

Vous pouvez définir des variables d'environnement dans un environnement de file d'attente à l'aide d'une `variables` carte pour les valeurs fixes ou à l'`script`aide d'une `onEnter` action pour les valeurs dynamiques.

Le modèle d'environnement de file d'attente suivant utilise une `variables` carte pour définir la `QT_QPA_PLATFORM` variable sur`offscreen`, ce qui permet aux applications utilisant le Qt Framework de s'exécuter sur des hôtes de travail sans affichage interactif.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Pour les valeurs dynamiques, telles que la modification `PATH` ou l'activation d'environnements virtuels, utilisez un script qui imprime des lignes `openjd_env: VAR=value` au format standard. Le `openjd_env:` préfixe est obligatoire. L'utilisation de `echo``export`, ou d'autres mécanismes shell sans le préfixe ne propage pas les variables vers les jobs et les tâches.

Le modèle d'environnement de file d'attente suivant définit la `QT_QPA_PLATFORM` variable à l'aide d'un script.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

Pour associer un environnement de file d'attente à votre file d'attente, utilisez la console Deadline Cloud ou le AWS CLI. Pour plus d'informations, consultez la section Création d'un environnement de file d'attente dans le guide de l'utilisateur de AWS Deadline Cloud. La AWS CLI commande suivante crée un environnement de file d'attente à partir d'un fichier modèle.

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

Pour des exemples plus complexes, tels que la création et l'activation d'environnements virtuels Conda, consultez les exemples d'environnement de file d'attente Deadline Cloud sur GitHub.

Définition de variables dans un modèle de tâche

Dans un modèle de tâche, ajoutez une `variables` carte à une `stepEnvironments` entrée `jobEnvironments` ou. Chaque entrée est une paire clé-valeur où la clé est le nom de la variable et la valeur est la valeur de la variable.

Le modèle de tâche suivant définit la variable d'`QT_QPA_PLATFORM`environnement sur`offscreen`, ce qui permet aux applications utilisant le Qt Framework de s'exécuter sur des hôtes de travail sans affichage interactif.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

Vous pouvez définir plusieurs variables dans une seule définition d'environnement.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
```

```
    QT_QPA_PLATFORM: offscreen
```

Vous pouvez référencer les paramètres de tâche dans des valeurs variables à l'aide de la `{{Param.ParameterName}}` syntaxe.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Pour remplacer une variable de niveau tâche pour une étape spécifique, définissez une `stepEnvironments` entrée portant le même nom de variable. L'exemple suivant définit `JOB_PROJECT_ID` au niveau de la tâche avec la valeur`project-12`, puis remplace la valeur au niveau de l'étape par`step-project-12`. Les tâches de l'étape utilisent la valeur du niveau de l'étape.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    JOB_PROJECT_ID: project-12
steps:
- name: MyStep
  stepEnvironments:
  - name: StepEnv
    variables:
      JOB_PROJECT_ID: step-project-12
```

Essayez-le : Exécution de l'exemple de variable d'environnement

Le référentiel d'exemples de Deadline Cloud inclut un ensemble de tâches qui explique la définition et l'affichage des variables d'environnement. L'exemple de modèle de tâche définit des variables au niveau de la tâche et de l'étape, puis exécute une tâche qui imprime le résultat fusionné. Utilisez la procédure suivante pour exécuter l'échantillon et inspecter les résultats.

Conditions préalables

1. Si vous ne possédez pas de ferme Deadline Cloud avec une file d'attente et un parc Linux associé, suivez l'expérience d'intégration guidée dans la console Deadline Cloud pour en créer une avec les paramètres par défaut.

2. Si vous ne disposez pas de la CLI Deadline Cloud et du moniteur AWS Deadline Cloud sur votre poste de travail, suivez les étapes décrites dans Configurer les émetteurs Deadline Cloud.
3. `git`À utiliser pour cloner le GitHubréférentiel d'échantillons de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cd deadline-cloud-samples/job_bundles
```

Exécution de l'exemple

1. Utilisez la CLI de Deadline Cloud pour envoyer l'`job_env_vars`échantillon.

```
deadline bundle submit job_env_vars
```

2. Dans le moniteur Deadline Cloud, sélectionnez la nouvelle tâche pour suivre sa progression. Une fois qu'un travailleur est disponible dans le Linux parc associé à la file d'attente, la tâche se termine en quelques secondes. Sélectionnez la tâche, puis choisissez Afficher les journaux dans le menu supérieur droit du panneau des tâches.

Comparaison des actions de session avec leurs définitions

La vue du journal montre trois actions de session. Ouvrez le fichier job_env_vars/template.yaml dans un éditeur de texte pour comparer chaque action avec sa définition dans le modèle de tâche.

1. Sélectionnez l'action Lancer une JobEnv session. La sortie du journal indique les variables d'environnement définies au niveau du travail.

```
Setting: JOB_VERBOSITY=MEDIUM
Setting: JOB_EXAMPLE_PARAM=An example parameter value
Setting: JOB_PROJECT_ID=project-12
Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/path
Setting: QT_QPA_PLATFORM=offscreen
```

   Les lignes suivantes du modèle de tâche définissent cet environnement.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

```
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. Sélectionnez l'action Lancer une StepEnv session. La sortie du journal indique les variables au niveau de l'étape, y compris les variables remplacéesJOB_PROJECT_ID.

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   Les lignes suivantes du modèle de tâche définissent cet environnement.

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Sélectionnez l'action Task Run session. La sortie du journal indique les variables d'environnement fusionnées disponibles pour la tâche. Notez que cela JOB_PROJECT_ID utilise la valeur `step-project-12` au niveau de l'étape.

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

Définir le chemin dans un environnement de file d'attente

Utilisez les environnements OpenJD pour fournir de nouvelles commandes dans un environnement. Vous créez d'abord un répertoire contenant des fichiers de script, puis vous ajoutez ce répertoire aux variables d'PATHenvironnement afin que les exécutables de votre script puissent les exécuter sans avoir à spécifier le chemin du répertoire à chaque fois. La liste des variables d'une définition d'environnement ne permet pas de modifier la variable. Pour ce faire, exécutez plutôt un script. Une

fois que le script a configuré les choses et les a modifiéesPATH, il exporte la variable vers le runtime OpenJD avec la commande. `echo "openjd_env: PATH=$PATH"`

Conditions préalables

Procédez comme suit pour exécuter l'exemple de bundle de tâches avec des variables d'environnement provenant du référentiel github d'échantillons de Deadline Cloud.

1. Si vous ne possédez pas de ferme Deadline Cloud avec une file d'attente et un parc Linux associé, suivez l'expérience d'intégration guidée dans la console Deadline Cloud pour en créer une avec les paramètres par défaut.
2. Si vous ne disposez pas de la CLI Deadline Cloud et du moniteur Deadline Cloud sur votre poste de travail, suivez les étapes décrites dans la section Configurer les émetteurs Deadline Cloud dans le guide de l'utilisateur.
3. `git`À utiliser pour cloner le GitHubréférentiel d'échantillons de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Exécutez l'exemple de chemin

1. Utilisez la CLI de Deadline Cloud pour envoyer l'`job_env_with_new_command`échantillon.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. Dans le moniteur Deadline Cloud, vous verrez la nouvelle tâche et pourrez suivre sa progression. Une fois que le Linux parc associé à la file d'attente dispose d'un collaborateur disponible pour exécuter la tâche, celle-ci se termine en quelques secondes. Sélectionnez la tâche, puis choisissez l'option Afficher les journaux dans le menu supérieur droit du panneau des tâches.

   Sur la droite se trouvent deux actions de session, Lancer RandomSleepCommand et Exécuter une tâche. L'afficheur de journal au centre de la fenêtre correspond à l'action de session sélectionnée sur la droite.

Comparez les actions de session avec leurs définitions

Dans cette section, vous utilisez le moniteur Deadline Cloud pour comparer les actions de session avec celles définies dans le modèle de tâche. Il fait suite à la section précédente.

Ouvrez le fichier job_env_with_new_command/template.yaml dans un éditeur de texte. Comparez les actions de session à l'endroit où elles sont définies dans le modèle de tâche.

1. Sélectionnez l'action Lancer une RandomSleepCommand session dans le moniteur Deadline Cloud. Vous verrez la sortie du journal comme suit.

```
 2024/07/16 17:25:32-07:00
 2024/07/16 17:25:32-07:00 ================================================
 2024/07/16 17:25:32-07:00 --------- Entering Environment: RandomSleepCommand
 2024/07/16 17:25:32-07:00 ================================================
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Setup
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
 2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Running action
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
 2024/07/16 17:25:32-07:00 Command started as pid: 2205
 2024/07/16 17:25:32-07:00 Output:
 2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
 No newer logs at this moment.
```

   Les lignes suivantes du modèle de tâche spécifient cette action.

```
jobEnvironments:
- name: RandomSleepCommand
  description: Adds a command 'random-sleep' to the environment.
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail

        # Make a bin directory inside the session's working directory for providing
 new commands
        mkdir -p '{{Session.WorkingDirectory}}/bin'

        # If this bin directory is not already in the PATH, then add it
        if ! [[ ":$PATH:" == *':{{Session.WorkingDirectory}}/bin:'* ]]; then
          export "PATH={{Session.WorkingDirectory}}/bin:$PATH"

          # This message to Open Job Description exports the new PATH value to the
 environment
          echo "openjd_env: PATH=$PATH"
        fi

        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...
```

2. Sélectionnez l'action Lancer une StepEnv session dans le moniteur Deadline Cloud. La sortie du journal s'affiche comme suit.

```
 2024/07/16 17:25:33-07:00
 2024/07/16 17:25:33-07:00 ==========================================
 2024/07/16 17:25:33-07:00 --------- Running Task
 2024/07/16 17:25:33-07:00 ==========================================
 2024/07/16 17:25:33-07:00 ------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Setup
 2024/07/16 17:25:33-07:00 ------------------------------------------
 2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
 2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 ------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Running action
 2024/07/16 17:25:33-07:00 ------------------------------------------
 2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
 2024/07/16 17:25:33-07:00 Command started as pid: 2256
 2024/07/16 17:25:33-07:00 Output:
 2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
 2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
 2024/07/16 17:26:00-07:00 ------------------------------------------
 2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
 2024/07/16 17:26:00-07:00 ------------------------------------------
```

3. Les lignes suivantes du modèle de tâche spécifient cette action.

```
steps:
- name: EnvWithCommand
  script:
    actions:
      onRun:
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      data: |
        set -xeuo pipefail

        # Run the script installed into PATH by the job environment
```

```
        random-sleep 12.5 27.5
  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Exécuter un processus daemon en arrière-plan depuis l'environnement de file d'attente

Dans de nombreux cas d'utilisation du rendu, le chargement de l'application et des données de scène peut prendre beaucoup de temps. Si une tâche les recharge pour chaque image, elle consacrera le plus clair de son temps aux frais généraux. Il est souvent possible de charger l'application une seule fois en tant que processus démon en arrière-plan, de lui faire charger les données de scène, puis de lui envoyer des commandes via une communication inter-processus (IPC) pour effectuer les rendus.

La plupart des intégrations open source de Deadline Cloud utilisent ce modèle. Le projet Open Job Description fournit une bibliothèque d'exécution d'adaptateurs avec des modèles IPC robustes sur tous les systèmes d'exploitation pris en charge.

Pour illustrer ce modèle, il existe un exemple de bundle de tâches autonome qui utilise Python et du code bash pour implémenter un démon d'arrière-plan et l'IPC pour les tâches permettant de communiquer avec celui-ci. Le démon est implémenté en Python et écoute un SIGUSR1 signal POSIX indiquant quand traiter une tâche. Les détails de la tâche sont transmis au démon dans un fichier JSON spécifique, et les résultats de l'exécution de la tâche sont renvoyés sous la forme d'un autre fichier JSON.

Conditions préalables

Procédez comme suit pour exécuter l'exemple de bundle de tâches avec un processus démon à partir du référentiel github d'échantillons de Deadline Cloud.

1. Si vous ne possédez pas de ferme Deadline Cloud avec une file d'attente et un parc Linux associé, suivez l'expérience d'intégration guidée dans la console Deadline Cloud pour en créer une avec les paramètres par défaut.
2. Si vous ne disposez pas de la CLI Deadline Cloud et du moniteur Deadline Cloud sur votre poste de travail, suivez les étapes décrites dans la section Configurer les émetteurs Deadline Cloud dans le guide de l'utilisateur.
3. `git`À utiliser pour cloner le GitHubréférentiel d'échantillons de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Exécutez l'exemple de daemon

1. Utilisez la CLI de Deadline Cloud pour envoyer l'`job_env_daemon_process`échantillon.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. Dans l'application de surveillance Deadline Cloud, vous verrez le nouveau travail et pourrez suivre sa progression. Une fois que le Linux parc associé à la file d'attente dispose d'un collaborateur disponible pour exécuter la tâche, celle-ci est terminée en une minute environ. Une fois l'une des tâches sélectionnée, choisissez l'option Afficher les journaux dans le menu supérieur droit du panneau des tâches.

   Sur la droite, il y a deux actions de session, Lancer DaemonProcess et Exécuter une tâche. L'afficheur de journal au centre de la fenêtre correspond à l'action de session sélectionnée sur la droite.

   Sélectionnez l'option Afficher les journaux de toutes les tâches. La chronologie montre le reste des tâches exécutées dans le cadre de la session, ainsi que l'`Shut down DaemonProcess`action qui a quitté l'environnement.

Afficher les journaux des démons

1. Dans cette section, vous utilisez le moniteur Deadline Cloud pour comparer les actions de session avec celles définies dans le modèle de tâche. Il fait suite à la section précédente.

   Ouvrez le fichier job_env_daemon_process/template.yaml dans un éditeur de texte. Comparez les actions de session à l'endroit où elles sont définies dans le modèle de tâche.

2. Sélectionnez l'action de `Launch DaemonProcess` session dans le moniteur Deadline Cloud. Vous verrez la sortie du journal comme suit.

```
 2024/07/17 16:27:20-07:00
 2024/07/17 16:27:20-07:00 ==============================================
 2024/07/17 16:27:20-07:00 --------- Entering Environment: DaemonProcess
 2024/07/17 16:27:20-07:00 ==============================================
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Setup
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Running action
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
 2024/07/17 16:27:20-07:00 Command started as pid: 2187
 2024/07/17 16:27:20-07:00 Output:
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
```

```
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

Les lignes suivantes du modèle de tâche spécifient cette action.

```
  stepEnvironments:
  - name: DaemonProcess
    description: Runs a daemon process for the step's tasks to share.
    script:
      actions:
        onEnter:
          command: bash
          args:
          - "{{Env.File.Enter}}"
        onExit:
          command: bash
          args:
          - "{{Env.File.Exit}}"
      embeddedFiles:
      - name: Enter
        filename: enter-daemon-process-env.sh
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail

          DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
          echo "openjd_env: DAEMON_LOG=$DAEMON_LOG"
          nohup python {{Env.File.DaemonScript}} > $DAEMON_LOG 2>&1 &
          echo "openjd_env: DAEMON_PID=$!"
          echo "openjd_env:
DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

          echo 0 > 'daemon_log_cursor.txt'
    ...
```

3. Sélectionnez l'une des actions de session Task run : N dans le moniteur Deadline Cloud. Vous verrez la sortie du journal comme suit.

```
2024/07/17 16:27:22-07:00
 2024/07/17 16:27:22-07:00 ============================================
 2024/07/17 16:27:22-07:00 --------- Running Task
```

```
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 Parameter values:
 2024/07/17 16:27:22-07:00 Frame(INT) = 2
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Setup
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
 2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Running action
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
 2024/07/17 16:27:22-07:00 Command started as pid: 2301
 2024/07/17 16:27:22-07:00 Output:
 2024/07/17 16:27:23-07:00 Daemon PID is 2223
 2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Previous output from daemon
 2024/07/17 16:27:23-07:00 ===
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 Sending command to daemon
 2024/07/17 16:27:23-07:00 Received task result:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00   "result": "SUCCESS",
 2024/07/17 16:27:23-07:00   "processedTaskCount": 1,
 2024/07/17 16:27:23-07:00   "randomValue": 0.2578537967668988,
 2024/07/17 16:27:23-07:00   "failureRate": 0.1
 2024/07/17 16:27:23-07:00 }
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Daemon log from running the task
 2024/07/17 16:27:23-07:00 Loading the task details file
 2024/07/17 16:27:23-07:00 Received task details:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00  "pid": 2329,
 2024/07/17 16:27:23-07:00  "frame": 2
 2024/07/17 16:27:23-07:00 }
 2024/07/17 16:27:23-07:00 Processing frame number 2
 2024/07/17 16:27:23-07:00 Writing result
 2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
```

```
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 ------------------------------------------------
2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
2024/07/17 16:27:23-07:00 ------------------------------------------------
```

Les lignes suivantes du modèle de tâche spécifient cette action. Étapes :

```
steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      filename: run-task.sh
      type: TEXT
      data: |
        # This bash script sends a task to the background daemon process,
        # then waits for it to respond with the output result.

        set -euo pipefail

        source "$DAEMON_BASH_HELPER_SCRIPT"

        echo "Daemon PID is $DAEMON_PID"
        echo "Daemon log file is $DAEMON_LOG"

        print_daemon_log "Previous output from daemon"
```

```
        send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
        wait_for_daemon_task_result

        echo Received task result:
        echo "$TASK_RESULT" | jq .

        print_daemon_log "Daemon log from running the task"

  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Soumettez des candidatures pour vos emplois

Vous pouvez utiliser un environnement de file d'attente pour charger des applications afin de traiter vos tâches. Lorsque vous créez un parc géré par des services à l'aide de la console Deadline Cloud, vous avez la possibilité de créer un environnement de file d'attente qui utilise le gestionnaire de packages Conda pour charger les applications.

Si vous souhaitez utiliser un autre gestionnaire de packages, vous pouvez créer un environnement de file d'attente pour ce gestionnaire. Pour un exemple d'utilisation de Rez, voirUtiliser un autre gestionnaire de packages.

Deadline Cloud fournit un canal conda pour charger une sélection d'applications de rendu dans votre environnement. Ils prennent en charge les soumetteurs fournis par Deadline Cloud pour les applications de création de contenu numérique.

Vous pouvez également charger un logiciel pour conda-forge à utiliser dans le cadre de vos travaux. Les exemples suivants montrent des modèles de tâches utilisant l'environnement de file d'attente fourni par Deadline Cloud pour charger des applications avant d'exécuter la tâche.

Rubriques

- Obtenir une application depuis un canal Conda
- Utiliser un autre gestionnaire de packages

Obtenir une application depuis un canal Conda

Vous pouvez créer un environnement de file d'attente personnalisé pour vos employés de Deadline Cloud, qui installe le logiciel de votre choix. Cet exemple d'environnement de file d'attente présente le même comportement que l'environnement utilisé par la console pour les flottes gérées par des services. Il exécute directement conda pour créer l'environnement.

L'environnement crée un nouvel environnement virtuel Conda pour chaque session Deadline Cloud exécutée sur un travailleur, puis supprime l'environnement une fois l'opération terminée.

Conda met en cache les packages téléchargés afin qu'ils n'aient pas besoin d'être téléchargés à nouveau, mais chaque session doit lier tous les packages à l'environnement.

L'environnement définit trois scripts qui s'exécutent lorsque Deadline Cloud démarre une session sur un worker. Le premier script s'exécute lorsque l'`onEnter`action est appelée. Il appelle les deux autres pour configurer les variables d'environnement. Une fois l'exécution du script terminée, l'environnement conda est disponible avec toutes les variables d'environnement spécifiées définies.

Pour la dernière version de l'exemple, consultez conda_queue_env_console_equivalent.yaml dans le référentiel sur. deadline-cloud-samples GitHub

Si vous souhaitez utiliser une application qui n'est pas disponible dans le canal conda, vous pouvez créer un canal conda dans Amazon S3, puis créer vos propres packages pour cette application. Pour en savoir plus, consultez Création d'un canal conda à l'aide de S3.

Obtenez des bibliothèques open source auprès de conda-forge

Cette section décrit comment utiliser les bibliothèques open source de la `conda-forge` chaîne. L'exemple suivant est un modèle de tâche qui utilise le package `polars` Python.

La tâche définit les `CondaChannels` paramètres `CondaPackages` et définis dans l'environnement de file d'attente qui indiquent à Deadline Cloud où se procurer le package.

La section du modèle de tâche qui définit les paramètres est la suivante :

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
 to handle this.
  type: STRING
  default: polars
```

```
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
 Environment to handle this.
  type: STRING
  default: conda-forge
```

Pour obtenir la dernière version de l'exemple complet de modèle de tâche, consultez stage_1_self_contained_template/template.yaml. Pour la dernière version de l'environnement de file d'attente qui charge les packages conda, consultez conda_queue_env_console_equivalent.yaml dans le référentiel sur. deadline-cloud-samples GitHub

Accédez Blender à la chaîne Deadline-Cloud

L'exemple suivant montre un modèle de tâche Blender issu du canal `deadline-cloud` conda. Ce canal prend en charge les émetteurs fournis par Deadline Cloud pour les logiciels de création de contenu numérique, mais vous pouvez utiliser le même canal pour charger le logiciel pour votre propre usage.

Pour obtenir la liste des logiciels fournis par le `deadline-cloud` canal, consultez la section Environnement de file d'attente par défaut dans le guide de l'utilisateur de AWS Deadline Cloud.

Cette tâche définit le `CondaPackages` paramètre défini dans l'environnement de file d'attente pour indiquer à Deadline Cloud de le charger Blender dans l'environnement.

La section du modèle de tâche qui définit le paramètre est la suivante :

```
- name: CondaPackages
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
 channel.
```

Pour obtenir la dernière version de l'exemple complet de modèle de tâche, consultez blender_render/template.yaml. Pour la dernière version de l'environnement de file d'attente qui charge les packages conda, consultez conda_queue_env_console_equivalent.yaml dans le référentiel sur. deadline-cloud-samplesGitHub

Utiliser un autre gestionnaire de packages

Le gestionnaire de packages par défaut pour Deadline Cloud est conda. Si vous devez utiliser un autre gestionnaire de packages, par exempleRez, vous pouvez créer un environnement de file d'attente personnalisé contenant des scripts utilisant votre gestionnaire de packages à la place.

Cet exemple d'environnement de file d'attente présente le même comportement que l'environnement utilisé par la console pour les flottes gérées par des services. Il remplace le gestionnaire de packages conda parRez.

L'environnement définit trois scripts qui s'exécutent lorsque Deadline Cloud démarre une session sur un worker. Le premier script s'exécute lorsque l'`onEnter`action est appelée. Il appelle les deux autres pour configurer les variables d'environnement. Une fois l'exécution du script terminée, l'Rezenvironnement est disponible avec toutes les variables d'environnement spécifiées définies.

L'exemple suppose que vous avez une flotte gérée par le client qui utilise un système de fichiers partagé pour les packages Rez.

Pour la dernière version de l'exemple, consultez rez_queue_env.yaml dans le référentiel sur. deadline-cloud-samplesGitHub

Création d'un canal conda à l'aide de S3

Si vos tâches doivent exécuter des applications non disponibles sur les `conda-forge`canaux `deadline-cloud`OR, vous pouvez héberger un canal conda personnalisé pour diffuser vos propres packages. Lorsque vous créez une file d'attente dans la console AWS Deadline Cloud (Deadline Cloud), la console ajoute un environnement de file d'attente conda par défaut. Pour que vos packages soient disponibles pour les tâches, ajoutez le canal personnalisé à l'environnement de file d'attente.

Un canal conda est un contenu hébergé statique que vous pouvez héberger de différentes manières, notamment sur un système de fichiers ou dans un bucket Amazon Simple Storage Service (Amazon S3). Si votre ferme Deadline Cloud utilise un système de fichiers partagé pour les ressources, vous pouvez utiliser n'importe quel chemin comme nom de canal. Vous pouvez héberger le canal dans un compartiment Amazon S3 pour un accès plus large à l'aide des autorisations Gestion des identités et des accès AWS (IAM).

Vous pouvez créer et tester des packages localement, puis les publier sur une chaîne. La création de packages localement est un moyen facile de commencer à itérer sur des recettes de construction

de packages sans configuration d'infrastructure. Vous pouvez également utiliser une file d'attente de création de packages Deadline Cloud pour créer des packages et les publier sur une chaîne. Une file d'attente de création de packages simplifie la maintenance des packages pour plusieurs systèmes d'exploitation et configurations d'accélérateurs. Vous pouvez mettre à jour les versions et soumettre des ensembles complets de compilations de packages où que vous soyez.

Vous pouvez configurer les chaînes pour votre studio et votre ferme Deadline Cloud de différentes manières. Vous pouvez avoir un canal Amazon S3 et configurer tous vos postes de travail et hôtes de ferme pour qu'ils l'utilisent. Vous pouvez également avoir plusieurs canaux et configurer la mise en miroir avec AWS DataSync (DataSync). Par exemple, la file d'attente de création de votre package Deadline Cloud peut être publiée sur un canal Amazon S3 qui est reproduit sur site pour les postes de travail et les hôtes de ferme sur site.

Rubriques

Créez et testez des packages localement

Avant de publier des packages sur Amazon S3 ou de configurer CI/CD l'automatisation sur votre ferme Deadline Cloud, vous pouvez créer et tester des packages conda sur votre poste de travail à l'aide d'un canal de système de fichiers local. Cette approche vous permet d'itérer rapidement localement sur les recettes et de vérifier les packages.

La `rattler-build publish` commande crée une recette, copie le package obtenu sur un canal et indexe le canal en une seule étape. Lorsque vous ciblez un répertoire de système de fichiers local, `rattler-build` crée et initialise le canal automatiquement si le répertoire n'existe pas.

Les instructions suivantes utilisent l'exemple de recette Blender 4.5 du référentiel d'exemples de Deadline Cloud surGitHub. Vous pouvez remplacer une recette différente depuis le référentiel d'échantillons ou utiliser votre propre recette.

Conditions préalables

Avant de commencer, installez les outils suivants sur votre poste de travail :

- pixi — Un gestionnaire de paquets que vous utilisez pour installer `rattler-build` et tester des packages. Installez pixi depuis pixi.sh.
- rattler-build — L'outil de création de packages utilisé par Deadline Cloud Conda Recipes. Après avoir installé Pixi, exécutez la commande suivante pour effectuer l'installation`rattler-build`.

```
pixi global install rattler-build
```

- git — Nécessaire pour cloner le dépôt d'échantillons. WindowsActivé, git for fournit Windows également un `bash` shell, dont certains Windows exemples de recettes ont besoin.

Création et publication d'un package sur une chaîne locale

Dans cette procédure, vous clonez le référentiel d'échantillons de Deadline Cloud et vous l'utilisez `rattler-build publish` pour créer et publier le package sur un canal de système de fichiers local.

> **Note**
>
> Les applications volumineuses peuvent nécessiter des dizaines de Go d'espace disque libre pour l'archive source, les fichiers extraits et la sortie de compilation. Assurez-vous d'utiliser un disque avec suffisamment d'espace disponible pour la sortie de génération du package.

Pour créer et publier un package sur une chaîne locale

1. Clonez le référentiel d'échantillons de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

2. Passez au répertoire `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Exécutez la commande suivante pour créer la recette Blender 4.5 et publier le package dans un répertoire de canaux local.

   LinuxActivé macOS et exécutez la commande suivante.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to file://$HOME/my-conda-channel \
       --build-number=+1
   ```

   Sur Windows (cmd), exécutez la commande suivante.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml ^
       --to file://%USERPROFILE%/my-conda-channel ^
       --build-number=+1
   ```

   La `rattler-build publish` commande exécute les actions suivantes :

   - Construit le package à partir de la recette.
   - Crée le répertoire des chaînes s'il n'existe pas.
   - Copie le fichier du package sur le canal.
   - Indexe le canal afin que les gestionnaires de packages puissent le trouver.

   Si la recette de votre package dépend de packages provenant d'un canal particulier, tel que conda-forge, ajoutez `-c conda-forge` à la commande.

> **À propos des numéros de version**
>
> L'`--build-number=+1`option sélectionne automatiquement le numéro de version suivant en fonction de ce qui existe déjà dans le canal de destination. La meilleure pratique consiste à ne jamais remplacer un package dans une chaîne. Créez toujours avec un nouveau numéro de version si le package aurait autrement le même nom de fichier. L'utilisation `--build-number=+1` permet d'atteindre cet objectif lorsque vous créez une chaîne de production ou une chaîne intermédiaire qui reflète la production.

Si vous souhaitez contrôler directement le numéro de version, vous pouvez le définir avec une valeur spécifique telle que`--build-number=7`. Si vous omettez cette option, `rattler-build` utilise le numéro de version défini dans le `recipe.yaml` fichier.

Pour plus d'informations`rattler-build publish`, consultez la documentation de publication de Rattler-build.

Débogage des versions

En cas d'échec d'une compilation, `rattler-build` préserve le répertoire de compilation afin que vous puissiez l'examiner. Exécutez la commande suivante pour ouvrir un shell interactif dans l'environnement de construction avec toutes les variables d'environnement configurées telles qu'elles étaient lors de la génération.

```
rattler-build debug shell
```

À partir du shell de débogage, vous pouvez modifier des fichiers, exécuter des commandes de compilation individuelles et ajouter des dépendances pour isoler le problème. Pour plus d'informations, consultez la section Débogage des versions dans la documentation de Rattler-build.

Tester le package

Après avoir créé et publié le package, créez un projet pixi temporaire. Utilisez le projet pour installer le package depuis le canal local et vérifiez qu'il fonctionne correctement.

Pour tester le package

1. Créez un répertoire de test temporaire et initialisez un projet pixi avec le canal local.

   LinuxActivez macOS et exécutez les commandes suivantes.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://$HOME/my-conda-channel
   ```

   Sur Windows (cmd), exécutez les commandes suivantes.

   ```
   mkdir package-test-env
   cd package-test-env
   ```

```
pixi init --channel file://%USERPROFILE%/my-conda-channel
```

2. Ajoutez le package au projet.

```
pixi add blender=4.5
```

3. Vérifiez que le package fonctionne correctement.

```
pixi run blender --version
```

La `pixi run`commande active l'environnement conda pour le répertoire du projet et exécute la commande spécifiée dans celui-ci. L'environnement est conservé dans le répertoire du projet, vous pouvez donc utiliser la même `pixi run` commande depuis d'autres terminaux.

Lorsque vous êtes satisfait du package, vous pouvez le publier sur un canal conda Amazon S3 afin que les utilisateurs de Deadline Cloud puissent l'installer. Voir Publier des packages sur un canal conda S3.

Supprimer des packages de la chaîne

Évitez de supprimer des packages des canaux que vous utilisez pour la production, car les fichiers de verrouillage font référence à des packages spécifiques par hachage. La suppression d'un package empêche de recréer des environnements à partir de ces fichiers de verrouillage. Pour les canaux de développement et de test, vous pouvez supprimer un package spécifique en supprimant le `.conda` fichier du répertoire des canaux, puis en réindexant le canal. Tout d'abord, installez`rattler-index`.

```
pixi global install rattler-index
```

Supprimez ensuite le fichier du package et réindexez le canal.

LinuxActivez macOS et exécutez les commandes suivantes.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

Sur Windows (cmd), exécutez les commandes suivantes.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

Les fichiers de package sont stockés dans des sous-répertoires spécifiques à la plate-forme tels `linux-64` que, ou. `win-64` `osx-arm64` Répertoriez le contenu de ces sous-répertoires pour trouver le nom de fichier exact du package que vous souhaitez supprimer.

Nettoyage

Après le test, vous pouvez supprimer le projet de test et le canal local.

Pour nettoyer les ressources de test

1. Supprimez le répertoire du projet de test.

   LinuxActivé macOS et exécutez la commande suivante.

   ```
   rm -rf package-test-env
   ```

   Sur Windows (cmd), exécutez la commande suivante.

   ```
   rmdir /s /q package-test-env
   ```

2. Supprimez le répertoire local des chaînes Conda.

   LinuxActivé macOS et exécutez la commande suivante.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Sur Windows (cmd), exécutez la commande suivante.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Facultatif) Supprimez le répertoire `rattler-build` de sortie qui contient le fichier de package créé.

   LinuxActivé macOS et exécutez la commande suivante.

   ```
   rm -rf deadline-cloud-samples/conda_recipes/output
   ```

   Sur Windows (cmd), exécutez la commande suivante.

   ```
   rmdir /s /q deadline-cloud-samples\conda_recipes\output
   ```

Publier des packages sur un canal conda Amazon S3

Vous pouvez publier des packages conda dans un bucket Amazon Simple Storage Service (Amazon S3) afin AWS que les employés de Deadline Cloud (Deadline Cloud) puissent les installer pour exécuter des tâches. La `rattler-build publish` commande fonctionne avec Amazon S3 de la même manière qu'avec un canal de système de fichiers local. La commande peut créer une recette et publier le résultat, ou publier un fichier de package que vous avez déjà créé. Dans les deux cas, la commande télécharge le package dans le bucket et indexe le canal en une seule étape.

La `rattler-build publish` commande s'authentifie à AWS l'aide de la chaîne d'identification standard. Elle utilise donc votre AWS configuration comme n'importe quel AWS outil. Pour plus d'informations sur la configuration des informations d'identification, consultez la section Configuration et paramètres du fichier d'informations d'identification dans le guide de l'utilisateur AWS Command Line Interface (AWS CLI).

Conditions préalables

Avant de publier des packages sur Amazon S3, remplissez les conditions préalables suivantes :

- pixi et rattler-build — Installez pixi depuis pixi.sh, puis installez-le. `rattler-build`

  ```
  pixi global install rattler-build
  ```

- git — Nécessaire pour cloner le dépôt d'échantillons. WindowsActivé, git for fournit Windows également un `bash` shell, ce dont certains Windows exemples de recettes ont besoin.
- Compartiment Amazon S3 : compartiment Amazon S3 à utiliser comme canal conda. Vous pouvez utiliser le compartiment des pièces jointes à des tâches de votre ferme Deadline Cloud ou créer un compartiment distinct.
- AWS informations d'identification — Configurez les informations d'identification sur votre poste de travail à l'aide de la `aws configure` commande ou de la `aws login` commande. Pour plus d'informations, consultez Configuration de AWS CLI dans le Guide de l'utilisateur AWS Command Line Interface .
- Autorisations IAM — (Facultatif) Pour réduire l'étendue des autorisations dont disposent vos informations d'identification, vous pouvez utiliser une politique Gestion des identités et des accès AWS (IAM) qui n'accorde que les autorisations suivantes sur le compartiment Amazon S3 et le préfixe de canal que vous utilisez (par exemple,) : `/Conda/*`
  - `s3:GetObject`

- `s3:PutObject`
- `s3:DeleteObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`

Publication d'un package sur un canal Amazon S3

À utiliser `rattler-build publish` avec une `s3://` cible pour publier un package sur votre canal conda Amazon S3. Si le canal n'existe pas dans le compartiment, `rattler-build` initialise le canal automatiquement. Avant de commencer, assurez-vous d'avoir rempli les prérequis.

L'exemple suivant publie l'exemple de recette Blender 4.5 à partir du référentiel d'échantillons de Deadline Cloud surGitHub. Vous pouvez remplacer une recette différente depuis le référentiel d'échantillons ou utiliser votre propre recette.

> **Note**
>
> Les applications volumineuses peuvent nécessiter des dizaines de Go d'espace disque libre pour l'archive source, les fichiers extraits et la sortie de compilation. Assurez-vous d'utiliser un disque avec suffisamment d'espace disponible pour la sortie de génération du package.

Pour publier un package sur un canal Amazon S3

1. Clonez le référentiel d'échantillons de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

2. Passez au répertoire `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Exécutez la commande suivante. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
Conda/Default --build-number=+1
```

Le `/Conda/Default` préfixe organise le canal au sein du bucket. Vous pouvez utiliser un préfixe différent, mais celui-ci doit être cohérent dans toutes les commandes et configurations de file d'attente qui font référence au canal.

> **À propos des numéros de version**
>
> L'`--build-number=+1`option sélectionne automatiquement le numéro de version suivant en fonction de ce qui existe déjà dans le canal de destination. La meilleure pratique consiste à ne jamais remplacer un package dans une chaîne. Créez toujours avec un nouveau numéro de version si le package aurait autrement le même nom de fichier. L'utilisation `--build-number=+1` permet d'atteindre cet objectif lorsque vous créez une chaîne de production ou une chaîne intermédiaire qui reflète la production.
> Si vous souhaitez contrôler directement le numéro de version, vous pouvez le définir avec une valeur spécifique telle que`--build-number=7`. Si vous omettez cette option, `rattler-build` utilise le numéro de version défini dans le `recipe.yaml` fichier.

Si la recette de votre package dépend de packages provenant d'un canal particulier, tel que conda-forge, ajoutez-le `-c conda-forge` à la commande.

Vous pouvez également publier un fichier de package que vous avez déjà créé, par exemple un `.conda` fichier issu d'une version locale. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Initialisation ou réindexation d'un canal

Lorsque vous `rattler-build publish` publiez un package, la commande initialise automatiquement le canal s'il n'existe pas déjà. Dans la plupart des cas, il n'est pas nécessaire d'initialiser ou de réindexer le canal manuellement.

Vous devrez peut-être initialiser ou réindexer manuellement un canal dans les situations suivantes :

- Vous souhaitez créer un canal vide avant de publier des packages, par exemple, pour vérifier que votre environnement de file d'attente Deadline Cloud peut se connecter au canal.

- Vous avez chargé ou supprimé `.conda` des fichiers directement avec les outils Amazon S3 au lieu de les utiliser`rattler-build publish`, et l'index des chaînes est obsolète.

Initialisation d'un canal vide

Pour initialiser un canal vide, créez un `repodata.json` fichier et téléchargez-le dans le `noarch` sous-répertoire du préfixe du canal. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
echo '{"info":{"subdir":"noarch"},"packages":{},"packages.conda":{},"removed":
[],"repodata_version":1}' > empty_channel_repodata.json
aws s3api put-object --body empty_channel_repodata.json --key Conda/Default/noarch/
repodata.json --bucket amzn-s3-demo-bucket
```

Le `/Conda/Default` préfixe doit correspondre au préfixe de canal utilisé par votre environnement de file d'attente. Après avoir initialisé le canal, vous pouvez publier des packages sur le canal en utilisant`rattler-build publish`.

Réindexation d'une chaîne

Si l'index du canal est obsolète, utilisez-le `rattler-index` pour le reconstruire à partir des fichiers de package du canal. Tout d'abord, installez`rattler-index`.

```
pixi global install rattler-index
```

Réindexez ensuite le canal. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

Tester le package

Après avoir publié le package, créez un projet pixi temporaire pour vérifier que le package fonctionne correctement. Le projet installe le package depuis le canal Amazon S3.

Pour tester le package

1. Créez un répertoire de test temporaire et initialisez un projet pixi avec le canal Amazon S3. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
mkdir package-test-env
cd package-test-env
pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
```

2. Ajoutez le package au projet.

```
pixi add blender=4.5
```

3. Vérifiez que le package fonctionne correctement.

```
pixi run blender --version
```

La `pixi run`commande active l'environnement conda pour le répertoire du projet et exécute la commande spécifiée dans celui-ci. L'environnement est conservé dans le répertoire du projet, vous pouvez donc utiliser la même `pixi run` commande depuis d'autres terminaux.

Supprimer des packages de la chaîne

Évitez de supprimer des packages des canaux que vous utilisez pour la production, car les fichiers de verrouillage font référence à des packages spécifiques par hachage. La suppression d'un package empêche de recréer des environnements à partir de ces fichiers de verrouillage. Pour les canaux de développement et de test, vous pouvez supprimer un package spécifique en supprimant le `.conda` fichier du bucket, puis en réindexant le canal.

Supprimez le fichier du package, puis réindexez le canal. *`amzn-s3-demo-bucket`*Remplacez-le par le nom de votre compartiment.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

Après avoir supprimé le fichier, réindexez le canal pour mettre à jour les métadonnées du canal. Pour obtenir des instructions, consultez la section Réindexation d'une chaîne.

Les fichiers de package sont stockés dans des sous-répertoires spécifiques à la plate-forme tels `linux-64` que, ou. `win-64` `osx-arm64` Pour répertorier les packages dans un sous-répertoire, exécutez la commande suivante.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

Nettoyage

Après le test, supprimez le répertoire du projet de test.

Pour nettoyer les ressources de test

- Supprimez le répertoire du projet de test.

  LinuxActivé macOS et exécutez la commande suivante.

  ```
  rm -rf package-test-env
  ```

  Sur Windows (cmd), exécutez la commande suivante.

  ```
  rmdir /s /q package-test-env
  ```

Débogage des versions

En cas d'échec d'une compilation, `rattler-build` préserve le répertoire de compilation afin que vous puissiez l'examiner. Exécutez la commande suivante pour ouvrir un shell interactif dans l'environnement de construction avec toutes les variables d'environnement configurées telles qu'elles étaient lors de la génération.

```
rattler-build debug shell
```

À partir du shell de débogage, vous pouvez modifier des fichiers, exécuter des commandes de compilation individuelles et ajouter des dépendances pour isoler le problème. Pour plus d'informations, consultez la section Débogage des versions dans la documentation de Rattler-build.

Création de packages pour d'autres plateformes

La `rattler-build publish` commande crée des packages pour le système d'exploitation du poste de travail sur lequel elle est exécutée. Si votre flotte Deadline Cloud utilise un système d'exploitation différent de celui de votre poste de travail, ou si votre package a d'autres exigences en matière d'hôte, vous disposez des options suivantes :

- Exécutez `rattler-build publish` sur un hôte correspondant au système d'exploitation cible. Par exemple, utilisez une instance Amazon Elastic Compute Cloud (Amazon EC2) Linux exécutée pour créer des packages pour une flotte. Linux

- Utilisez une file d'attente de création de packages Deadline Cloud pour automatiser les builds sur la plateforme cible. Voir Création d'une file d'attente de création de packages.
- (Avancé) Utilisez la compilation croisée pour créer des packages pour une plate-forme différente de celle de votre poste de travail. Pour plus d'informations, consultez la section Compilation croisée dans la documentation de Rattler-build.

Étapes suivantes

Après avoir publié des packages sur votre canal Amazon S3 Conda, configurez vos files d'attente Deadline Cloud pour utiliser le canal :

- Configurez les autorisations des files d'attente de production pour les packages conda personnalisés : accordez à vos files d'attente de production un accès en lecture seule au canal conda Amazon S3.
- Ajouter un canal conda à un environnement de file d'attente : configurez l'environnement de file d'attente pour installer des packages à partir du canal conda Amazon S3.

Configurer les autorisations de file d'attente de production pour les packages conda personnalisés

Votre file d'attente de production a besoin d'autorisations en lecture seule sur le `/Conda` préfixe du compartiment S3 de la file d'attente. Ouvrez la page Gestion des identités et des accès AWS (IAM) du rôle associé à la file d'attente de production et modifiez la politique comme suit :

1. Ouvrez la console Deadline Cloud et accédez à la page de détails de la file d'attente de création du package.
2. Choisissez le rôle du service de file d'attente, puis choisissez Modifier la file d'attente.
3. Accédez à la section Rôle du service de file d'attente, puis choisissez Afficher ce rôle dans la console IAM.
4. Dans la liste des politiques d'autorisation, choisissez celle qui convient AmazonDeadlineCloudQueuePolicyà votre file d'attente.
5. Dans l'onglet Autorisations, choisissez Modifier.
6. Ajoutez une nouvelle section au rôle de service de file d'attente comme suit. Remplacez *amzn-s3-demo-bucket* et *111122223333* par votre propre bucket et votre propre compte.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadOnly",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

Ajouter un canal conda à un environnement de file d'attente

Pour utiliser le canal conda S3, vous devez ajouter l'emplacement du `s3://amzn-s3-demo-bucket/Conda/Default` canal au `CondaChannels` paramètre des tâches que vous soumettez à Deadline Cloud. Les émetteurs fournis avec Deadline Cloud fournissent des champs pour spécifier les canaux et les packages Conda personnalisés.

Vous pouvez éviter de modifier chaque tâche en modifiant l'environnement de file d'attente conda pour votre file d'attente de production. Utilisez la procédure suivante.

1. Ouvrez la console Deadline Cloud et accédez à la page de détails de la file d'attente de production.
2. Choisissez l'onglet Environnements.
3. Sélectionnez l'environnement de file d'attente Conda, puis choisissez Modifier.
4. Choisissez l'éditeur JSON, puis dans le script, recherchez la définition du paramètre pour`CondaChannels`.
5. Modifiez la ligne `default: "deadline-cloud"` pour qu'elle commence par le canal conda S3 nouvellement créé :

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Les flottes gérées par les services offrent par défaut une priorité de canal flexible à Conda. Pour une tâche demandant `blender=4.5` si la Blender version 4.5 figure à la fois dans la nouvelle chaîne et dans la `deadline-cloud` chaîne, le package sera extrait de la chaîne figurant en premier dans la liste des chaînes. Si aucune version de package spécifiée n'est trouvée dans le premier canal, les canaux suivants seront vérifiés dans l'ordre correspondant à la version du package.

Pour les flottes gérées par le client, vous pouvez activer l'utilisation de packages conda en utilisant l'un des exemples d'environnement de file d'attente conda dans le référentiel d'exemples de Deadline Cloud. GitHub

Création d'un package conda pour une application ou un plugin

Un package conda est une archive compressée de logiciels écrits dans n'importe quelle langue. Conda prend en charge une variété de combinaisons de systèmes d'exploitation et d'architectures, de sorte que vous pouvez empaqueter des applications complètes telles que BlenderMaya, et Nuke parallèlement à des bibliothèques pour Python et d'autres langages. Pour plus d'informations sur les packages conda, consultez la section Packages dans la documentation conda.

Pour utiliser un package conda, vous devez l'installer dans un environnement virtuel. Un environnement virtuel conda possède un répertoire de préfixes dans lequel les packages sont installés. L'installation d'un package utilise la création de liens physiques ou de reconnexion de fichiers lorsqu'elle est prise en charge. La création de plusieurs environnements avec les mêmes packages n'utilise donc pas beaucoup d'espace disque supplémentaire. Pour utiliser un environnement virtuel, vous devez l'activer pour définir des variables d'environnement. L'activation exécute les scripts fournis par les packages, donnant à chaque package la possibilité de modifier le PATH ou d'autres variables d'environnement. Les packages Conda contiennent généralement des applications ou des bibliothèques, mais leur activation flexible signifie qu'ils peuvent également pointer vers des applications installées sur un système de fichiers partagé.

La création d'un package personnalisé comporte trois étapes : une recette contient les instructions de construction, un package est l'artefact (`.conda`ou `.tar.bz2` fichier) généré et un canal héberge les packages à installer. La `rattler-build publish` commande gère les trois étapes : elle peut intégrer une recette dans un package et la publier sur une chaîne, ou elle peut utiliser un artefact de package directement pour la publier.

La communauté conda-forge gère des recettes de packages pour un large éventail de logiciels open source et héberge des artefacts de packages dans la `conda-forge` chaîne. Vous pouvez configurer votre file d'attente pour l'inclure `conda-forge` en tant que source de package, puis créer des packages personnalisés qui dépendent de l'exécution des packages conda-forge. En effetLinux, conda-forge héberge une chaîne d'outils de compilation complète, y compris le support CUDA, avec des options de compilation et de liaison cohérentes sélectionnées. Vous pouvez utiliser les packages conda-forge comme dépendances dans vos propres recettes ou les installer avec vos packages personnalisés dans le même environnement.

Vous pouvez combiner une application entière, y compris les dépendances, dans un package conda. Les packages fournis par Deadline Cloud dans le canal Deadline-Cloud pour les flottes gérées par des services utilisent cette approche de reconditionnement binaire. Cela permet d'organiser les mêmes fichiers qu'une installation pour les adapter à l'environnement virtuel de Conda.

> **Note**
>
> Les applications volumineuses peuvent nécessiter des dizaines de Go d'espace disque libre pour l'archive source, les fichiers extraits et la sortie de compilation. Assurez-vous d'utiliser un disque avec suffisamment d'espace disponible pour la sortie de génération du package.

Package d'une application

Lorsque vous reconditionnez une application pour conda, vous avez deux objectifs :

- La plupart des fichiers de l'application doivent être séparés de la structure principale de l'environnement virtuel Conda. Les environnements peuvent ensuite mélanger l'application avec des packages provenant d'autres sources telles que conda-forge.
- Lorsqu'un environnement virtuel conda est activé, l'application doit être disponible à partir de la variable d'environnement PATH.

Pour reconditionner une application pour conda

1. Écrivez des recettes de construction conda qui installent l'application dans un sous-répertoire tel que. `$CONDA_PREFIX/opt/<application-name>` Cela le sépare des répertoires de préfixes standard tels que `bin` et`lib`.
2. Ajoutez des liens symboliques ou lancez des scripts `$CONDA_PREFIX/bin` pour exécuter les fichiers binaires de l'application.

Vous pouvez également créer des scripts .d activés que la `conda activate` commande exécutera pour ajouter les répertoires binaires de l'application au PATH. SiWindows, lorsque les liens symboliques ne sont pas pris en charge partout où des environnements peuvent être créés, utilisez plutôt des scripts de lancement ou d'activation d'applications.

3. Certaines applications dépendent de bibliothèques qui ne sont pas installées par défaut sur les flottes gérées par le service Deadline Cloud. Par exemple, le système de fenêtres X11 n'est généralement pas nécessaire pour les tâches non interactives, mais certaines applications nécessitent tout de même qu'il s'exécute sans interface graphique. Vous devez fournir ces dépendances dans le package que vous créez.
4. Si l'application prend en charge les plug-ins, indiquez une convention claire que les packages de plugins doivent suivre pour s'intégrer à l'application dans un environnement virtuel. Par exemple, l'exemple de recette Maya 2026 documente cette convention pour les Maya plugins.
5. Assurez-vous de respecter les droits d'auteur et les contrats de licence pour les applications que vous créez. Nous vous recommandons d'utiliser un compartiment Amazon S3 privé pour votre canal Conda afin de contrôler la distribution et de limiter l'accès aux packages à votre ferme.

Des exemples de recettes pour les packages de la `deadline-cloud` chaîne sont disponibles dans le référentiel d'exemples de Deadline Cloud surGitHub.

Package d'un plugin

Les plugins d'application peuvent être empaquetés sous la forme de leurs propres packages conda. Lorsque vous créez un package de plug-in, suivez les instructions suivantes :

- Incluez le package de l'application hôte en tant que dépendance de compilation et d'exécution dans la recette de générationrecipe.yaml. Utilisez une contrainte de version afin que la recette de construction ne soit installée qu'avec des packages compatibles.
- Respectez les conventions du package de l'application hôte pour enregistrer le plug-in.

Ensembles d'adaptateurs

Certaines intégrations d'applications Deadline Cloud utilisent un adaptateur qui étend l'interface de l'application afin de simplifier la rédaction de modèles de tâches. Un adaptateur est une interface de ligne de commande qui permet d'exécuter un démon en arrière-plan, de signaler l'état et d'appliquer le mappage de chemins. Pour plus d'informations, reportez-vous à la section Open Job Description

Adaptor Runtime surGitHub. Par exemple, deadline-cloud-for-mayaon GitHub inclut une interface graphique intégrée de soumission de tâches et un Maya adaptateur disponible sous forme de `maya-openjd` package sur les flottes gérées par des services.

Les soumissions de tâches depuis Deadline Cloud Submitter GUIs incluent une valeur de `CondaPackages` paramètre qui spécifie les packages conda à inclure dans un environnement virtuel pour exécuter la tâche. La valeur du `CondaPackages` paramètre pour ressemble Maya généralement à des entrées alternatives pour les packages de plugins `maya=2026.* maya-openjd=0.15.* maya-mtoa` et peut contenir ces entrées. Lorsque l'environnement de file d'attente configure un environnement virtuel conda pour exécuter le travail, il résout ces noms de packages et ces contraintes de version pour qu'ils soient compatibles et ajoute tous les packages de dépendance dont ils ont besoin pour s'exécuter. Chaque package d'adaptateur et de plugin spécifie ce avec quoi il est compatible, y compris les versions de PythonMaya, les versions de Python et les autres dépendances.

Pour créer vos propres packages adaptateurs à l'aide de nos exemples tels que la recette maya-openjd onGitHub, vous pouvez utiliser les packages pour Python et d'autres dépendances fournis par conda-forge. Vous devrez peut-être d'abord créer la date limite et les openjd-adaptor-runtimerecettes pour satisfaire les dépendances.

Créez une recette de construction de conda pour Blender

Blenderest gratuit et simple à empaqueter avec conda, ce qui en fait un bon point de départ pour apprendre à créer des packages conda pour AWS Deadline Cloud (Deadline Cloud). La Blender Fondation fournit des archives d'applications pour plusieurs systèmes d'exploitation. L'exemple de recette Blender 4.5 contenu dans le référentiel d'échantillons de Deadline Cloud GitHub permet de regrouper ces archives dans un package conda.

Comprendre la recette

Le fichier recipe.yaml définit les métadonnées, la source URLs et les options de compilation du package dans la syntaxe du modèle Rattler-build. La recette spécifie le numéro de version une seule fois et fournit une source différente URLs en fonction du système d'exploitation.

La `build` section `recipe.yaml` désactive la relocalisation binaire et les vérifications de liaison d'objets partagés dynamiques (DSO). Ces options contrôlent le fonctionnement du package lorsqu'il est installé dans un environnement virtuel Conda, quel que soit le préfixe de répertoire. Les valeurs par défaut utilisées dans `build` cette section sont conçues pour empaqueter chaque bibliothèque de

dépendances séparément, mais lors du reconditionnement binaire d'une application, vous devez les modifier. Blenderne nécessite aucun ajustement RPATH car les archives de l'application sont créées dans un souci de relocalisation. Voir Créer une recette de conda pour Maya pour un exemple d'ajout de relocatabilité.

Lors de la création du package, le script build.sh ou build_win.sh s'exécute pour installer les fichiers dans l'environnement. Ces scripts copient les fichiers d'installation dans`$PREFIX/opt/blender`, créent des liens symboliques à partir de `$PREFIX/bin` (onLinux) et configurent des scripts d'activation qui configurent des variables d'environnement telles que`BLENDER_LOCATION`. ActivéWindows, le script d'activation ajoute le Blender répertoire au PATH au lieu de créer des liens symboliques.

Le script de Windows construction utilise un fichier .bat à la `bash` place d'un fichier `cmd.exe` .bat pour assurer la cohérence entre les plateformes. Vous pouvez installer git for `bash` pour Windows permettre la création de paquets.

La recette inclut également un `deadline-cloud.yaml` fichier qui spécifie les plateformes conda et les métadonnées permettant de soumettre des tâches de création de packages automatisées à Deadline Cloud. Pour plus d'informations, voir Soumettre une tâche de création de package.

Création du Blender package

`rattler-build publish`À utiliser pour créer la recette Blender 4.5 et publier le package sur une chaîne. Vous pouvez publier sur un canal de système de fichiers local à des fins de test ou directement sur un canal Amazon S3 pour une utilisation en production. Si vous avez terminé la configuration dans Construire et tester les packages localement, exécutez la commande suivante depuis le `conda_recipes` répertoire.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Pour les autres options de publication :

- Pour publier sur un canal Amazon S3, consultez Publier des packages sur un canal conda S3.
- Pour automatiser les builds à l'aide d'une file d'attente de création de packages Deadline Cloud, voir Automatiser les builds de packages avec Deadline Cloud.

Testez votre package avec une tâche Blender de rendu

Après avoir créé le package Blender 4.5, vous pouvez le tester avec une tâche de rendu. Si vous n'avez pas de Blender scène, téléchargez la scène Blender 3.5 - Cozy Kitchen depuis la page des fichiers de Blender démonstration. Le référentiel d'exemples de Deadline Cloud contient un ensemble de `blender_render` tâches et un environnement de file d'attente conda que vous pouvez utiliser pour les tests locaux et dans le cloud.

Réalisation de tests locaux

Vous pouvez exécuter le modèle de tâche sur votre poste de travail à l'aide de la CLI Open Job Description. Installez la CLI avec`pip`.

```
pip install openjd-cli
```

À partir du `job_bundles` répertoire du référentiel d'échantillons, exécutez la commande suivante. Remplacez */path/to/scene.blend* par le chemin d'accès à votre fichier de Blender scène.

```
openjd run blender_render/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages=blender=4.5 \
    -p CondaChannels=file://$HOME/my-conda-channel \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

L'`--environment`option applique l'environnement de file d'attente conda, qui crée un environnement virtuel conda avec les packages spécifiés dans. `CondaPackages` Le `CondaChannels` paramètre indique à l'environnement de file d'attente où se trouvent les packages. Si vous avez publié sur un canal Amazon S3 plutôt que sur un canal local, remplacez le `file://` chemin par l'URL de votre `s3://` canal.

Tests sur Deadline Cloud

Après avoir configuré votre file d'attente de production pour utiliser le canal conda d'Amazon S3, vous pouvez envoyer la tâche de rendu à Deadline Cloud. À partir du `job_bundles` répertoire du référentiel d'échantillons, exécutez la commande suivante.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
```

```
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Utilisez le moniteur Deadline Cloud pour suivre l'avancement du travail. Dans le moniteur, sélectionnez la tâche correspondant à la tâche et choisissez Afficher les journaux. Sélectionnez l'action Lancer la session Conda pour vérifier que le package a été trouvé dans le canal Amazon S3.

Créez une recette de construction de conda pour Autodesk Maya

Les applications commerciales telles que. Autodesk Maya introduisent des exigences d'emballage supplémentaires par rapport aux applications open source telles queBlender. La Blenderrecette contient une simple archive relocalisable sous une licence open source. Les applications commerciales sont souvent distribuées par le biais d'installateurs et nécessitent une configuration de gestion des licences.

Considérations relatives aux applications commerciales

Les considérations suivantes s'appliquent lors de l'emballage d'applications commerciales. Les détails illustrent comment chacun s'applique àMaya.

- Licences : comprenez les droits de licence et les restrictions de l'application. Il se peut que vous deviez configurer un système de gestion des licences. Lisez la FAQ sur les avantages de l'Autodeskabonnement concernant les droits liés au cloud pour comprendre les droits liés au cloud pourMaya. Autodeskles produits s'appuient sur un `ProductInformation.pit` fichier dont la configuration nécessite généralement l'accès d'un administrateur. Les fonctionnalités du produit destinées aux clients légers constituent une alternative délocalisable. Consultez Thin Client Licensing pour Maya et MotionBuilder pour plus d'informations.
- Dépendances des bibliothèques système : certaines applications dépendent de bibliothèques qui ne sont pas installées sur des hôtes de parc gérés par des services. Mayadépend de bibliothèques telles que freetype et fontconfig. Lorsque ces bibliothèques sont disponibles dans le gestionnaire de packages système, par exemple `dnf` for AL2023, vous pouvez utiliser le gestionnaire de packages comme source. Comme les packages RPM ne sont pas conçus pour être relocalisables, vous devez utiliser des outils tels que `patchelf` la résolution des dépendances au sein du préfixe Maya d'installation.
- Accès administrateur pour l'installation : certains programmes d'installation nécessitent un accès administrateur. Les flottes gérées par les services ne fournissent pas d'accès administrateur. Vous devez donc installer l'application sur un système distinct et créer une archive des fichiers pour la

création du package. Le Windows programme d'installation de Maya nécessite cette approche. Le fichier README.md de la recette décrit une procédure reproductible utilisant une instance Amazon Elastic Compute Cloud (Amazon EC2) récemment lancée.

- Intégration de plugins — L'exemple de Maya package définit `MAYA_NO_HOME=1` pour isoler l'application de la configuration au niveau de l'utilisateur et ajoute des chemins de recherche de modules `MAYA_MODULE_PATH` afin que les packages de plugins puissent placer `.mod` des fichiers dans l'environnement virtuel. Consultez l'exemple de recette Maya 2026 pour connaître la convention complète d'intégration des plugins.

Comprendre la recette

Le fichier recipe.yaml définit les métadonnées du package dans la syntaxe du modèle Rattler-build. Passez en revue les sections suivantes du fichier :

- source — Fait référence aux archives du programme d'installation, y compris le hachage sha256. ActivéLinux, la source est l'archive du Autodesk programme d'installation. ActivéWindows, la source inclut à la fois l'archive du programme d'installation et un `cleanMayaForCloud.py` script Autodesk qui Maya prépare le déploiement dans le cloud. Mettez à jour les hachages lorsque vous modifiez les fichiers source, par exemple lors de l'empaquetage d'une nouvelle version.
- build — Désactive les vérifications de relocalisation binaire et de liaison DSO par défaut car les mécanismes automatiques ne fonctionnent pas correctement pour la bibliothèque et les répertoires binaires qui les Maya utilisent. ActivéLinux, la recette inclut `patchelf` en tant que dépendance de construction pour définir manuellement la valeur relative RPATHs.
- about — Métadonnées relatives à l'application permettant de parcourir ou de traiter le contenu d'un canal conda.

Les scripts de compilation (build.sh pourLinux, build_win.sh pourWindows) incluent des commentaires expliquant chaque étape. Les scripts exécutent les tâches clés suivantes :

- Extraire le programme d'installation — Extrait les fichiers Maya d'installation dans le préfixe conda. Les Windows scripts Linux and gèrent cela différemment en raison des formats d'installation. Consultez les scripts de compilation pour plus de détails.
- Installer les dépendances des bibliothèques système : activéLinux, le script télécharge et extrait les bibliothèques système Maya nécessaires mais absentes sur les hôtes de flotte gérés par des services. Le script copie ces bibliothèques dans le Maya `lib` répertoire afin qu'elles soient disponibles dans l'environnement conda.

- Définir relatif RPATHs avec patchelf — ActivéLinux, le script permet `patchelf --add-rpath` d'ajouter des chemins `$ORIGIN` -relatifs aux bibliothèques partagées. Cette approche suit la recommandation de conda de ne jamais utiliser `LD_LIBRARY_PATH` dans des environnements conda. Le script applique des correctifs aux bibliothèques à plusieurs niveaux de répertoire (`liblib/python*/site-packages`,,`lib/python*/lib-dynload`) afin que chaque bibliothèque puisse trouver ses dépendances par rapport à son propre emplacement. La recette suit la meilleure pratique qui consiste à définir `DT_RUNPATH` au lieu de`DT_RPATH`, ce qui permet de `LD_LIBRARY_PATH` remplacer le chemin de recherche lorsque cela est nécessaire pour le débogage.
- Configurer les licences pour les clients légers : le script configure les licences pour les clients légers, comme indiqué par la documentation, Autodesk afin que le `ProductInformation.pit` fichier puisse être localisé dans l'environnement conda au lieu de nécessiter un accès administrateur au niveau du système.
- Configurer des scripts d'activation : les scripts créent des scripts d'activation et de désactivation qui définissent des variables d'environnement telles que `MAYA_LOCATION`MAYA_VERSION,`MAYA_NO_HOME`, et`MAYA_MODULE_PATH`. ActivéWindows, les scripts produisent à la fois `.sh` des fichiers `.bat` d'activation, car les environnements de file d'attente d'exemple de Deadline Cloud sont utilisés `bash` pour activer les environnements surWindows.

Création du Maya package

Avant de créer le Maya package, téléchargez le Maya programme d'installation depuis votre Autodesk compte. Pour Linux cela, placez l'archive directement dans le `conda_recipes/archive_files` répertoire. Pour créer l'archiveWindows, suivez la procédure décrite dans le fichier README.md.

`rattler-build publish`À utiliser pour créer et publier le package. La Maya recette nécessite `patchelf` en tant que build une dépendance àLinux, qui est disponible auprès de conda-forge. Ajoutez `-c conda-forge` pour rendre la dépendance disponible lors de la construction. Dans le `conda_recipes` répertoire, exécutez la commande suivante.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Pour les autres options de publication :

- Pour publier sur un canal Amazon S3, consultez Publier des packages sur un canal conda S3.
- Pour automatiser les builds à l'aide d'une file d'attente de création de packages Deadline Cloud, voir Automatiser les builds de packages avec Deadline Cloud. Pour créer à la fois Linux des Windows packages et des packages, utilisez l'`--all-platforms`option associée au `submit-package-job` script.

Pour afficher l'échantillon de platine vinyle avec Maya etArnold, créez à la fois les packages du MtoAplugin et de l'Mayaadaptateur. Après avoir publié les trois packages, vous pouvez soumettre une tâche de rendu de test à l'aide de la table tournante avec le bundleMaya/Arnoldjob du référentiel d'échantillons de Deadline Cloud. Consultez la section Tester vos packages avec une tâche de rendu Maya.

Créez une recette de construction conda pour l'adaptateur Maya

Le `maya-openjd` package fournit l'adaptateur qui s'intègre Maya aux soumissions de tâches de AWS Deadline Cloud (Deadline Cloud). Lorsque vous soumettez une tâche de Maya rendu à l'aide d'une interface graphique de soumission de Deadline Cloud, le `CondaPackages` paramètre est inclus à `maya-openjd` côté du `maya` package. L'adaptateur gère le lancementMaya, la communication des paramètres de rendu et la gestion du cycle de vie de l'application pendant une session de travail. Pour plus d'informations sur les adaptateurs, consultez la section Packages d'adaptateurs.

Comprendre la recette

L'exemple de recette maya-openjd construit l'adaptateur à partir du paquet deadline-cloud-for-mayasource publié sur PyPI. Le fichier recipe.yaml installe le package en utilisant `pip` l'environnement conda.

La recette dépend de Python et de deux autres packages du référentiel d'échantillons de Deadline Cloud que vous devez d'abord créer :

- deadline — La bibliothèque cliente de Deadline Cloud.
- openjd-adaptor-runtime— Le moteur d'exécution de l'adaptateur Open Job Description.

Python et d'autres dépendances sont disponibles sur conda-forge, alors ajoutez-les `-c conda-forge` à la `rattler-build publish` commande lorsque vous créez le package adaptateur.

Création du package d'adaptateurs

Le `maya-openjd` package dépend de deux autres packages du référentiel d'échantillons de Deadline Cloud. Construisez les trois packages dans l'ordre à partir du `conda_recipes` répertoire. L'`-c conda-forge`option de chaque commande est de satisfaire les dépendances des recettes pour Python et les bibliothèques Python.

Créez le `deadline` package.

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Créez le `openjd-adaptor-runtime` package.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Créez le `maya-openjd` package.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Pour les autres options de publication :

- Pour publier sur un canal Amazon S3, consultez Publier des packages sur un canal conda S3.
- Pour automatiser les builds à l'aide d'une file d'attente de création de packages Deadline Cloud, voir Automatiser les builds de packages avec Deadline Cloud.

Créez une recette de construction conda pour le plugin Autodesk Maya to Arnold (MtoA)

Le Maya to Arnold (MtoA) plugin ajoute le Arnold moteur de rendu en tant qu'option. Maya L'exemple de recette MToA montre comment empaqueter un plugin sous la forme d'un package conda distinct qui s'intègre au package d'application hôte.

Comprendre la recette

Le fichier recipe.yaml spécifie une dépendance vis-à-vis `maya` du package pour les exigences de compilation et d'exécution. Cette dépendance utilise une contrainte de version afin que le plugin ne soit installé qu'avec une Maya version compatible.

La recette utilise les mêmes archives sources que la Maya recette. Le script de compilation installe MtoA et crée un `mtoa.mod` fichier dans le `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` répertoire dans lequel le Maya package est configuré. `MAYA_MODULE_PATH` Arnoldet Maya utilisez la même technologie de licence, de sorte que le Maya package inclut déjà les informations de licence Arnold nécessaires.

Création du MtoA package

Construisez le Maya package avant de le MtoA créer, car MtoA cela dépend du moment de Maya la construction. `rattler-build publish`À utiliser pour créer et publier le package. Dans le `conda_recipes` répertoire, exécutez la commande suivante.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

La `rattler-build publish` commande utilise le canal cible comme canal prioritaire lors de la résolution des dépendances, de sorte que le `maya` package que vous avez publié précédemment est automatiquement disponible.

Pour les autres options de publication :

- Pour publier sur un canal Amazon S3, consultez Publier des packages sur un canal conda S3.
- Pour automatiser les builds à l'aide d'une file d'attente de création de packages Deadline Cloud, voir Automatiser les builds de packages avec Deadline Cloud.

Testez vos packages avec une tâche Maya de rendu

Après avoir créé les maya-openjd packages,, etMaya,,,MtoA,,,,,,,,,,,,,,,,,,,,,,,, Le référentiel d'échantillons de Deadline Cloud contient une [plaque tournante avec le bundleMaya/Arnold]job qui affiche une animation à l'aide Maya de et. Arnold Le job bundle permet également FFmpeg d'encoder une vidéo, qui est disponible sur la `conda-forge` chaîne.

Réalisation de tests locaux

Vous pouvez exécuter le modèle de tâche sur votre poste de travail à l'aide de la [CLI Open Job Description]. Installez la CLI avec`pip`.

```
pip install openjd-cli
```

À partir du `job_bundles` répertoire du référentiel d'échantillons, exécutez la commande suivante. Le `ErrorOnArnoldLicenseFail=false` paramètre indique Arnold d'effectuer le rendu avec des filigranes au lieu d'échouer lorsqu'aucune licence n'est disponible.

```
openjd run turntable_with_maya_arnold/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages="maya maya-mtoa maya-openjd ffmpeg" \
    -p CondaChannels="file://$HOME/my-conda-channel conda-forge" \
    -p ErrorOnArnoldLicenseFail=false \
    -p FrameRange=1-5
```

L'`--environment`option applique l'environnement de file d'attente conda, qui crée un environnement virtuel conda avec les packages spécifiés dans. `CondaPackages` Le `CondaChannels` paramètre inclut à la fois le canal local pour vos packages personnalisés et `conda-forge` pour`ffmpeg`. Si vous avez publié sur un canal Amazon S3 plutôt que sur un canal local, remplacez le `file://` chemin par l'URL de votre `s3://` canal.

Lorsque le travail est terminé, la sortie rendue se trouve dans le `turntable_with_maya_arnold/output/` répertoire.

Tests sur Deadline Cloud

Après avoir configuré votre file d'attente de production pour utiliser le canal conda Amazon S3, soumettez la tâche de rendu à Deadline Cloud. Ajoutez le `conda-forge` canal au `CondaChannels` paramètre dans votre environnement de file d'attente Conda pour fournir une source `ffmpeg` et les

dépendances Python requises par l'adaptateur. À partir du `job_bundles` répertoire du référentiel d'échantillons, exécutez la commande suivante.

```
deadline bundle submit turntable_with_maya_arnold
```

Utilisez le moniteur Deadline Cloud pour suivre l'avancement du travail. Dans le moniteur, sélectionnez la tâche correspondant à la tâche et choisissez Afficher les journaux. Sélectionnez l'action Lancer la session Conda pour vérifier que les `maya-openjd` packages `mayamaya-mtoa`, et ont été trouvés dans le canal Amazon S3.

Automatisez la création de packages avec Deadline Cloud

Pour les CI/CD flux de travail ou lorsque vous devez créer des packages pour plusieurs systèmes d'exploitation, vous pouvez créer une file d'attente de création de packages Deadline Cloud. Les plannings des files d'attente créent des tâches sur votre flotte, qui crée les packages et les publie sur votre canal conda Amazon Simple Storage Service (Amazon S3). Cela simplifie la gestion continue des compilations de packages pour les versions logicielles dans toutes les configurations requises.

Vous pouvez créer une file d'attente de création de packages à l'aide d'un modèle AWS CloudFormation (CloudFormation) ou manuellement depuis la console Deadline Cloud. Le CloudFormation modèle déploie un parc complet avec une file d'attente de production et une file d'attente de création de packages déjà configurées. La création de la file d'attente depuis la console vous permet de mieux contrôler les paramètres individuels.

Créez une file d'attente de création de packages avec CloudFormation

Vous pouvez utiliser un CloudFormation modèle pour créer une ferme Deadline Cloud qui inclut une file d'attente pour la création de packages. Le modèle configure une file d'attente de production et une file d'attente de création de packages avec un canal conda privé Amazon S3.

Avant de déployer le modèle, créez un compartiment Amazon S3 contenant les pièces jointes aux tâches et votre canal Conda. Vous pouvez créer un compartiment à partir de la console Amazon S3. Vous avez besoin du nom du compartiment lorsque vous déployez le modèle.

Pour déployer le CloudFormation modèle

1. Téléchargez le modèle deadline-cloud-starter-farm-template.yaml depuis le référentiel d'exemples de Deadline Cloud sur. GitHub

2. Dans la CloudFormation console, choisissez Create Stack, puis Avec de nouvelles ressources (standard).
3. Sélectionnez l'option permettant de télécharger un fichier modèle, puis téléversez le `deadline-cloud-starter-farm-template.yaml` fichier.
4. Entrez un nom pour la pile, par exemple**`StarterFarm`**, et fournissez le nom d'un compartiment Amazon S3 pour les pièces jointes aux tâches et le canal conda.
5. Suivez les étapes de CloudFormation la console pour terminer la création de la pile.

Pour plus d'informations sur les paramètres du modèle et les options de personnalisation, consultez le fichier README de la ferme de démarrage dans le référentiel d'exemples de Deadline Cloud surGitHub.

Création d'une file d'attente pour la création de packages à partir de la console

Suivez les instructions de la section Création d'une file d'attente dans le guide de l'utilisateur de Deadline Cloud. Effectuez les modifications suivantes :

- À l'étape 5, choisissez un compartiment Amazon S3 existant. Spécifiez un nom de dossier racine, de **`DeadlineCloudPackageBuild`** manière à ce que les artefacts de build restent séparés de vos pièces jointes habituelles de Deadline Cloud.
- À l'étape 6, vous pouvez associer la file d'attente de création de packages à une flotte existante, ou vous pouvez créer une toute nouvelle flotte si votre flotte actuelle n'est pas adaptée.
- À l'étape 9, créez un nouveau rôle de service pour votre file d'attente de création de packages. Vous allez modifier les autorisations pour donner à la file d'attente les autorisations requises pour télécharger des packages et réindexer un canal conda.

Configurer les autorisations de file d'attente de création du package

Pour permettre à la file d'attente de création du package d'accéder au `/Conda` préfixe du compartiment Amazon S3 de la file d'attente, vous devez modifier le rôle de la file d'attente pour lui donner read/write accès. Le rôle a besoin des autorisations suivantes pour que les tâches de création de packages puissent télécharger de nouveaux packages et réindexer le canal.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`

- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Ouvrez la console Deadline Cloud et accédez à la page de détails de la file d'attente de création du package.
2. Choisissez le rôle du service de file d'attente, puis choisissez Modifier la file d'attente.
3. Accédez à la section Rôle du service de file d'attente, puis choisissez Afficher ce rôle dans la console IAM.
4. Dans la liste des politiques d'autorisation, choisissez celle qui convient AmazonDeadlineCloudQueuePolicyà votre file d'attente.
5. Dans l'onglet Autorisations, choisissez Modifier.
6. Ajoutez une nouvelle section au rôle de service de file d'attente comme suit. Remplacez *amzn-s3-demo-bucket* et *111122223333* par votre propre bucket et votre propre compte.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadWrite",
   "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

Soumettre une tâche de création de package

Après avoir créé une file d'attente de création de packages et configuré les autorisations de file d'attente, vous pouvez soumettre des tâches pour créer des packages conda. Le `submit-package-job` script figurant dans le référentiel d'exemples de Deadline Cloud GitHub soumet une tâche de génération pour une recette de conda.

Vous avez besoin des éléments suivants :

- La CLI Deadline Cloud installée sur votre poste de travail.
- Une session de connexion active AWS au moniteur Deadline Cloud (moniteur Deadline Cloud).
- Un clone du référentiel d'échantillons de Deadline Cloud.

Pour soumettre une tâche de création de package

1. Ouvrez l'interface graphique de configuration de Deadline Cloud et définissez le parc et la file d'attente par défaut sur votre file d'attente de création de packages.

   ```
   deadline config gui
   ```

2. Accédez au `conda_recipes` répertoire du référentiel d'échantillons.

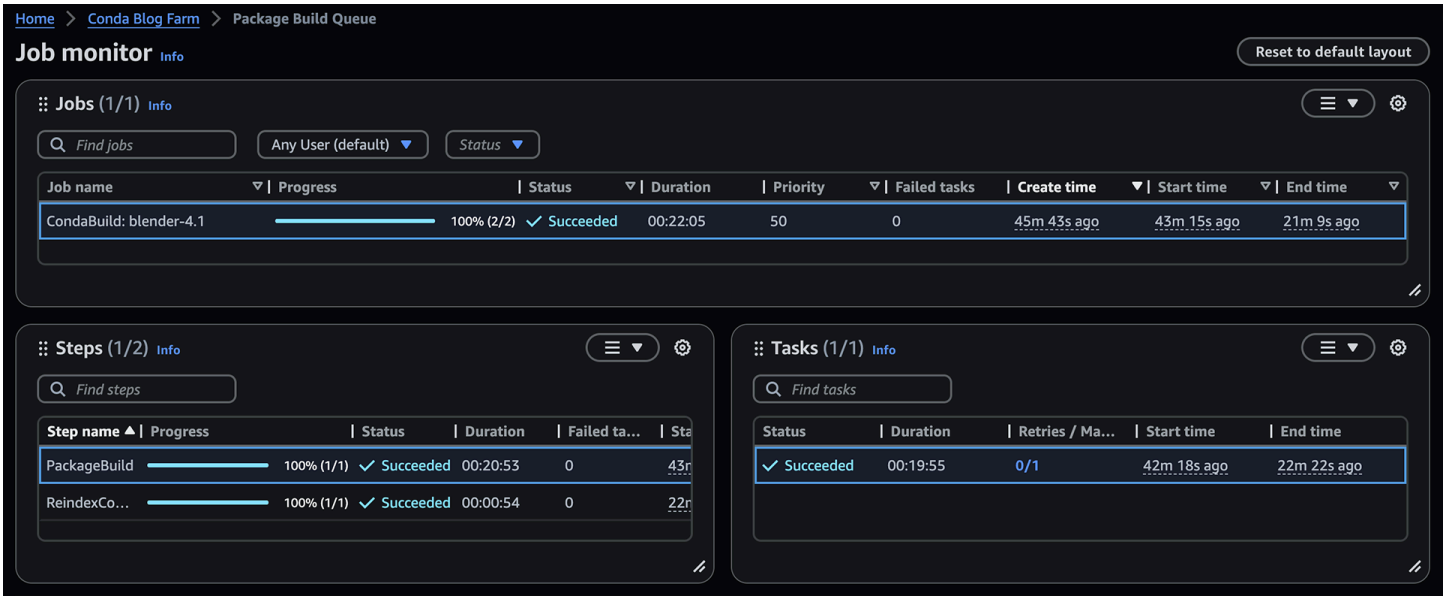
   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Exécutez le `submit-package-job` script avec le répertoire des recettes. L'exemple suivant crée la recette Blender 4.5.

   ```
   ./submit-package-job blender-4.5/
   ```

   Si la recette nécessite une archive source que vous n'avez pas encore téléchargée, le script fournit des instructions de téléchargement. Téléchargez l'archive et réexécutez le script.

Après avoir soumis la tâche, utilisez le moniteur Deadline Cloud pour voir la progression et le statut de la tâche.



Le moniteur montre les deux étapes de la tâche : créer le package, puis réindexer le canal Conda. Lorsque vous cliquez avec le bouton droit sur la tâche correspondant à l'étape de création du package et que vous choisissez Afficher les journaux, le moniteur affiche les actions de session :

- Synchroniser les pièces jointes : copie les pièces jointes des tâches d'entrée ou monte un système de fichiers virtuel.
- Lancez Conda — L'action relative à l'environnement de file d'attente. La tâche de construction ne spécifie pas les packages conda, cette action se termine donc rapidement.
- Launch CondaBuild Env — Crée un environnement virtuel conda avec le logiciel nécessaire pour créer un package conda et réindexer un canal.
- Exécution de la tâche — Construit le package et télécharge les résultats sur Amazon S3.

Au fur et à mesure que les actions s'exécutent, elles envoient des journaux à Amazon CloudWatch (CloudWatch). Lorsqu'une tâche est terminée, sélectionnez Afficher les journaux de toutes les tâches pour voir les journaux supplémentaires relatifs à la configuration et au démontage de l'environnement.

Exécuter des scripts de configuration de l'hôte avec des privilèges d'administrateur

Les scripts de configuration de l'hôte vous permettent d'effectuer des tâches administratives, telles que l'installation de logiciels, auprès des employés de votre parc géré par des services. Ces scripts

s'exécutent avec des privilèges élevés (`sudo`activéLinux, administrateur activéWindows), ce qui vous donne la flexibilité de configurer vos travailleurs pour votre système.

Deadline Cloud exécute le script une fois que le travailleur est entré dans l'`STARTING`état et avant qu'il n'exécute les tâches.

> **Important**
>
> Le script s'exécute avec des autorisations élevées. Il est de votre responsabilité de vous assurer que le script ne présente aucun problème de sécurité.
> Lorsque vous utilisez un script de configuration hôte, vous êtes chargé de surveiller l'état de santé de votre flotte.

Les utilisations courantes des scripts de configuration d'hôte sont les suivantes :

- Installation de logiciels nécessitant l'accès d'un administrateur
- Installation de Docker conteneurs
- Installation de solutions de stockage cloud tierces telles queLucidLink. Pour une présentation détaillée, voir Configurer LucidLink avec des scripts de flotte gérés par des services pour Deadline Cloud sur le blog AWS for M&E.

Vous pouvez créer et mettre à jour un script de configuration d'hôte à l'aide de la console ou du AWS CLI.

Console

1. Sur la page des détails de la flotte, choisissez l'onglet Configurations.
2. Dans le champ Script, entrez le script à exécuter avec des autorisations élevées. Vous pouvez choisir Importer pour charger un script depuis votre poste de travail.
3. Définissez un délai d'expiration en secondes pour exécuter le script. La valeur par défaut est de 300 secondes (5 minutes).
4. Choisissez Enregistrer les modifications pour enregistrer le script.

Create with CLI

Utilisez la AWS CLI commande suivante pour créer un parc à l'aide d'un script de configuration d'hôte. Remplacez le *placeholder* texte par vos informations.

```
aws deadline create-fleet \
--farm-id farm-12345 \
--display-name "fleet-name" \
--max-worker-count 1 \
--configuration '{
"serviceManagedEc2": {
  "instanceCapabilities": {
    "vCpuCount": {"min": 2},
    "memoryMiB": {"min": 4096},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
  },
  "instanceMarketOptions": {"type":"spot"}
}
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Update with CLI

Utilisez la AWS CLI commande suivante pour mettre à jour le script de configuration de l'hôte d'une flotte. Remplacez le *placeholder* texte par vos informations.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Les scripts suivants illustrent ce qui suit :

- Les variables d'environnement disponibles pour le script
- Ces AWS informations d'identification fonctionnent dans le shell
- Que le script s'exécute dans un shell surélevé

Linux

Utilisez le script suivant pour montrer qu'un script s'exécute avec des `root` privilèges :

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Utilisez le PowerShell script suivant pour montrer qu'un script est exécuté avec des privilèges d'administrateur :

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
  $currentUser = New-Object
 Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
  $isAdmin =
 $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

  return $isAdmin
}

if (Test-AdminPrivileges) {
  Write-Host "The current PowerShell session is elevated (running as
 Administrator)."
} else {
  Write-Host "The current PowerShell session is not elevated (not running as
 Administrator)."
}
exit 0
```

Résolution des problèmes liés aux scripts de configuration de l'hôte

Lorsque vous exécutez le script de configuration de l'hôte :

- En cas de succès : le travailleur exécute le travail
- En cas d'échec (code de sortie différent de zéro ou crash) :
  - Le travailleur s'arrête

La flotte lance automatiquement un nouveau travailleur à l'aide du dernier script de configuration de l'hôte

Pour surveiller le script, procédez comme suit :

1. Ouvrez la page du parc dans la console Deadline Cloud.
2. Choisissez View workers pour ouvrir le moniteur Deadline Cloud.
3. Consultez le statut du travailleur sur la page du moniteur.

> **Tip**
>
> Lorsque vous testez des scripts de configuration de l'hôte, définissez le nombre maximum de travailleurs du parc sur 1 afin d'éviter de démarrer plusieurs travailleurs pendant l'itération du script.

Remarques importantes :

- Les travailleurs qui s'arrêtent en raison d'une erreur ne figurent pas dans la liste des travailleurs du moniteur. Utilisez CloudWatch Logs pour afficher les journaux des travailleurs dans le groupe de journaux suivant :

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  Dans ce groupe de journaux, recherchez un flux nomméworker-*ZZZZZ*.
- CloudWatch Logs conserve les journaux des employés conformément à la période de conservation que vous avez configurée.

Surveiller l'exécution du script de configuration de l'hôte

Avec les scripts de configuration de l'hôte, vous pouvez prendre le contrôle total d'un travailleur de Deadline Cloud. Vous pouvez installer n'importe quel package logiciel, reconfigurer les paramètres du système d'exploitation ou monter des systèmes de fichiers partagés. Grâce à cette fonctionnalité avancée et à la capacité de Deadline Cloud à s'adapter à des milliers de travailleurs, vous pouvez surveiller si les scripts de configuration sont exécutés avec succès ou ont échoué.

Nous recommandons les solutions suivantes pour surveiller l'exécution des scripts de configuration de l'hôte.

CloudWatch Surveillance des journaux

Tous les journaux de configuration des hôtes du parc sont transmis au groupe de CloudWatch journaux du parc, et plus particulièrement au flux de CloudWatch journaux d'un travailleur. Par exemple, `/aws/deadline/farm-123456789012/fleet-777788889999` est le groupe de journaux pour la ferme `123456789012` et la flotte`777788889999`.

Chaque collaborateur fournit un flux de journal dédié, par exemple`worker-123456789012`. Les journaux de configuration de l'hôte incluent des bannières telles que Running Host Configuration Script et Finished running Host Configuration Script, code de sortie : 0. Le code de sortie du script est inclus dans la bannière finale et peut être demandé à l'aide CloudWatch d'outils.

CloudWatch Informations sur les journaux

CloudWatch Logs Insights propose des fonctionnalités avancées pour analyser les informations des journaux. Par exemple, la requête Log Insights suivante analyse le code de sortie de la configuration de l'hôte, trié par heure :

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

Pour plus d'informations sur CloudWatch Logs Insights, consultez Analyser les données des CloudWatch journaux avec Logs Insights dans le guide de l'utilisateur Amazon CloudWatch Logs.

Journalisation structurée par un agent de travail

L'agent de travail de Deadline Cloud publie des journaux JSON structurés sur CloudWatch.
Le Worker Agent propose un large éventail de journaux structurés pour analyser la santé des travailleurs. Pour plus d'informations, consultez la section Connexion de l'agent de travail à Deadline Cloud GitHub.

Les attributs des journaux structurés sont décompressés dans les champs de Log Insights. Vous pouvez utiliser cette CloudWatch fonctionnalité pour compter et analyser les échecs de démarrage de

la configuration de l'hôte. Par exemple, une requête count et bin peut être utilisée pour déterminer la fréquence des défaillances :

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

CloudWatch filtres métriques pour les métriques et les alarmes

Vous pouvez configurer CloudWatch des filtres de mesures pour générer des CloudWatch métriques à partir des journaux. Les filtres métriques vous permettent de créer des alarmes et des tableaux de bord pour surveiller l'exécution des scripts de configuration de l'hôte.

Pour créer un filtre de métrique

1. Ouvrez la CloudWatch console.
2. Dans le volet de navigation, choisissez Logs, puis Log groups.
3. Sélectionnez le groupe de journaux de votre flotte.
4. Choisissez Créer un filtre de métriques.
5. Définissez votre modèle de filtre à l'aide de l'une des méthodes suivantes :

   - Pour les indicateurs de réussite :

     ```
     {$.message = "*Worker Agent host configuration succeeded.*"}
     ```

   - Pour les mesures de défaillance :

     ```
     {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
      exit code*"}
     ```

6. Choisissez Next pour créer une métrique avec les valeurs suivantes :

   - Espace de noms métrique : votre espace de noms métrique (par exemple,) **`MyDeadlineFarm`**
   - Nom de la métrique : le nom de la métrique que vous avez demandé (par exemple,**`host_config_failure`**)
   - Valeur de la métrique : **`1`** (chaque instance est un compte de 1)
   - Valeur par défaut : Laisser vide
   - Unité : **`Count`**

Après avoir créé des filtres métriques, vous pouvez configurer des CloudWatch alarmes standard pour agir en cas de taux d'échec élevé de la configuration des hôtes, ou ajouter les métriques à un CloudWatch tableau de bord pour les day-to-day opérations et la surveillance.

Pour plus de détails, consultez la section Syntaxe des filtres et des modèles dans le guide de l'utilisateur Amazon CloudWatch Logs.

Utilisation de licences logicielles avec Deadline Cloud

Deadline Cloud propose deux méthodes pour fournir des licences logicielles pour vos tâches :

- Licences basées sur l'utilisation (UBL) : suivi et facturation en fonction du nombre d'heures que votre flotte utilise pour traiter une tâche. Il n'y a pas de nombre défini de licences, de sorte que votre flotte peut évoluer selon vos besoins. L'UBL est la norme pour les flottes gérées par des services. Pour les flottes gérées par le client, vous pouvez connecter un point de terminaison de licence Deadline Cloud pour UBL. UBL fournit des licences à vos employés de Deadline Cloud, mais pas de licences pour vos applications DCC.
- Bring your own license (BYOL) : vous permet d'utiliser les licences logicielles existantes avec vos flottes gérées par des services ou des clients. Vous pouvez utiliser BYOL pour vous connecter aux serveurs de licences pour les logiciels non pris en charge par les licences basées sur l'utilisation de Deadline Cloud. Vous pouvez utiliser BYOL avec des flottes gérées par des services en vous connectant à un serveur de licences personnalisé.

Rubriques

Combiner BYOL et UBL

Vous pouvez combiner le BYOL et l'UBL afin que vos employés utilisent d'abord vos licences existantes et reviennent automatiquement aux licences basées sur l'utilisation de Deadline Cloud lorsque vos licences BYOL sont épuisées. Cette approche est utile lorsque vous disposez d'un nombre limité de licences existantes, mais que vous devez étendre cette capacité pendant les pics de charge de travail.

Comment fonctionne le système de licences combinées

Lorsque vous configurez des licences combinées, l'environnement de file d'attente définit les variables d'environnement de licence afin que le serveur de licences BYOL soit répertorié avant le point de terminaison de licence UBL. La plupart des applications tierces vérifient les serveurs de licences dans l'ordre dans lequel ils apparaissent dans la variable d'environnement. Lorsqu'un

travailleur demande une licence, l'application contacte d'abord votre serveur de licences BYOL. Si aucune licence BYOL n'est disponible, l'application revient au point de terminaison de licence UBL.

Le modèle d'environnement de file d'attente BYOL fourni dans Connectez des flottes gérées par des services à un serveur de licences personnalisé configure automatiquement ce comportement de secours. Le script Python de l'environnement de file d'attente ajoute l'adresse de votre serveur de licences BYOL aux variables d'environnement de licence UBL existantes. Pour utiliser les licences combinées, conservez les sections UBL dans le script d'environnement de file d'attente pour les produits pour lesquels vous souhaitez un comportement de remplacement.

Pour utiliser uniquement le BYOL sans solution de secours UBL pour un produit spécifique, supprimez la section UBL de ce produit du script et ajoutez la variable d'environnement de licence directement dans la `variables` section de l'environnement de file d'attente. Par exemple, pour utiliser uniquement BYOL pour Cinema 4D, supprimez la section Cinema 4D du script et ajoutez-la `g_licenseServerRLM: 127.0.0.1:7057` à la `variables` section.

Exemple : utilisation de licences BYOL Cinema 4D avec UBL fallback

Prenons l'exemple d'un studio qui possède des licences Cinema 4D existantes sur un serveur de licences de son réseau local. Le studio souhaite utiliser ces licences pour le rendu sur Deadline Cloud, mais souhaite également dépasser le nombre de licences en revenant à UBL lorsque toutes les licences BYOL seront utilisées.

Pour configurer cette configuration, suivez les étapes décrites Connectez des flottes gérées par des services à un serveur de licences personnalisé et apportez les modifications suivantes au modèle d'environnement de file d'attente :

Pour configurer les licences BYOL Cinema 4D avec UBL fallback

1. Définissez le `LicenseInstanceId` paramètre sur l'ID d'instance Amazon Elastic Compute Cloud (Amazon EC2) du serveur de licences ou du proxy ayant accès au serveur de licences Cinema 4D.
2. Définissez le `LicensePorts` paramètre pour inclure le port `7057` (le port de licence RLM de Cinema 4D).
3. Dans le script Python, conservez la section Cinema 4D qui ajoute le serveur BYOL à la configuration UBL :

```
# Cinema4D
```

```
os.environ["g_licenseServerRLM"] = f"127.0.0.1:7057;
{os.environ.get('g_licenseServerRLM', '')}"
print(f"openjd_env: g_licenseServerRLM={os.environ['g_licenseServerRLM']}")
```

Cette configuration est définie `g_licenseServerRLM` sur`127.0.0.1:7057;`*`UBL_endpoint`*`:7057`. Cinema 4D vérifie d'`127.0.0.1:7057`abord le serveur BYOL. Si aucune licence n'est disponible, Cinema 4D revient au point de terminaison UBL.

4. Supprimez les sections relatives aux produits que vous n'utilisez pas (par exemple, Arnold, Nuke ou SideFX) pour que la configuration reste propre.

Si vous avez également d'autres produits qui utilisent uniquement le BYOL sans solution de repli UBL, ajoutez ces variables d'environnement de licence directement dans la `variables` section de l'environnement de file d'attente et supprimez les sections correspondantes du script Python.

Considérations relatives aux licences combinées

Tenez compte des considérations suivantes lorsque vous utilisez des licences combinées :

- Certaines applications ne prennent pas en charge plusieurs serveurs de licences dans une seule variable d'environnement. Par exemple, V-Ray utilise plutôt un fichier de configuration XML. Le modèle d'environnement de file d'attente gère la configuration V-Ray séparément. Pour plus d'informations, consultez la section V-Ray du modèle d'environnement de file d'attente dansConnectez des flottes gérées par des services à un serveur de licences personnalisé.
- L'ordre des serveurs de licences dans la variable d'environnement détermine la priorité. Répertoriez d'abord le serveur BYOL afin que vos licences existantes soient consommées avant les licences UBL.
- Sur Windows les serveurs de production, séparez les entrées du serveur de licences dans les variables d'environnement par un point-virgule (`;`) au lieu de deux points (). `:` Pour plus d'informations sur la configuration des variables d'environnement de licence, consultezConnectez les flottes gérées par le client à un point de terminaison de licence.
- Pour utiliser la solution de repli UBL avec des flottes gérées par le client, configurez un point de terminaison de licence. Pour de plus amples informations, veuillez consulter Connectez les flottes gérées par le client à un point de terminaison de licence.

Connectez des flottes gérées par des services à un serveur de licences personnalisé

Vous pouvez utiliser votre propre serveur de licences avec un parc géré par le service Deadline Cloud. Pour apporter votre propre licence, vous pouvez configurer un serveur de licences à l'aide d'un environnement de file d'attente dans votre parc de serveurs. Pour configurer votre serveur de licences, vous devez déjà avoir configuré une batterie de serveurs et une file d'attente.

La manière dont vous vous connectez à un serveur de licences logicielles dépend de la configuration de votre parc et des exigences du fournisseur du logiciel. Généralement, vous pouvez accéder au serveur de deux manières :

- Directement sur le serveur de licences. Vos employés obtiennent une licence auprès du serveur de licences du fournisseur de logiciels via Internet. Tous vos employés doivent être en mesure de se connecter au serveur.
- Par le biais d'un proxy de licence. Vos employés se connectent à un serveur proxy de votre réseau local. Seul le serveur proxy est autorisé à se connecter au serveur de licences du fournisseur via Internet.

En suivant les instructions ci-dessous, vous pouvez utiliser Amazon EC2 Systems Manager (SSM) pour transférer les ports d'une instance de travail vers votre serveur de licences ou votre instance proxy. Dans l'exemple ci-dessous, si votre serveur de licences n'est pas en mesure de fournir de licence, la licence basée sur l'utilisation de Deadline Cloud sera utilisée. Supprimez les sections qui ne s'appliquent pas à votre pipeline ou aux produits pour lesquels vous ne souhaitez pas utiliser les licences basées sur l'utilisation après avoir épuisé vos licences.

Rubriques

Étape 1 : Configuration de l'environnement de file d'attente

Vous pouvez configurer un environnement de file d'attente dans votre file d'attente pour accéder à votre serveur de licences. Tout d'abord, assurez-vous que vous disposez d'une AWS instance configurée avec un accès au serveur de licences à l'aide de l'une des méthodes suivantes :

- Serveur de licences : l'instance héberge directement les serveurs de licences.
- Proxy de licence : l'instance dispose d'un accès réseau au serveur de licences et transmet les ports du serveur de licences au serveur de licences. Pour plus de détails sur la configuration d'une instance de proxy de licence, consultezÉtape 2 : (Facultatif) Configuration de l'instance de proxy de licence.

Pour plus d'informations sur la configuration des variables d'environnement de licence, consultezÉtape 3 : Connecter une application de rendu à un point de terminaison. Pour une configuration de serveur de licences personnalisée, l'adresse du serveur de licences reste localhost au lieu du point de terminaison Amazon VPC.

Pour ajouter les autorisations requises au rôle de file d'attente

1. Dans la console Deadline Cloud, choisissez Accéder au tableau de bord.
2. Dans le tableau de bord, sélectionnez le parc, puis la file d'attente que vous souhaitez configurer.
3. Dans Détails de la file d'attente > rôle de service, sélectionnez le rôle.
4. Choisissez Ajouter une autorisation, puis choisissez Créer une politique intégrée.
5. Sélectionnez l'éditeur de politique JSON, puis copiez-collez le texte suivant dans l'éditeur.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
                "arn:aws:ec2:us-east-1:111122223333:instance/instance_id"
```

```
            ]
        }
    ]
}
```

6. Avant d'enregistrer la nouvelle politique, remplacez les valeurs suivantes dans le texte de la politique :
   - Remplacez `region` par la AWS région où se trouve votre ferme
   - Remplacez `instance_id` par l'ID d'instance du serveur de licences ou de l'instance proxy que vous utilisez
   - Remplacez `account_id` par le numéro de AWS compte contenant votre ferme
7. Choisissez Suivant.
8. Pour le nom de la politique, entrez**`LicenseForwarding`**.
9. Choisissez Créer une politique pour enregistrer vos modifications et créer la politique avec les autorisations requises.

Pour ajouter un nouvel environnement de file d'attente à la file

1. Dans la console Deadline Cloud, choisissez Accéder au tableau de bord si ce n'est pas déjà fait.
2. Dans le tableau de bord, sélectionnez le parc, puis la file d'attente que vous souhaitez configurer.
3. Choisissez Environnements de file d'attente > Actions > Créer un nouveau fichier avec YAML.
4. Copiez et collez le texte suivant dans l'éditeur de script YAML.

   Windows

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
```

```
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
windows/SessionManagerPlugin.zip" -o "{{Session.WorkingDirectory}}/ssm-
plugin.zip"
      powershell -Command "Expand-Archive -Path '{{Session.WorkingDirectory}}/
ssm-plugin.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin'
 -Force; Expand-Archive -Path '{{Session.WorkingDirectory}}/ssm-plugin/
package.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin/package'
 -Force"
      conda activate
      python "{{Env.File.StartSession}}" "{{Session.WorkingDirectory}}/ssm-
plugin/package/bin/session-manager-plugin.exe"
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
```

```
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
```

```
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost;
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Cinema4D
      os.environ["g_licenseServerRLM"] = f"localhost:7057;
{os.environ.get('g_licenseServerRLM', '')}"
      print(f"openjd_env:
 g_licenseServerRLM={os.environ['g_licenseServerRLM']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost;
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost;7055@localhost;
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
```

```
            <User></User>
            <Pass></Pass>
          </LicServer>
        </VRLClient>"""

        temp_dir = tempfile.gettempdir()
        xml_path = os.path.join(temp_dir, 'vrlclient.xml')

        with open(xml_path, 'w') as f:
            f.write(xml_content)

        os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
        print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

        # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
file is used.
        os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
        print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

        # Print out the created xml file's contents
        print(f"V-Ray configuration file: {xml_path}")
        with open(xml_path, 'r') as f:
            print(f"{f.read()}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
```

```
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
      chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
      conda activate
      python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
```

```
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")
```

```
      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')

      with open(xml_path, 'w') as f:
          f.write(xml_content)
```

```
    os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
    print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

    # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
file is used.
    os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
    print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

    # Print out the created xml file's contents
    print(f"V-Ray configuration file: {xml_path}")
    with open(xml_path, 'r') as f:
        print(f"{f.read()}")
```

5. Avant d'enregistrer l'environnement de file d'attente, apportez les modifications suivantes au texte de l'environnement selon vos besoins :

   - Mettez à jour les valeurs par défaut pour les paramètres suivants afin de refléter votre environnement :
     - LicenseInstanceID : ID d'instance Amazon EC2 de votre serveur de licences ou de votre instance proxy
     - LicenseInstanceRegion— La AWS région dans laquelle se trouve votre ferme
     - LicensePorts— Une liste de ports séparés par des virgules à transférer vers le serveur de licences ou l'instance proxy (par exemple 2700,2701)
   - Si vous souhaitez utiliser les licences basées sur l'utilisation (UBL) une fois la licence Bring your own (BYOL) épuisée, assurez-vous que le port est correct pour votre serveur de licences. Si vous ne souhaitez pas utiliser UBL après avoir épuisé le BYOL, ajoutez les variables d'environnement de licence requises dans la section des variables.

     Ces variables doivent diriger le fichier DCCs vers localhost sur le port du serveur de licences. Par exemple, si votre serveur de licences Foundry écoute sur le port 6101, vous devez ajouter la variable as. **`foundry_LICENSE: 6101@localhost`**

6. (Facultatif) Vous pouvez laisser la priorité définie sur 0, ou vous pouvez la modifier pour organiser la priorité différemment selon les environnements de files d'attente multiples.
7. Choisissez Créer un environnement de file d'attente pour enregistrer le nouvel environnement.

Une fois l'environnement de file d'attente défini, les tâches soumises à cette file d'attente récupéreront les licences du serveur de licences configuré.

Étape 2 : (Facultatif) Configuration de l'instance de proxy de licence

Au lieu d'utiliser un serveur de licences, vous pouvez utiliser un proxy de licence. Pour créer un proxy de licence, créez une nouvelle instance Amazon Linux 2023 disposant d'un accès réseau au serveur de licences. Si nécessaire, vous pouvez configurer cet accès à l'aide d'une connexion VPN. Pour plus d'informations, consultez la section Connexions VPN dans le guide de l'utilisateur Amazon VPC.

Pour configurer une instance de proxy de licence pour Deadline Cloud, suivez les étapes de cette procédure. Effectuez les étapes de configuration suivantes sur cette nouvelle instance pour permettre le transfert du trafic de licences vers votre serveur de licences

1. Pour installer le HAProxy package, entrez

```
sudo yum install haproxy
```

2. Mettez à jour la section Listen License-Server du fichier de configuration/etc/haproxy/haproxy.cfg avec ce qui suit :

   a. Remplacez LicensePort1 et LicensePort2 par les numéros de port à transférer au serveur de licences. Ajoutez ou supprimez des valeurs séparées par des virgules pour répondre au nombre de ports requis.

   b. Remplacez LicenseServerHostpar le nom d'hôte ou l'adresse IP du serveur de licences.

```
lobal
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    user        haproxy
    group       haproxy
    daemon

defaults
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
```

```
    timeout http-keep-alive 10s
    timeout check           10s

listen license-server
     bind *:LicensePort1,*:LicensePort2
     server license-server LicenseServerHost
```

3. Pour activer et démarrer le HAProxy service, exécutez les commandes suivantes :

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Une fois les étapes terminées, les demandes de licence envoyées à localhost depuis l'environnement de file d'attente de transfert doivent être transmises au serveur de licences spécifié.

Étape 3 : configuration CloudFormation du modèle

Vous pouvez utiliser un CloudFormation modèle pour configurer une ferme complète afin qu'elle utilise vos propres licences.

1. Modifiez le modèle fourni à l'étape suivante pour ajouter les variables d'environnement de licence requises dans la section des variables sous BYOLQueueEnvironnement.
2. Utilisez le CloudFormation modèle suivant.

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create &ADC; resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
```

```
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm

  QueuePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLQueuePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
              - s3:GetBucketLocation
            Resource:
              - !Sub ${JobAttachmentBucket.Arn}
              - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
            Condition:
              StringEquals:
                aws:ResourceAccount: !Sub ${AWS::AccountId}
          - Effect: Allow
            Action: logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          - Effect: Allow
            Action:
              - s3:ListBucket
              - s3:GetObject
            Resource:
              - "*"
            Condition:
              ArnLike:
                s3:DataAccessPointArn:
                  - arn:aws:s3:*:*:accesspoint/deadline-software-*
              StringEquals:
```

```
                s3:AccessPointNetworkOrigin: VPC

  BYOLSSMPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLSSMPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - ssm:StartSession
            Resource:
              - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
              - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}


  WorkerPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLWorkerPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - logs:CreateLogStream
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
            Condition:
              ForAnyValue:StringEquals:
                aws:CalledVia:
                  - deadline.amazonaws.com
          - Effect: Allow
            Action:
              - logs:PutLogEvents
              - logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*


  QueueRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLQueueRole
      ManagedPolicyArns:
        - !Ref QueuePolicy
        - !Ref BYOLSSMPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service:
                - credentials.deadline.amazonaws.com
                - deadline.amazonaws.com
            Condition:
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
              ArnEquals:
                aws:SourceArn: !Ref Farm

  WorkerRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLWorkerRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
        - !Ref WorkerPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service: credentials.deadline.amazonaws.com


  Queue:
    Type: AWS::Deadline::Queue
    Properties:
      DisplayName: BYOLQueue
      FarmId: !GetAtt Farm.FarmId
```

```
      RoleArn: !GetAtt QueueRole.Arn
      JobRunAsUser:
        Posix:
          Group: ""
          User: ""
        RunAs: WORKER_AGENT_USER
      JobAttachmentSettings:
        RootPrefix: job-attachments
        S3BucketName: !Ref JobAttachmentBucket

  Fleet:
    Type: AWS::Deadline::Fleet
    Properties:
      DisplayName: BYOLFleet
      FarmId: !GetAtt Farm.FarmId
      MinWorkerCount: 1
      MaxWorkerCount: 2
      Configuration:
        ServiceManagedEc2:
          InstanceCapabilities:
            VCpuCount:
              Min: 4
              Max: 16
            MemoryMiB:
              Min: 4096
              Max: 16384
            OsFamily: LINUX
            CpuArchitectureType: x86_64
          InstanceMarketOptions:
            Type: on-demand
      RoleArn: !GetAtt WorkerRole.Arn

  QFA:
    Type: AWS::Deadline::QueueFleetAssociation
    Properties:
      FarmId: !GetAtt Farm.FarmId
      FleetId: !GetAtt Fleet.FleetId
      QueueId: !GetAtt Queue.QueueId

  CondaQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 5
```

```
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: |
        specificationVersion: 'environment-2023-09'
        parameterDefinitions:
        - name: CondaPackages
          type: STRING
          description: >
            This is a space-separated list of conda package match specifications to
 install for the job.
            E.g. "blender=3.6" for a job that renders frames in Blender 3.6.

            See https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/
pkg-specs.html#package-match-specifications
          default: ""
          userInterface:
            control: LINE_EDIT
            label: Conda Packages
        - name: CondaChannels
          type: STRING
          description: >
            This is a space-separated list of conda channels from which to install
 packages. &ADC; SMF packages are
            installed from the "deadline-cloud" channel that is configured by
 &ADC;.

            Add "conda-forge" to get packages from the https://conda-forge.org/
 community, and "defaults" to get packages
            from Anaconda Inc (make sure your usage complies with https://
www.anaconda.com/terms-of-use).
          default: "deadline-cloud"
          userInterface:
            control: LINE_EDIT
            label: Conda Channels
        environment:
          name: Conda
          script:
            actions:
              onEnter:
                command: "conda-queue-env-enter"
                args: ["{{Session.WorkingDirectory}}/.env", "--packages",
 "{{Param.CondaPackages}}", "--channels", "{{Param.CondaChannels}}"]
              onExit:
                command: "conda-queue-env-exit"
```

```
  BYOLQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 10
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: !Sub |
        specificationVersion: "environment-2023-09"
        parameterDefinitions:
        - name: LicenseInstanceId
          type: STRING
          description: >
            The Instance ID of the license server/proxy instance
          default: ""
        - name: LicenseInstanceRegion
          type: STRING
          description: >
            The region containing this farm
          default: ""
        - name: LicensePorts
          type: STRING
          description: >
            Comma-separated list of ports to be forwarded to the license server/
proxy
            instance. Example:
 "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
          default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
        environment:
        name: BYOL License Forwarding
        variables:
          example_LICENSE: 2701@localhost
        script:
          actions:
          onEnter:
            command: bash
            args: [ "{{Env.File.Enter}}" ]
          onExit:
            command: bash
            args: [ "{{Env.File.Exit}}" ]
          embeddedFiles:
          - name: Enter
            type: TEXT
```

```
            runnable: True
            data: |
              curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
              chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
              conda activate
              python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
session-manager-plugin
          - name: Exit
            type: TEXT
            runnable: True
            data: |
              echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
              for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
          - name: StartSession
            type: TEXT
            data: |
              import boto3
              import json
              import subprocess
              import sys
              import os
              import tempfile

              instance_id = "{{Param.LicenseInstanceId}}"
              region = "{{Param.LicenseInstanceRegion}}"
              license_ports_list = "{{Param.LicensePorts}}".split(",")

              ssm_client = boto3.client("ssm", region_name=region)
              pids = []

              for port in license_ports_list:
                session_response = ssm_client.start_session(
                  Target=instance_id,
                  DocumentName="AWS-StartPortForwardingSession",
                  Parameters={"portNumber": [port], "localPortNumber": [port]}
                )

                cmd = [
                  sys.argv[1],
                  json.dumps(session_response),
                  region,
```

```
                  "StartSession",
                  "",
                  json.dumps({"Target": instance_id}),
                  f"https://ssm.{region}.amazonaws.com"
                ]

                process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
                pids.append(process.pid)
                print(f"SSM Port Forwarding Session started for port {port}")

            print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in
 pids)}")

            # Enabling UBL after the "bring your own license" (BYOL) has run out
 requires prepending the BYOL configuration to the existing license setup
            # Remove the sections that do not apply to your pipeline, or you do
 not want to use UBL after exhausting the BYOL licenses.
            # The port numbers used may not match what your license server is
 serving.

            # Arnold
            os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
            print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

            # Nuke
            os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
            print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

            # SideFX
            os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
            print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

            # Redshift and Red Giant
            os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
            print(f"openjd_env:
 redshift_LICENSE={os.environ['redshift_LICENSE']}")
```

```
            # V-Ray doesn't support multiple license servers in a single
 environment variable
            # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
            vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
            xml_content = """<VRLClient>
              <LicServer>
                <Host>localhost</Host>
                <Port>30304</Port>"""

            if vray_license and vray_license.startswith('licset://'):
                server_parts = vray_license.removeprefix('licset://').split(':')
                if len(server_parts) >= 2:
                    xml_content += f"""
                <Host1>{server_parts[0]}</Host1>
                <Port1>{server_parts[1]}</Port1>"""

            xml_content += """
                <User></User>
                <Pass></Pass>
              </LicServer>
            </VRLClient>"""

            temp_dir = tempfile.gettempdir()
            xml_path = os.path.join(temp_dir, 'vrlclient.xml')

            with open(xml_path, 'w') as f:
                f.write(xml_content)

            os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
            print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

            # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the
 vrlclient.xml file is used.
            os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
            print(f"openjd_env:
 VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

            # Print out the created xml file's contents
            print(f"V-Ray configuration file: {xml_path}")
            with open(xml_path, 'r') as f:
                print(f"{f.read()}")
```

3. Lors du déploiement du CloudFormation modèle, fournissez les paramètres suivants :

   - Mettez à jour l'LicenseInstanceID avec l'ID d'instance Amazon EC2 de votre serveur de licences ou de votre instance proxy
   - Mettez à jour le LicensePortsavec une liste de ports séparés par des virgules à transférer vers le serveur de licences ou l'instance proxy (par exemple 2700,2701)
   - Ajoutez les variables d'environnement de licence en les remplaçant **`example_LICENSE: 2700@localhost`** dans le modèle

4. Déployez le modèle pour configurer votre ferme avec la fonctionnalité « Apportez votre propre licence ».

Connectez les flottes gérées par le client à un point de terminaison de licence

Le serveur de licences basé sur l'utilisation de AWS Deadline Cloud fournit des licences à la demande pour certains produits tiers. Avec les licences basées sur l'utilisation, vous pouvez payer au fur et à mesure. Vous n'êtes facturé que pour le temps que vous utilisez. Les licences basées sur l'utilisation fournissent des licences que les employés de Deadline Cloud peuvent afficher, elles ne fournissent pas de licences pour vos applications DCC.

Le serveur de licences basé sur l'utilisation de Deadline Cloud peut être utilisé avec n'importe quel type de flotte, à condition que les employés de Deadline Cloud puissent communiquer avec le serveur de licences. Le serveur de licences est automatiquement configuré dans les flottes gérées par les services. La configuration suivante n'est requise que pour les flottes gérées par le client.

Pour créer le serveur de licences, vous avez besoin d'un groupe de sécurité pour le VPC de votre parc qui autorise le trafic pour les licences tierces.

Rubriques

Étape 1 : créer un groupe de sécurité

Utilisez la console Amazon VPC pour créer un groupe de sécurité pour le VPC de votre ferme. Configurez le groupe de sécurité pour autoriser les règles entrantes suivantes :

- Autodesk Maya et Arnold — 2701 à 2702, TCP, IPv4 IPv6
- Cinéma 4D — 7057, TCP, IPv4 IPv6
- Foundry Nuke — 6101, TCP, IPv4 IPv6
- Géant rouge — 7055, TCP, IPV4
- Redshift — 7054, TCP, IPv4 IPv6
- SideFX Houdini, Mantra et Karma — 1715-1717, TCP, IPv4 IPv6
- VRay — 30304, TCP, IPV4

La source de chaque règle entrante est le groupe de sécurité des employés de la flotte.

Pour plus d'informations sur la création d'un groupe de sécurité, consultez la section Créer un groupe de sécurité dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

Étape 2 : configurer le point de terminaison de licence

Un point de terminaison de licence fournit un accès aux serveurs de licences pour les produits tiers. Les demandes de licence sont envoyées au point de terminaison de licence. Le point de terminaison les achemine vers le serveur de licences approprié. Le serveur de licences assure le suivi des limites d'utilisation et des droits. La création d'un point de terminaison de licence dans Deadline Cloud fournit un point de terminaison d' AWS PrivateLink interface dans votre VPC. Ces terminaux sont facturés conformément à la tarification standard. AWS PrivateLink Pour en savoir plus, consultez Pricing AWS PrivateLink (Tarification).

Avec les autorisations appropriées, vous pouvez créer votre point de terminaison de licence. Pour connaître la politique requise pour créer un point de terminaison de licence, voir Politique autorisant la création d'un point de terminaison de licence.

Vous pouvez créer votre point de terminaison de licence depuis votre tableau de bord dans la console Deadline Cloud.

1. Dans le volet de navigation de gauche, choisissez License endpoints, puis Create license endpoint.

2. Sur la page Créer un point de terminaison de licence, effectuez les opérations suivantes :
   - Sélectionnez un VPC.
   - Sélectionnez les sous-réseaux qui contiennent vos collaborateurs de Deadline Cloud. Vous pouvez sélectionner jusqu'à 10 sous-réseaux.
   - Sélectionnez le groupe de sécurité que vous avez créé à l'étape 1. Vous pouvez sélectionner jusqu'à 10 groupes de sécurité pour des scénarios plus complexes.
   - (Facultatif) Choisissez Ajouter un nouveau tag et ajoutez un ou plusieurs tags. Vous pouvez ajouter jusqu'à 50 balises.
3. Choisissez Créer un point de terminaison de licence. Lorsque le point de terminaison de licence est créé, il s'affiche sur la page des points de terminaison de licence.
4. Dans la section des produits mesurés, choisissez Ajouter des produits, puis sélectionnez les produits que vous souhaitez ajouter à votre point de terminaison de licence. Choisissez Ajouter.

Pour supprimer un produit d'un point de terminaison de licence, dans la section des produits mesurés, sélectionnez le produit, puis choisissez Supprimer. Dans la confirmation, choisissez à nouveau Supprimer.

Étape 3 : Connecter une application de rendu à un point de terminaison

Une fois le point de terminaison de licence configuré, les applications l'utilisent de la même manière qu'elles utilisent un serveur de licences tiers. Vous configurez généralement le serveur de licences de l'application en définissant une variable d'environnement ou un autre paramètre système, tel qu'une clé de registre Microsoft Windows, sur un port et une adresse de serveur de licences.

Pour obtenir le nom DNS du point de terminaison de licence, sélectionnez le point de terminaison de licence dans la console, puis cliquez sur l'icône de copie dans la section Nom DNS.

Exemples de configuration

Example— Autodesk Maya et Arnold

> **Note**
>
> Vous pouvez utiliser Autodesk Maya et Arnold ensemble ou séparément. Utilisez le port 2702 pour Autodesk Maya et le port 2701 pour Arnold.

Pour Autodesk Maya, définissez la variable d'environnement `ADSKFLEX_LICENSE_FILE` sur :

```
2702@VPC_Endpoint_DNS_Name
```

Pour Arnold, définissez la variable d'environnement `ADSKFLEX_LICENSE_FILE` sur :

```
2701@VPC_Endpoint_DNS_Name
```

Pour Autodesk Maya et Arnold, définissez la variable d'environnement `ADSKFLEX_LICENSE_FILE` sur :

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Pour les Windows utilisateurs, utilisez un point-virgule (;) au lieu de deux points (:)) pour séparer les points de terminaison.

Example— Cinéma 4D

Définissez la variable d'environnement `g_licenseServerRLM` sur :

```
VPC_Endpoint_DNS_Name:7057
```

Après avoir créé la variable d'environnement, vous devriez pouvoir afficher une image à l'aide d'une ligne de commande similaire à celle-ci :

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^
    "C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^
    -oimage "C:\Users\Administrator\User\MyOutputImage.png
```

Example— Foundry Nuke

Définissez la variable d'environnement `foundry_LICENSE` sur :

```
6101@VPC_Endpoint_DNS_Name
```

Pour vérifier que les licences fonctionnent correctement, vous pouvez exécuter Nuke dans un terminal :

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Géant rouge

Définissez la variable d'environnement `redshift_LICENSE` sur :

```
7055@VPC_Endpoint_DNS_Name
```

Notez que Red Giant et Redshift ont la même variable d'`redshift_LICENSE`environnement. Si vous souhaitez utiliser les deux applications, vous pouvez définir la variable d'environnement comme suit :

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Pour les Windows utilisateurs, utilisez un point-virgule (;) au lieu de deux points (:)) pour séparer les points de terminaison.

Pour vérifier que les licences fonctionnent correctement, assurez-vous d'avoir installé After Effects et Red Giant. Vous pouvez ensuite afficher un projet à l'aide d'une commande similaire à celle-ci :

```
C:\Program Files\Adobe\Adobe After Effects 2025\Support Files\aerender.exe -comp "Comp
 1" -project
         C:\Users\MyUser\myAfterEffectsProjectUsingRedGiant.aep -output
           C:\Users\MyUser\myMovieWithRedGiant.mp4
```

Example— Redshift

Définissez la variable d'environnement `redshift_LICENSE` sur :

```
7054@VPC_Endpoint_DNS_Name
```

Après avoir créé la variable d'environnement, vous devriez pouvoir afficher une image à l'aide d'une ligne de commande similaire à celle-ci :

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SideFX Houdini, Mantra et Karma

Exécutez la commande suivante :

```
/opt/hfs19.5.640/bin/hserver -S
 "http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://
VPC_Endpoint_DNS_Name:1717;"
```

Pour vérifier que les licences fonctionnent correctement, vous pouvez effectuer le rendu d'une scène Houdini à l'aide de cette commande :

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Définissez la variable d'environnement `VRAY_AUTH_CLIENT_SETTINGS` sur :

```
licset://VPC_Endpoint_DNS_Name:30304
```

Définissez la variable d'environnement `VRAY_AUTH_CLIENT_FILE_PATH` sur :

```
/null
```

Pour vérifier que les licences fonctionnent correctement, vous pouvez afficher une image à VRay l'aide d'une commande similaire à celle-ci :

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Étape 4 : Supprimer un point de terminaison de licence

Lorsque vous supprimez votre flotte gérée par le client, n'oubliez pas de supprimer le point de terminaison de votre licence. Si vous ne supprimez pas le point de terminaison de licence, les frais AWS PrivateLink fixes continueront de vous être facturés

Vous pouvez supprimer le point de terminaison de votre licence depuis votre tableau de bord dans la console Deadline Cloud.

1. Dans le volet de navigation de gauche, choisissez License endpoints.
2. Sélectionnez le point de terminaison que vous souhaitez supprimer et choisissez Supprimer, puis choisissez à nouveau Supprimer pour confirmer.

Utilisation d'agents d'IA avec Deadline Cloud

Utilisez des agents d'intelligence artificielle pour rédiger des ensembles de tâches, développer des packages Conda et résoudre des problèmes de tâches dans Deadline Cloud. Cette rubrique explique ce que sont les agents IA, les points essentiels pour travailler efficacement avec eux et les ressources destinées à aider les agents à comprendre Deadline Cloud.

Un agent d'intelligence artificielle est un outil logiciel qui utilise un grand modèle de langage (LLM) pour effectuer des tâches de manière autonome. Les agents d'intelligence artificielle peuvent lire et écrire des fichiers, exécuter des commandes et modifier des solutions en fonction des commentaires. Les exemples incluent les outils de ligne de commande tels que Kiro et les assistants intégrés à l'IDE.

Points clés de la collaboration avec des agents de l'IA

Les points clés suivants vous aident à obtenir de meilleurs résultats lorsque vous utilisez des agents d'intelligence artificielle avec Deadline Cloud.

- Fournir une base : les agents d'intelligence artificielle sont plus performants lorsqu'ils ont accès à la documentation, aux spécifications et aux exemples pertinents. Vous pouvez fournir une base en dirigeant l'agent vers des pages de documentation spécifiques, en partageant des exemples de code existants comme références, en clonant des référentiels open source pertinents dans l'espace de travail local et en fournissant de la documentation pour des applications tierces.
- Spécifier les critères de succès : définissez le résultat attendu et les exigences techniques pour l'agent. Par exemple, lorsque vous demandez à un agent de développer un ensemble de tâches, spécifiez les entrées, les paramètres et les résultats attendus de la tâche. Si vous n'êtes pas sûr des spécifications, demandez d'abord à l'agent de proposer des options, puis d'affiner les exigences ensemble.
- Activez une boucle de feedback : les agents d'IA itèrent plus efficacement lorsqu'ils peuvent tester leurs solutions et recevoir des commentaires. Au lieu de s'attendre à une solution efficace dès la première tentative, donnez à l'agent la possibilité d'exécuter sa solution et d'examiner les résultats. Cette approche fonctionne bien lorsque l'agent peut accéder aux mises à jour de statut, aux journaux et aux erreurs de validation. Par exemple, lorsque vous développez un ensemble de tâches, autorisez l'agent à soumettre la tâche et à consulter les journaux.
- Attendez-vous à répéter : même dans un contexte favorable, les agents peuvent faire fausse route ou émettre des suppositions qui ne correspondent pas à votre environnement. Observez la façon dont l'agent aborde la tâche et donnez-lui des conseils tout au long du processus. Ajoutez

le contexte manquant si l'agent a des difficultés, aidez à détecter les erreurs en pointant vers des fichiers journaux spécifiques, affinez les exigences au fur et à mesure que vous les découvrez et ajoutez des exigences négatives pour indiquer explicitement ce que l'agent doit éviter.

Ressources pour le contexte des agents

Les ressources suivantes aident les agents d'intelligence artificielle à comprendre les concepts de Deadline Cloud et à produire des résultats précis.

- Serveur MCP (Deadline Cloud Model Context Protocol) : pour les agents qui prennent en charge le Model Context Protocol, le référentiel Deadline-Cloud contient le client Deadline Cloud, qui inclut un serveur MCP pour interagir avec les tâches.
- AWSServeur MCP de documentation — Pour les agents qui prennent en charge le MCP, configurez le serveur MCP de AWS documentation pour donner à l'agent un accès direct à la AWS documentation, y compris le guide de l'utilisateur et le guide du développeur de Deadline Cloud.
- Spécification Open Job Description — La spécification Open Job Description GitHub définit le schéma des modèles de tâches. Référez-vous à ce référentiel lorsque les agents ont besoin de comprendre la structure et la syntaxe des modèles de tâches.
- deadline-cloud-samples— Le deadline-cloud-samplesréférentiel contient des exemples de lots de tâches, des recettes de conda et des CloudFormation modèles pour des applications et des cas d'utilisation courants.
- organisation aws-deadline — GitHub L'organisation aws-deadline GitHub contient des plugins de référence pour de nombreuses applications tierces que vous pouvez utiliser comme exemples pour d'autres intégrations.

Exemple d'invite : rédaction d'un ensemble de tâches

L'exemple d'invite suivant montre comment utiliser un agent d'intelligence artificielle pour créer des ensembles de tâches qui entraînent un adaptateur LoRa (Low-Rank Adaptation) à la génération d'images d'IA. L'invite illustre les points clés évoqués précédemment : elle fournit une base en pointant vers les référentiels pertinents, définit les critères de réussite pour les résultats des lots de tâches et décrit une boucle de rétroaction pour le développement itératif.

```
Write a pair of job bundles for Deadline Cloud that use the diffusers Python library to
 train a LoRA adapter on a set of images and then generate images from it.

Requirements:
```

```
- The training job takes a set of JPEG images as input, uses an image description,
 LoRA rank, learning rate, batch size, and number of training steps as parameters, and
 outputs a `.safetensors` file.
- The generation job takes the `.safetensors` file as input and the number of images
 to generate, then outputs JPEG images. The jobs use Stable Diffusion 1.5 as the base
 model.
- The jobs run `diffusers` as a Python script. Install the necessary packages using
 conda by setting the job parameters:
    - `CondaChannels`: `conda-forge`
    - `CondaPackages`: list of conda packages to install

For context, clone the following repositories to your workspace and review their
 documentation and code:
- OpenJobDescription specification: https://github.com/OpenJobDescription/openjd-
specifications/blob/mainline/wiki/2023-09-Template-Schemas.md
- Deadline Cloud sample job bundles: https://github.com/aws-deadline/deadline-cloud-
samples/tree/mainline/job_bundles
- diffusers library: https://github.com/huggingface/diffusers

Read through the provided context before you start. To develop a job bundle, iterate
 with the following steps until the submitted job succeeds. If a step fails, update the
 job bundle and restart the loop:
1. Create a job bundle.
2. Validate the job template syntax: `openjd check`
3. Submit the job to Deadline Cloud: `deadline bundle submit`
4. Wait for the job to complete: `deadline job wait`
5. View the job status and logs: `deadline job logs`
6. Download the job output: `deadline job download-output`

To verify the training and generation jobs work together, iterate with the following
 steps until the generation job produces images that resemble the dog in the training
 data:
1. Develop and submit a training job using the training images in `./exdog`
2. Wait for the job to succeed then download its output.
3. Develop and submit a generation job using the LoRA adapter from the training job.
4. Wait for the job to succeed then download its output.
5. Inspect the generated images. If they resemble the dog in the training data, you're
 done. Otherwise, review the job template, job parameters, and job logs to identify and
 fix the issue.
```

Surveillance de AWS Deadline Cloud

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de AWS Deadline Cloud (Deadline Cloud) et de vos AWS solutions. Collectez des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Avant de commencer à surveiller Deadline Cloud, vous devez créer un plan de surveillance comprenant des réponses aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- A quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

AWSet Deadline Cloud fournissent des outils que vous pouvez utiliser pour surveiller vos ressources et répondre aux incidents potentiels. Certains de ces outils assurent la surveillance à votre place, d'autres nécessitent une intervention manuelle. Vous devez automatiser les tâches de surveillance autant que possible.

- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos EC2 instances Amazon et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

  Deadline Cloud dispose de trois CloudWatch indicateurs.

- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux à partir d' EC2 instances Amazon et d'autres sources. CloudTrail CloudWatch Les journaux peuvent surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux

dans une solution de stockage hautement durable. Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon CloudWatch Logs.

- Amazon EventBridge peut être utilisé pour automatiser vos AWS services et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements AWS liés aux services sont diffusés EventBridge en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.
- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelésAWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d’informations, consultez le Guide de l’utilisateur AWS CloudTrail.

Rubriques

- Journalisation des appels Deadline Cloud d'API à l'aide AWS CloudTrail
- Surveillance avec CloudWatch
- Gestion des événements Deadline Cloud à l'aide de Amazon EventBridge

Journalisation des appels Deadline Cloud d'API à l'aide AWS CloudTrail

Deadline Cloud est intégré à AWS CloudTrailun service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API Deadline Cloud sous forme d'événements. Les appels capturés incluent des appels provenant de la Deadline Cloud console et des appels de code vers les opérations de l' Deadline Cloud API. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite Deadline Cloud, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des informations supplémentaires.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l’identité permettent de déterminer :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou d'utilisateur root.
- Si la demande a été faite au nom d'un utilisateur du centre d'identité IAM.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section Utilisation de l'historique des CloudTrail événements dans le guide de AWS CloudTrail l'utilisateur. L'affichage de CloudTrail l'historique des événements est gratuit.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou CloudTrailLake.

CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous ne pouvez créer un journal de suivi en une ou plusieurs régions à l'aide de l' AWS CLI. Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un journal de suivi pour une seule région, il convient de n'afficher que les événements enregistrés dans le journal de suivi pour une seule région Région AWS. Pour plus d'informations sur les journaux de suivi, consultez Créez un journal de suivi dans vos Compte AWS et Création d'un journal de suivi pour une organisation dans le AWS CloudTrail Guide de l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours dans votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification. Pour obtenir des informations sur la tarification Amazon S3, consultez Tarification Amazon S3.

CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format Apache ORC. ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des sélecteurs d'événements avancés. Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section Travailler avec AWS CloudTrail Lake dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'option de tarification que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Deadline Cloud événements de données dans CloudTrail

Les événements de données fournissent des informations sur les opérations de ressources effectuées sur ou dans une ressource (par exemple, lecture ou écriture de données dans un objet Amazon S3). Ils sont également connus sous le nom opérations de plans de données. Les événements de données sont souvent des activités à fort volume. Par défaut, CloudTrail n'enregistre pas les événements liés aux données. L'historique des CloudTrail événements n'enregistre pas les événements liés aux données.

Des frais supplémentaires s'appliquent pour les événements de données. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Vous pouvez enregistrer les événements de données pour les types de Deadline Cloud ressources à l'aide de la CloudTrail console ou AWS CLI des opérations de CloudTrail l'API. Pour plus d'informations sur la façon de journaliser les événements de données, consultez Journalisation des événements de données avec la AWS Management Console et Journalisation des événements de données avec l' AWS Command Line Interface dans le Guide de l'utilisateur AWS CloudTrail .

Le tableau suivant répertorie les types de Deadline Cloud ressources pour lesquels vous pouvez enregistrer des événements de données. La colonne Type d'événement de données (console)

indique la valeur à choisir dans la liste des types d'événements de données de la CloudTrail console. La colonne de valeur resources.type indique la **`resources.type`** valeur que vous devez spécifier lors de la configuration de sélecteurs d'événements avancés à l'aide du ou. AWS CLI CloudTrail APIs La CloudTrail colonne Données APIs enregistrées indique les appels d'API enregistrés CloudTrail pour le type de ressource.

Type d’événement de données (console)	valeur resources.type	Données APIs enregistrées sur CloudTrail
Flotte Deadline	`AWS::Deadline::Fleet`	• SearchWorkers
File d'attente pour les	`AWS::Deadline::Fleet`	• SearchJobs
Deadline Worker	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers
Deadline Job	`AWS::Deadline::Job`	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep

Type d'événement de données (console)	valeur resources.type	Données APIs enregistrées sur CloudTrail
		• UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

Vous pouvez configurer des sélecteurs d'événements avancés pour filtrer les champs `eventName`, `readOnly` et `resources.ARN` afin de ne journaliser que les événements importants pour vous. Pour plus d'informations sur ces champs, consultez AdvancedFieldSelector dans la Référence des API AWS CloudTrail .

Deadline Cloud événements de gestion dans CloudTrail

Les événements de gestion fournissent des informations sur les opérations de gestion effectuées sur les ressources de votre Compte AWS. Ils sont également connus sous le nom opérations de plan de contrôle. Par défaut, CloudTrail enregistre les événements de gestion.

AWS Deadline Cloud enregistre les opérations du plan Deadline Cloud de contrôle suivantes en CloudTrail tant qu'événements de gestion.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-lire
- assume-fleet-role-for-travailleur
- assume-queue-role-for-lire
- assume-queue-role-for-utilisateur
- assume-queue-role-for-travailleur
- créer un budget

- créer une ferme
- create-fleet
- create-license-endpoint
- créer une limite
- créer un moniteur
- créer une file
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- create-worker
- supprimer-budget
- supprime-ferme
- delete-fleet
- delete-license-endpoint
- supprimer-limite
- delete-metered-product
- supprimer-moniteur
- supprimer-file
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- supprime-travailleur
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- obtenir un budget

- get-farm
- get-feature-map
- get-fleet
- get-license-endpoint
- get-limit
- get-monitor
- get-queue
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-file d'attente
- list-available-metered-products
- liste-budgets
- list-farm-members
- listez les fermes
- list-fleet-members
- listes de flottes
- list-job-members
- list-license-endpoints
- limite de liste
- list-metered-products
- moniteurs de liste
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- list-queues
- list-storage-profiles

- list-storage-profiles-for-file d'attente
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- mise à jour du budget
- ferme de mise à jour
- mettre à jour le parc
- limite de mise à jour
- moniteur de mise à jour
- file d'attente de mise à jour
- update-queue-environment
- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- agent de mise à jour

Deadline Cloud exemples d'événements

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un CloudTrail événement illustrant l'`CreateFarm`opération.

```
{
    "eventVersion": "0",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
        "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-
Session",
        "accountId": "111122223333",
```

```
        "accessKeyId": "EXAMPLE-accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE-PrincipalID",
                "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
                "accountId": "111122223333",
                "userName": "EXAMPLE-UserName"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-03-08T23:25:49Z"
            }
        }
    },
    "eventTime": "2021-03-08T23:25:49Z",
    "eventSource": "deadline.amazonaws.com",
    "eventName": "CreateFarm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "EXAMPLE-userAgent",
    "requestParameters": {
        "displayName": "example-farm",
        "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
        "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
        "description": "example-description",
        "tags": {
            "purpose_1": "e2e"
            "purpose_2": "tag_test"
        }
    },
    "responseElements": {
        "farmId": "EXAMPLE-farmID"
    },
    "requestID": "EXAMPLE-requestID",
    "eventID": "EXAMPLE-eventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
}
```

L'exemple JSON montre l' Région AWS adresse IP et d'autres « `requestParameters` », tels que le « `displayName` » et le « `kmsKeyArn` », qui peuvent vous aider à identifier l'événement.

Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir le contenu des CloudTrail enregistrements dans le Guide de AWS CloudTrail l'utilisateur.

Surveillance avec CloudWatch

Amazon CloudWatch (CloudWatch) collecte des données brutes et les transforme en indicateurs lisibles en temps quasi réel. Vous pouvez ouvrir la CloudWatch console à l'adresse suivante https://console.aws.amazon.com/cloudwatch/pour consulter et filtrer les métriques de Deadline Cloud.

Ces statistiques sont conservées pendant 15 mois afin que vous puissiez accéder aux informations historiques afin d'avoir une meilleure idée des performances de votre application ou service Web. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

Deadline Cloud possède deux types de journaux : les journaux des tâches et les journaux des travailleurs. Un journal des tâches se produit lorsque vous exécutez des journaux d'exécution sous forme de script ou lors de l'exécution de DCC. Un journal des tâches peut afficher des événements tels que le chargement de ressources, le rendu des tuiles ou l'impossibilité de trouver des textures.

Un journal des travailleurs indique les processus des agents des travailleurs. Cela peut inclure des éléments tels que le moment où les agents de travail démarrent, s'enregistrent, signalent les progrès, chargent des configurations ou terminent des tâches.

L'espace de noms de ces journaux est`/aws/deadline/*`.

Pour Deadline Cloud, les employés téléchargent ces CloudWatch journaux dans Logs. Par défaut, les journaux n'expirent jamais. Si une tâche produit un volume élevé de données, vous pouvez encourir des coûts supplémentaires. Pour plus d'informations, consultez les CloudWatch tarifs Amazon.

Vous pouvez ajuster la politique de rétention pour chaque groupe de journaux. Une durée de conservation plus courte permet de supprimer les anciens journaux et de réduire les coûts de stockage. Pour conserver les journaux, vous pouvez les archiver sur Amazon Simple Storage Service avant de les supprimer. Pour plus d'informations, consultez Exporter les données du journal vers Amazon S3 à l'aide de la console dans le guide de CloudWatch l'utilisateur Amazon.

> **Note**
>
> CloudWatch les lectures du journal sont limitées parAWS. Si vous prévoyez d'intégrer de nombreux artistes, nous vous suggérons de contacter AWS le service client et de demander une augmentation du `GetLogEvents` quota en CloudWatch. En outre, nous vous recommandons de fermer le portail de suivi des journaux lorsque vous n'êtes pas en train de déboguer.

Pour plus d'informations, consultez la section Quotas de CloudWatch journaux dans le guide de CloudWatch l'utilisateur Amazon.

CloudWatch métriques

Deadline Cloud envoie des métriques à Amazon CloudWatch. Vous pouvez utiliser l'API AWS Management ConsoleAWS CLI, le ou une API pour répertorier les métriques auxquelles Deadline Cloud envoie CloudWatch. Par défaut, chaque point de données couvre la minute qui suit l'heure de début de l'activité. Pour plus d'informations sur la façon de consulter les statistiques disponibles à l'aide du AWS Management Console ou duAWS CLI, consultez la section Afficher les mesures disponibles dans le guide de CloudWatch l'utilisateur Amazon.

Indicateurs de flotte gérés par le client

L'espace de `AWS/DeadlineCloud` noms contient les métriques suivantes pour vos flottes gérées par le client :

Métrique	Description	Unit
`RecommendedFleetSize`	Le nombre de travailleurs que Deadline Cloud vous recommande d'utiliser pour traiter les tâches. Vous pouvez utiliser cette métrique pour augmenter ou réduire le nombre de travailleurs de votre flotte.	Nombre

Métrique	Description	Unit
`UnhealthyWorkerCount`	Le nombre de travailleurs affectés à des tâches de traitement qui ne sont pas saines.	Nombre

Vous pouvez utiliser les dimensions suivantes pour affiner les indicateurs de flotte gérés par le client :

Dimension	Description
FarmId	Cette dimension filtre les données que vous demandez à la ferme spécifiée.
FleetId	Cette dimension filtre les données que vous demandez au parc de travailleurs spécifié.

Métriques relatives aux licences

L'espace de `AWS/DeadlineCloud` noms contient les métriques suivantes pour les licences :

Métrique	Description	Unit
`LicensesInUse`	Le nombre de sessions de licence en cours d'utilisation.	Nombre

Vous pouvez utiliser les dimensions suivantes pour affiner les indicateurs de licence :

Dimension	Description
FleetId	Utilisez cette dimension pour filtrer les données en fonction du parc géré par le service spécifié. Pour les flottes gérées par le client, utilisez plutôt la LicenseEndpointId dimension.

Dimension	Description
LicenseEndpointId	Utilisez cette dimension pour filtrer les données vers le point de terminaison de licence spécifié.
Produit (langue française non garantie)	Utilisez cette dimension pour filtrer les données en fonction du produit mesuré spécifié.

Mesures relatives aux limites de ressources

L'espace de `AWS/DeadlineCloud` noms contient les métriques suivantes concernant les limites de ressources :

Métrique	Description	Unit
`CurrentCount`	Le nombre de ressources modélisées par cette limite d'utilisation.	Nombre
`MaxCount`	Le nombre maximum de ressources modélisées par cette limite. Si vous définisse z la `maxCount` valeur sur -1 à l'aide de l'API, Deadline Cloud n'émet pas la `MaxCount` métrique.	Nombre

Vous pouvez utiliser les dimensions suivantes pour affiner les mesures de limite simultanée :

Dimension	Description
FarmId	Cette dimension filtre les données que vous demandez à la ferme spécifiée.
LimitId	Cette dimension filtre les données que vous demandez jusqu'à la limite spécifiée.

Alarmes recommandées

Vous pouvez ainsi créer des alarmes qui surveillent les métriques et vous envoient une notification ou exécutent une autre action lorsqu'un seuil est dépassé. CloudWatch Pour plus d'informations sur la configuration des CloudWatch alarmes, consultez le guide de CloudWatch l'utilisateur Amazon.

Nous vous recommandons de définir des alarmes pour les métriques de Deadline Cloud suivantes :

LicensesInUse

Dimensions : FleetId, LicenseEndpointId

Description de l'alarme : Cette alarme détecte le moment où les sessions de licence actives pour un parc géré par un service ou un point de terminaison de licence approchent le quota de votre compte. Dans ce cas, vous pouvez augmenter le quota de comptes pour les sessions de licence. Consultez vos quotas actuels et demandez des augmentations à l'aide de Service Quotas. Pour en savoir plus, consultez le Guide de l'utilisateur du Service Quotas.

Intention : Prévenir les échecs de récupération des licences en surveillant l'utilisation avant qu'elle n'atteigne la limite de quota.

Statistique : maximum

Seuil recommandé : 90 % de votre quota de sessions de licence

Justification du seuil : définissez le seuil à un pourcentage de votre quota, afin de pouvoir prendre des mesures avant qu'il n'atteigne la limite.

Durée : 1 minute

Points de données pour le déclenchement d'alarme : 1

Période d'évaluation : 1

Opérateur de comparaison : `GREATER_THAN_THRESHOLD`

Ressources supplémentaires

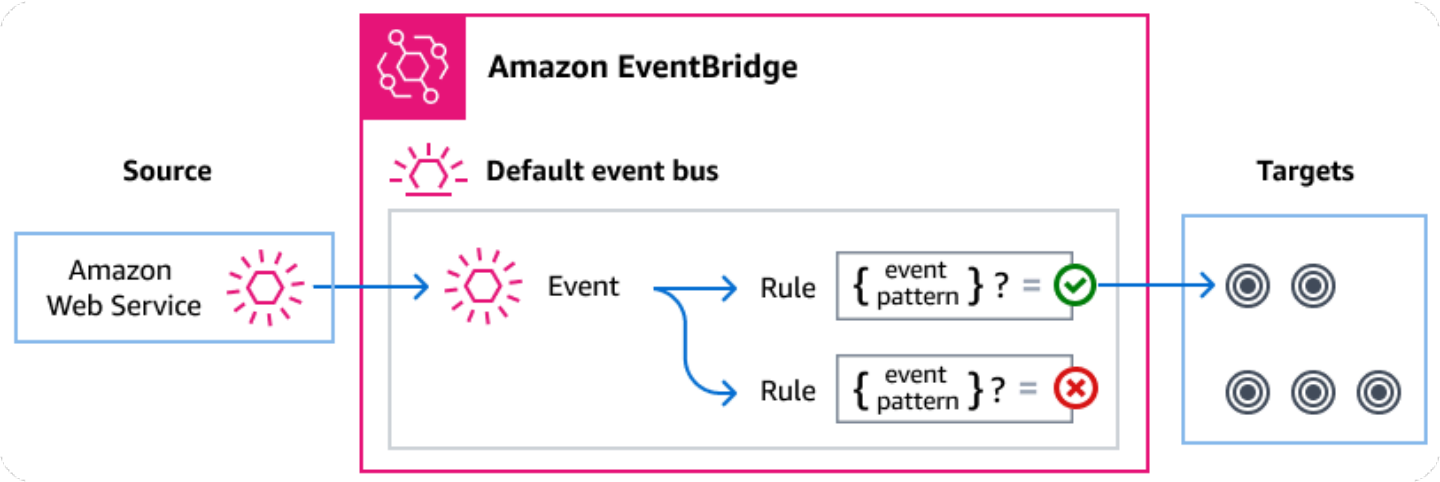
- Guide de CloudWatch l'utilisateur Amazon
- Guide de l’utilisateur de Service Quotas

Gestion des événements Deadline Cloud à l'aide de Amazon EventBridge

Amazon EventBridge est un service sans serveur qui utilise des événements pour connecter les composants de l'application entre eux, ce qui vous permet de créer plus facilement des applications évolutives pilotées par des événements. Une architecture pilotée par les événements est un style de création de systèmes logiciels faiblement couplés qui fonctionnent ensemble en émettant des événements et en y répondant. Les événements représentent une modification d'une ressource ou d'un environnement.

Voici comment cela fonctionne :

Comme c'est le cas pour de nombreux AWS services, Deadline Cloud génère et envoie des événements au bus d'événements EventBridge par défaut. (Le bus d'événements par défaut est automatiquement configuré dans chaque AWS compte.) Un bus d'événements est un routeur qui reçoit des événements et les transmet à zéro ou plusieurs destinations, ou cibles. Les règles que vous spécifiez pour le bus d'événements évaluent les événements à mesure qu'ils arrivent. Chaque règle vérifie si un événement correspond au modèle d'événement de la règle. Si l'événement correspond, le bus d'événements envoie l'événement à la ou aux cibles spécifiées.



Rubriques

- Événements Deadline Cloud
- Diffusion d'événements Deadline Cloud à l'aide de EventBridge règles
- Référence détaillée des événements de Deadline Cloud

Événements Deadline Cloud

Deadline Cloud envoie automatiquement les événements suivants au bus EventBridge d'événements par défaut. Les événements qui correspondent au modèle d'événements d'une règle sont transmis aux cibles spécifiées dans la mesure du possible. Les événements peuvent être livrés hors service.

Pour plus d'informations, consultez les EventBridge événements dans le guide de Amazon EventBridge l'utilisateur.

Type de détail d'événement	Description
Seuil budgétaire atteint	Envoyé lorsqu'une file d'attente atteint un pourcentage du budget qui lui est attribué.
Modification de l'état du cycle de vie des tâches	Envoyé en cas de modification de l'état du cycle de vie d'une tâche.
Modification du statut de Job Run	Envoyé lorsque le statut général des tâches d'une tâche change.
Étape Modification de l'état du cycle de vie	Envoyé en cas de modification de l'état du cycle de vie d'une étape d'une tâche.
Étape Exécuter le changement de statut	Envoyé lorsque le statut général des tâches d'une étape change.
Modification du statut d'exécution de la tâche	Envoyé lorsque le statut d'une tâche change.

Diffusion d'événements Deadline Cloud à l'aide de EventBridge règles

Pour que le bus d'événements EventBridge par défaut envoie les événements Deadline Cloud à une cible, vous devez créer une règle. Chaque règle contient un modèle d'événement, qui EventBridge correspond à chaque événement reçu sur le bus d'événements. Si les données d'événement correspondent au modèle d'événement spécifié, EventBridge transmet cet événement aux cibles de la règle.

Pour obtenir des instructions complètes sur la création de règles de bus d'événements, voir Création de règles réagissant aux événements dans le Guide de EventBridge l'utilisateur.

Création de modèles d'événements correspondant aux événements de Deadline Cloud

Chaque modèle d'événement est un objet JSON qui contient :

- Un attribut `source` qui identifie le service qui envoie l'événement. Pour les événements Deadline Cloud, la source est `aws.deadline`.
- (Facultatif) : un attribut `detail-type` qui contient un tableau des types d'événements à associer.
- (Facultatif) : un attribut `detail` qui contient toute autre donnée d'événement à rechercher.

Par exemple, le modèle d'événement suivant correspond à tous les événements de modification de la taille de flotte recommandés `farmId` pour Deadline Cloud :

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

Pour plus d'informations sur la rédaction de modèles d'événements, consultez la section Modèles d'événements dans le guide de EventBridge l'utilisateur.

Référence détaillée des événements de Deadline Cloud

Tous les événements issus AWS des services ont un ensemble commun de champs contenant des métadonnées relatives à l'événement, telles que le AWS service à l'origine de l'événement, l'heure à laquelle l'événement a été généré, le compte et la région dans lesquels l'événement a eu lieu, etc. Pour les définitions de ces champs généraux, voir la référence relative à la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

En outre, chaque événement possède un champ `detail` qui contient des données spécifiques à cet événement en particulier. La référence ci-dessous définit les champs détaillés des différents événements de Deadline Cloud.

Lorsque vous EventBridge les utilisez pour sélectionner et gérer des événements Deadline Cloud, il est utile de garder à l'esprit les points suivants :

- Le `source` champ pour tous les événements de Deadline Cloud est défini sur `aws.deadline`.

- Le champ `detail-type` indique le type d'événement.

  Par exemple, `Fleet Size Recommendation Change`.
- Le champ `detail` contient les données spécifiques à cet événement en particulier.

Pour plus d'informations sur la création de modèles d'événements permettant aux règles de correspondre aux événements de Deadline Cloud, consultez la section Modèles d'événements dans le guide de Amazon EventBridge l'utilisateur.

Pour plus d'informations sur les événements et leur EventBridge traitement, reportez-vous à la section Amazon EventBridge Événements du Guide de Amazon EventBridge l'utilisateur.

Rubriques

Événement portant sur le seuil budgétaire atteint

Vous pouvez utiliser l'événement Budget Threshold Reached pour surveiller le pourcentage d'un budget qui a été utilisé. Deadline Cloud envoie des événements lorsque le pourcentage utilisé dépasse les seuils suivants :

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

La fréquence à laquelle Deadline Cloud envoie des événements Budget Threshold Reached augmente à mesure que le budget approche de sa limite. Cette fréquence vous permet de suivre de près un budget lorsqu'il approche de sa limite et de prendre des mesures pour éviter de trop dépenser. Vous pouvez également définir vos propres seuils budgétaires. Deadline Cloud envoie un événement lorsque l'utilisation dépasse vos seuils personnalisés.

Si vous modifiez le montant d'un budget, la prochaine fois que Deadline Cloud enverra un événement Budget Threshold Reached, celui-ci sera basé sur le pourcentage actuel du budget utilisé. Par exemple, si vous ajoutez 50$ à un budget de 100$ qui a atteint sa limite, le prochain événement relatif au seuil budgétaire atteint indique que le budget est de 75 %.

Vous trouverez ci-dessous les champs détaillés de l'`Budget Threshold Reached`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Budget Threshold Reached",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "budgetId": "budget-12345678900000000000000000000000",
        "thresholdInPercent": 0
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Budget Threshold Reached`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`budgetId`

Identifiant du budget ayant atteint un seuil.

`thresholdInPercent`

Pourcentage du budget qui a été utilisé.

Événement de modification de la recommandation de taille du parc

Lorsque vous configurez votre flotte pour utiliser le dimensionnement automatique basé sur les événements, Deadline Cloud envoie des événements que vous pouvez utiliser pour gérer vos flottes. Chacun de ces événements contient des informations sur la taille actuelle et la taille demandée d'une flotte. Pour un exemple d'utilisation d'un EventBridge événement et un exemple de fonction Lambda pour gérer l'événement, consultez la section Dimensionnement automatique de votre flotte Amazon EC2 avec la fonctionnalité de recommandation d'échelle de Deadline Cloud.

L'événement de modification de la recommandation de taille de flotte est envoyé lorsque les événements suivants se produisent :

- Lorsque la taille de flotte recommandée change et oldFleetSize est différente de newFleetSize.
- Lorsque le service détecte que la taille réelle de la flotte ne correspond pas à la taille de flotte recommandée. Vous pouvez obtenir la taille réelle du parc à partir du WorkerCount dans la réponse de l' GetFleet opération. Cela peut se produire lorsqu'une instance Amazon EC2 active ne parvient pas à s'enregistrer en tant que travailleur de Deadline Cloud.

Vous trouverez ci-dessous les champs détaillés de l'`Fleet Size Recommendation Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

```
{
```

```
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Fleet Size Recommendation Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "fleetId": "fleet-12345678900000000000000000000000",
        "oldFleetSize": 1,
        "newFleetSize": 5,
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Fleet Size Recommendation Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`fleetId`

Identifiant de la flotte dont la taille doit être modifiée.

`oldFleetSize`

La taille actuelle de la flotte.

`newFleetSize`

La nouvelle taille recommandée pour la flotte.

Événement de modification du statut du cycle de vie du travail

Lorsque vous créez ou mettez à jour une tâche, Deadline Cloud définit l'état du cycle de vie pour afficher l'état de l'action la plus récente initiée par l'utilisateur.

Un événement de modification du statut du cycle de vie de la tâche est envoyé pour toute modification de l'état du cycle de vie, y compris lors de la création de la tâche.

Vous trouverez ci-dessous les champs détaillés de l'`Job Lifecycle Status Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Job Lifecycle Status Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est `aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`queueId`

Identifiant de la file d'attente contenant le travail.

`jobId`

Identifiant de la tâche.

`previousLifecycleStatus`

État du cycle de vie auquel la tâche prend fin. Ce champ n'est pas inclus lorsque vous soumettez une offre d'emploi pour la première fois.

`lifecycleStatus`

État du cycle de vie dans lequel la tâche entre.

Événement de changement de statut de Job Run

Un travail est composé de nombreuses tâches. Chaque tâche possède un statut. Le statut de toutes les tâches est combiné pour donner le statut global d'une tâche. Pour plus d'informations, consultez la section États des tâches dans Deadline Cloud dans le guide de l'utilisateur de AWS Deadline Cloud.

Un événement de changement de statut d'exécution d'une tâche est envoyé lorsque :

- Le `taskRunStatus` champ combiné change.

- La tâche est mise en attente, sauf si elle est dans l'état PRÊT.

Un événement de changement de statut d'exécution d'une tâche n'est PAS envoyé lorsque :

- L'emploi est d'abord créé. Pour surveiller la création de tâches, surveillez les événements de modification du statut du cycle de vie des tâches pour détecter les modifications.
- Le `taskRunStatusCounts` champ de la tâche change, mais le statut d'exécution de la tâche combinée reste le même.

Vous trouverez ci-dessous les champs détaillés de l'`Job Run Status Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.


```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
```

```
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
        }
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Job Run Status Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`queueId`

Identifiant de la file d'attente contenant le travail.

`jobId`

Identifiant de la tâche.

`previousTaskRunStatus`

L'état d'exécution de la tâche indique que le travail est en train de quitter.

`taskRunStatus`

État d'exécution de la tâche dans lequel le travail est en train de saisir.

`taskRunStatusCounts`

Le nombre de tâches de la tâche dans chaque État.

Evénement de changement de statut du cycle de vie

Lorsque vous créez ou mettez à jour un événement, Deadline Cloud définit l'état du cycle de vie de la tâche afin de décrire l'état de l'action la plus récente initiée par l'utilisateur.

Un événement de changement de statut du cycle de vie d'une étape est envoyé lorsque :

- Une mise à jour progressive démarre (UPDATE_IN_PROGRESS).
- La mise à jour d'une étape s'est terminée avec succès (UPDATE_SUCCEDED).
- La mise à jour d'une étape a échoué (UPDATE_FAILED).

Aucun événement n'est envoyé lorsque l'étape est créée pour la première fois. Pour surveiller la création d'étapes, surveillez les événements de modification du statut du cycle de vie des tâches pour détecter les modifications.

Vous trouverez ci-dessous les champs détaillés de l'`Step Lifecycle Status Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
```

```
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Step Lifecycle Status Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`queueId`

Identifiant de la file d'attente contenant le travail.

`jobId`

Identifiant de la tâche.

`stepId`

Identifiant de l'étape de travail en cours.

`previousLifecycleStatus`

État du cycle de vie que quitte l'étape.

`lifecycleStatus`

État du cycle de vie dans lequel l'étape entre.

Étape Exécuter un événement de changement de statut

Chaque étape d'un travail est composée de nombreuses tâches. Chaque tâche possède un statut. Les statuts des tâches sont combinés pour donner un statut global aux étapes et aux tâches.

Un événement de changement de statut Step Run est envoyé lorsque :

- Les `taskRunStatus` modifications combinées.
- L'étape est mise en attente, sauf si elle est dans l'état PRÊT.

Aucun événement n'est envoyé lorsque :

- L'étape est d'abord créée. Pour surveiller la création d'étapes, surveillez les événements de modification du statut du cycle de vie des tâches pour détecter les modifications.
- L'étape `taskRunStatusCounts` change, mais le statut d'exécution de la tâche d'étape combinée ne change pas.

Vous trouverez ci-dessous les champs détaillés de l'`Step Run Status Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.


```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
```

```
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
        }
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Step Run Status Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`queueId`

Identifiant de la file d'attente contenant le travail.

`jobId`

Identifiant de la tâche.

`stepId`

Identifiant de l'étape de travail en cours.

`previousTaskRunStatus`

État d'exécution que quitte l'étape.

`taskRunStatus`

État d'exécution dans lequel l'étape est en train d'entrer.

`taskRunStatusCounts`

Le nombre de tâches de l'étape dans chaque état.

Événement de changement de statut d'exécution de la tâche

Le `runStatus` champ est mis à jour au fur et à mesure de l'exécution de la tâche. Un événement est envoyé lorsque :

- Le statut d'exécution de la tâche change.
- La tâche est mise en attente, sauf si elle est dans l'état PRÊT.

Aucun événement n'est envoyé lorsque :

- La tâche est d'abord créée. Pour surveiller la création de tâches, surveillez les événements de modification du statut du cycle de vie des tâches pour détecter les modifications.

Vous trouverez ci-dessous les champs détaillés de l'`Task Run Status Change`événement.

Les `detail-type` champs `source` et sont inclus ci-dessous car ils contiennent des valeurs spécifiques pour les événements Deadline Cloud. Pour les définitions des autres champs de métadonnées inclus dans tous les événements, consultez la section Référence de la structure des événements dans le guide de Amazon EventBridge l'utilisateur.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Task Run Status Change",
    "source": "aws.aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
```

```
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "taskId": "task-12345678900000000000000000000000-0",
        "previousRunStatus": "RUNNING",
        "runStatus": "SUCCEEDED"
    }
}
```

`detail-type`

Identifie le type d'événement.

Pour cet événement, cette valeur est`Fleet Size Recommendation Change`.

`source`

Identifie le service qui a généré l'événement. Pour les événements Deadline Cloud, cette valeur est`aws.deadline`.

`detail`

Un objet JSON qui contient des informations sur l'événement. Le service qui génère l'événement détermine le contenu de ce champ.

Pour cet événement, ces données incluent :

`farmId`

Identifiant de la batterie de serveurs qui contient la tâche.

`queueId`

Identifiant de la file d'attente contenant le travail.

`jobId`

Identifiant de la tâche.

stepId

Identifiant de l'étape de travail en cours.

`taskId`

Identifiant de la tâche en cours d'exécution.

`previousRunStatus`

État d'exécution que quitte la tâche.

`runStatus`

État d'exécution saisi par la tâche.

Interrogation de données agrégées sur les statistiques de session à l'aide du AWS CLI

Pour suivre les coûts, analyser l'utilisation des ressources ou identifier les utilisateurs qui consomment le plus de ressources, vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour consulter les statistiques de session agrégées de vos fermes AWS Deadline Cloud (Deadline Cloud). L'API de statistiques de session fournit des données sur les coûts, le temps d'exécution et l'utilisation que vous pouvez regrouper selon différentes dimensions telles que la file d'attente, le parc, le type d'instance ou l'utilisateur.

L'interrogation des statistiques de session est un processus asynchrone. Tout d'abord, vous lancez une demande d'agrégation, puis vous récupérez les résultats à l'aide de l'ID d'agrégation.

Lancer une demande d'agrégation

Pour démarrer une demande d'agrégation, exécutez la `start-sessions-statistics-aggregation` commande. L'exemple suivant regroupe les statistiques par ID utilisateur pour une file d'attente spécifique. Remplacez le *`placeholder`* texte par vos informations.

```
aws deadline start-sessions-statistics-aggregation \
    --farm-id farm-id \
    --resource-ids '{"queueIds":["queue-id"]}' \
    --start-time 2025-11-24T10:00:00Z \
    --end-time 2025-11-25T18:00:00Z \
    --group-by '["USER_ID"]' \
    --period HOURLY \
    --statistics '["SUM"]' \
    --timezone UTC-08:00 \
    --region region-name
```

Vous pouvez regrouper les statistiques selon d'autres dimensions telles que `QUEUE_IDFLEET_ID`,`JOB_ID`,`INSTANCE_TYPE`, ou`LICENSE_PRODUCT`. Pour plus d'informations sur tous les paramètres disponibles, consultez start-sessions-statistics-aggregationla référence des AWS CLI commandes.

La réponse contient un identifiant d'agrégation :

```
{
```

```
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Récupération des résultats

Exécutez la `get-sessions-statistics-aggregation` commande avec l'ID d'agrégation pour récupérer les résultats. Remplacez le *`placeholder`* texte par vos informations.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

L'exemple suivant montre une réponse lorsque vous regroupez les statistiques par ID utilisateur. Le `userId` champ contient un UUID que vous devez associer à un nom d'utilisateur pour identifier l'utilisateur :

```
{
    "statistics": [
        {
            "userId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
            "count": 1,
            "costInUsd": {
                "sum": 0.0
            },
            "runtimeInSeconds": {
                "sum": 53.773
            },
            "aggregationStartTime": "2025-11-24T22:00:00Z",
            "aggregationEndTime": "2025-11-24T23:00:00Z"
        }
    ],
    "status": "COMPLETED"
}
```

Pour trouver le nom d'utilisateur associé à un`userId`, consultezthe section called “Récupération des métadonnées utilisateur à l'aide de l'ID utilisateur”.

Pour plus d'informations sur l'API, consultez le document de référence de l'API Deadline Cloud.

Rubriques

- the section called “Récupération des métadonnées utilisateur à l'aide de l'ID utilisateur”

Récupération des métadonnées et des attributs utilisateur à l'aide de l'ID utilisateur dans un magasin d'identités

> **Note**
>
> Cette procédure s'applique également au `createdBy` champ renvoyé par l'SearchJobsAPI, qui utilise le même format d'ID utilisateur.

Le `userId` champ des statistiques de session contient l'une des valeurs suivantes :

- Un ARN de rôle Gestion des identités et des accès AWS (IAM), par exemple :`arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`.
- Un ID utilisateur (UUID) du IAM Identity Center, par exemple :. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Pour le rôle IAM ARNs, le nom d'utilisateur est visible dans l'ARN lui-même. Pour les utilisateurs d'IAM Identity Center IDs, vous pouvez rechercher le nom d'utilisateur à l'aide de l'API IAM Identity Center Identity Store.

Pour identifier le nom d'utilisateur associé à un ID utilisateur IAM Identity Center, procédez comme suit. Avant de commencer, récupérez l'identifiant Identity Store dans les paramètres de votre IAM Identity Center. Pour de plus amples informations, veuillez consulter the section called “Trouver votre identifiant Identity Store”.

Pour mapper un ID utilisateur

1. Exécutez la commande suivante, en *`IdentityStoreId`* remplaçant par votre identifiant Identity Store et *`userUUID`* par la réponse des statistiques `userId` de session :

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Passez en revue la réponse, qui inclut le nom d'utilisateur :

```
{
    "UserName": "jdoe",
    "UserId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
    "Name": {
        "FamilyName": "Doe",
        "GivenName": "Jane"
    },
    "DisplayName": "Jane Doe",
    "Emails": [{
        "Value": "jdoe@example.com",
        "Type": "work",
        "Primary": true
    }],
    "IdentityStoreId": "d-xxxxxxxxxx"
}
```

Trouver votre identifiant Identity Store

Pour associer un utilisateur IDs à un nom d'utilisateur, vous avez besoin de l'Identity Store ID. Vous pouvez trouver l'identifiant Identity Store à l'aide de la console IAM Identity Center ou du AWS CLI.

Console

Pour trouver votre identifiant Identity Store à l'aide de la console, procédez comme suit.

1. Connectez-vous à la console AWS de gestion et ouvrez la console IAM Identity Center.
2. Dans le panneau de navigation, sélectionnez Settings (Paramètres).
3. Copiez la valeur de l'identifiant IAM Identity Center Identity Store. Le format est `d-xxxxxxxxxx`.

AWS CLI

Exécutez la commande suivante, en la *`region-name`* remplaçant par la région dans laquelle votre instance IAM Identity Center est configurée :

```
aws sso-admin list-instances --region region-name
```

La réponse inclut les éléments `IdentityStoreId` suivants :

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
    ]
}
```

Vérification du mappage des utilisateurs

Après avoir mappé un ID utilisateur à un nom d'utilisateur, vous pouvez vérifier dans la console IAM Identity Center que l'ID utilisateur correspond à l'utilisateur attendu. Pour vérifier le mappage des utilisateurs, procédez comme suit.

1. Connectez-vous à la console AWS de gestion et ouvrez la console IAM Identity Center.
2. Dans le panneau de navigation, choisissez utilisateurs.
3. Choisissez le nom d'utilisateur dans la AWS CLI réponse.
4. Dans la section Informations générales, vérifiez que le nom d'utilisateur correspond aux statistiques `userId` de votre session.

Ressources supplémentaires

- Guide de l'utilisateur d'IAM Identity Center
- Référence de l'API IAM Identity Center Identity Store

Sécurité dans Deadline Cloud

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui s'exécute Services AWS dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Third-party les auditeurs testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Deadline Cloud, voir Services AWS Portée par programme de conformitéServices AWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par Service AWS ce que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation Deadline Cloud. Les rubriques suivantes expliquent comment procéder à la configuration Deadline Cloud pour atteindre vos objectifs de sécurité et de conformité. Vous apprenez également à utiliser d'autres outils Services AWS qui vous aident à surveiller et à sécuriser vos Deadline Cloud ressources.

Rubriques

- Accès AWS Deadline Cloud en utilisant un point de terminaison d'interface (AWS PrivateLink)
- Environnements réseau restreints
- Bonnes pratiques de sécurité pour Deadline Cloud

Protection des données dans Deadline Cloud

Le modèle de responsabilité AWS partagée Le modèle de s'applique à la protection des données dans AWS Deadline Cloud. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l’authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu’Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d’informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Deadline Cloud ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Les données saisies dans les champs de nom des modèles de Deadline Cloud tâches peuvent également être incluses dans les journaux de facturation ou de diagnostic et ne doivent pas contenir d'informations confidentielles ou sensibles.

Rubriques

Chiffrement au repos

AWS Deadline Cloud protège les données sensibles en les chiffrant au repos à l'aide des clés de chiffrement stockées dans AWS Key Management Service (AWS KMS). Le chiffrement au repos est disponible partout Régions AWS où il Deadline Cloud est disponible.

Le chiffrement des données signifie que les données sensibles enregistrées sur les disques ne sont pas lisibles par un utilisateur ou une application sans clé valide. Seule une partie disposant d'une clé gérée valide peut déchiffrer les données.

Deadline Cloud supprime les volumes Amazon Elastic Block Store lorsque les instances de Fleet Worker gérées par le service prennent fin.

Pour plus d'informations sur AWS KMS les Deadline Cloud utilisations du chiffrement des données au repos, consultezGestion des clés.

Chiffrement en transit

Pour les données en transit, AWS Deadline Cloud utilise le protocole TLS (Transport Layer Security) 1.2 ou 1.3 pour chiffrer les données envoyées entre le service et les employés. Nous exigeons TLS 1.2 et recommandons TLS 1.3. En outre, si vous utilisez un cloud privé virtuel (VPC), vous pouvez l'utiliser AWS PrivateLink pour établir une connexion privée entre votre VPC et. Deadline Cloud

Gestion des clés

Lorsque vous créez un nouveau parc de serveurs, vous pouvez choisir l'une des clés suivantes pour chiffrer les données de votre parc de serveurs :

- AWS clé KMS détenue : type de chiffrement par défaut si vous ne spécifiez pas de clé lors de la création du parc de serveurs. La clé KMS appartient à AWS Deadline Cloud. Vous ne pouvez pas afficher, gérer ou utiliser les clés que vous AWS possédez. Cependant, vous n'avez aucune action à effectuer pour protéger les clés qui chiffrent vos données. Pour plus d'informations, consultez la section sur les clés AWS détenues dans le guide du AWS Key Management Service développeur.
- Clé KMS gérée par le client : vous spécifiez une clé gérée par le client lorsque vous créez un parc de serveurs. Tout le contenu de la ferme est chiffré à l'aide de la clé KMS. La clé est stockée dans votre compte et vous la créez, la détenez et la gérez. AWS KMS Des frais s'appliquent. Vous avez le contrôle total de la clé KMS. Vous pouvez effectuer des tâches telles que :
  - Établissement et mise à jour de politiques clés
  - Établissement et gestion des politiques IAM et des octrois
  - Activation et désactivation des stratégies de clé
  - Ajout de balises
  - Création d’alias de clé

  Vous ne pouvez pas faire pivoter manuellement une clé appartenant à un client utilisée avec une Deadline Cloud ferme. La rotation automatique de la clé est prise en charge.

  Pour plus d'informations, consultez la section Clés détenues par le client dans le guide du AWS Key Management Service développeur.

  Pour créer une clé gérée par le client, suivez les étapes de création de clés gérées par le client symétriques dans le guide du AWS Key Management Service développeur.

Comment ? Deadline Cloud utilise AWS KMS bourses

Deadline Cloud nécessite une autorisation pour utiliser votre clé gérée par le client. Lorsque vous créez un parc chiffré à l'aide d'une clé gérée par le client, vous Deadline Cloud créez une autorisation en votre nom en envoyant une `CreateGrant` demande d'accès à la clé KMS que vous avez spécifiée. AWS KMS

Deadline Cloud utilise plusieurs subventions. Chaque autorisation est utilisée par un service différent Deadline Cloud qui doit chiffrer ou déchiffrer vos données. Deadline Cloud utilise également des subventions pour permettre l'accès à d'autres AWS services utilisés pour stocker des données en votre nom, tels qu'Amazon Simple Storage Service, Amazon Elastic Block Store ou OpenSearch.

Les subventions qui permettent Deadline Cloud de gérer les machines d'un parc géré par des services incluent un numéro de Deadline Cloud compte et un rôle au `GranteePrincipal` lieu d'un directeur de service. Bien que cela ne soit pas habituel, cela est nécessaire pour chiffrer les volumes Amazon EBS destinés aux employés des flottes gérées par des services à l'aide de la clé KMS gérée par le client spécifiée pour le parc de serveurs.

Politique de clé gérée par le client

Les stratégies de clés contrôlent l’accès à votre clé gérée par le client. Chaque clé doit avoir exactement une politique clé contenant des instructions qui déterminent qui peut utiliser la clé et comment ils peuvent l'utiliser. Lorsque vous créez votre clé gérée par le client, vous pouvez définir une politique clé. Pour plus d’informations, consultez Gestion de l’accès aux clés gérées par le client dans le Guide du développeur AWS Key Management Service .

Politique IAM minimale pour CreateFarm

Pour utiliser votre clé gérée par le client afin de créer des fermes à l'aide de la console ou de l'opération d'`CreateFarm`API, les opérations d' AWS KMS API suivantes doivent être autorisées :

- `kms:CreateGrant` : ajoute une attribution à une clé gérée par le client. Accorde l'accès à la console à une AWS KMS clé spécifiée. Pour plus d'informations, consultez la section Utilisation des subventions dans le guide du AWS Key Management Service développeur.
- `kms:Decrypt`— Permet Deadline Cloud de déchiffrer les données de la ferme.
- `kms:DescribeKey`— Fournit les informations clés gérées par le client Deadline Cloud pour permettre de valider la clé.
- `kms:GenerateDataKey`— Permet de chiffrer Deadline Cloud les données à l'aide d'une clé de données unique.

La déclaration de politique suivante accorde les autorisations nécessaires à l'`CreateFarm`opération.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Politique IAM minimale pour les opérations en lecture seule

Utiliser votre clé gérée par le client pour des Deadline Cloud opérations en lecture seule, telles que l'obtention d'informations sur les fermes, les files d'attente et les flottes. Les opérations AWS KMS d'API suivantes doivent être autorisées :

- `kms:Decrypt`— Permet Deadline Cloud de déchiffrer les données de la ferme.
- `kms:DescribeKey`— Fournit les informations clés gérées par le client Deadline Cloud pour permettre de valider la clé.

La déclaration de politique suivante accorde les autorisations nécessaires pour les opérations en lecture seule.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Politique IAM minimale pour les opérations de lecture-écriture

Utiliser votre clé gérée par le client pour les Deadline Cloud opérations de lecture-écriture, telles que la création et la mise à jour de parcs, de files d'attente et de flottes. Les opérations AWS KMS d'API suivantes doivent être autorisées :

- `kms:Decrypt`— Permet Deadline Cloud de déchiffrer les données de la ferme.
- `kms:DescribeKey`— Fournit les informations clés gérées par le client Deadline Cloud pour permettre de valider la clé.
- `kms:GenerateDataKey`— Permet de chiffrer Deadline Cloud les données à l'aide d'une clé de données unique.

La déclaration de politique suivante accorde les autorisations nécessaires à l'`CreateFarm`opération.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Surveillance de vos clés de chiffrement

Lorsque vous utilisez une clé gérée par le AWS KMS client pour vos Deadline Cloud fermes, vous pouvez utiliser AWS CloudTrailAmazon CloudWatch Logs pour suivre les demandes Deadline Cloud envoyées à AWS KMS.

CloudTrail événement pour les subventions

L'exemple d' CloudTrail événement suivant se produit lorsque des autorisations sont créées, généralement lorsque vous appelez l'`CreateFleet`opération `CreateFarm` `CreateMonitor`, ou.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


CloudTrail événement de déchiffrement

L'exemple d' CloudTrail événement suivant se produit lors du déchiffrement de valeurs à l'aide de la clé KMS gérée par le client.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
```

```
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

CloudTrail événement pour le chiffrement

L'exemple d' CloudTrail événement suivant se produit lors du chiffrement de valeurs à l'aide de la clé KMS gérée par le client.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 32,
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
```

```
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Supprimer une clé KMS gérée par le client

La suppression d'une clé KMS gérée par le client dans AWS Key Management Service (AWS KMS) est destructrice et potentiellement dangereuse. Il supprime de manière irréversible le contenu clé et toutes les métadonnées associées à la clé. Après la suppression d'une clé KMS gérée par le client, vous ne pouvez plus déchiffrer les données chiffrées par cette clé. La suppression de la clé signifie que les données deviennent irrécupérables.

C'est pourquoi les AWS KMS clients disposent d'un délai d'attente pouvant aller jusqu'à 30 jours avant de supprimer la clé KMS. La période d'attente par défaut est de 30 jours.

À propos de la période d'attente

Comme il est destructeur et potentiellement dangereux de supprimer une clé KMS gérée par le client, nous vous demandons de définir un délai d'attente de 7 à 30 jours. La période d'attente par défaut est de 30 jours.

Cependant, la période d'attente réelle peut être supérieure de 24 heures à la période que vous avez planifiée. Pour obtenir la date et l'heure réelles auxquelles la clé sera supprimée, utilisez l'DescribeKeyopération. Vous pouvez également voir la date de suppression planifiée d'une clé dans

la AWS KMS console sur la page détaillée de la clé, dans la section Configuration générale. Notez le fuseau horaire.

Pendant la période d'attente, le statut de la clé gérée par le client et l'état de la clé sont En attente de suppression.

- Une clé KMS gérée par le client en attente de suppression ne peut être utilisée dans aucune opération cryptographique.
- AWS KMS ne fait pas pivoter les clés de sauvegarde des clés KMS gérées par le client en attente de suppression.

Pour plus d'informations sur la suppression d'une clé KMS gérée par le client, consultez la section Suppression des clés principales du client dans le guide du AWS Key Management Service développeur.

Inter-network confidentialité du trafic

AWS Deadline Cloud prend en charge Amazon Virtual Private Cloud (Amazon VPC) pour sécuriser les connexions. Amazon VPC fournit des fonctionnalités que vous pouvez utiliser pour renforcer et surveiller la sécurité de votre cloud privé virtuel (VPC).

Vous pouvez configurer un parc géré par le client (CMF) avec des instances Amazon Elastic Compute Cloud (Amazon EC2) exécutées au sein d'un VPC. En déployant les points de terminaison Amazon VPC à utiliser AWS PrivateLink, le trafic entre les travailleurs de votre CMF et le point de Deadline Cloud terminaison reste au sein de votre VPC. En outre, vous pouvez configurer votre VPC pour restreindre l'accès Internet à vos instances.

Dans les flottes gérées par des services, les employés ne sont pas joignables depuis Internet, mais ils ont accès à Internet et se connectent au Deadline Cloud service via Internet. Chaque flotte gérée par des services fonctionne sur son propre réseau isolé, et les instances de travail restent dédiées aux clients individuels.

Refus

AWS Deadline Cloud collecte certaines informations opérationnelles pour nous aider à nous développer et à nous améliorer Deadline Cloud. Les données collectées incluent des éléments tels que votre identifiant de AWS compte et votre identifiant d'utilisateur, afin que nous puissions vous identifier correctement en cas de problème avec le Deadline Cloud. Nous collectons également Deadline Cloud des informations spécifiques, telles que les identifiants de ressource (un FarmID ou

un QueueID le cas échéant), le nom du produit (par exemple, JobAttachments WorkerAgent, et plus encore) et la version du produit.

Vous pouvez choisir de ne pas participer à cette collecte de données à l'aide de la configuration de l'application. Chaque ordinateur interagissant avec Deadline Cloud, à la fois les postes de travail des clients et les employés du parc, doit se désinscrire séparément.

Deadline Cloud moniteur - ordinateur de bureau

Deadline Cloud monitor - desktop collecte des informations opérationnelles, telles que les pannes et l'ouverture de l'application, pour nous aider à savoir quand vous rencontrez des problèmes avec l'application. Pour refuser la collecte de ces informations opérationnelles, rendez-vous sur la page des paramètres et désactivez Activer la collecte de données pour mesurer les performances de Deadline Cloud Monitor.

Une fois que vous vous êtes désinscrit, le moniteur de bureau n'envoie plus les données opérationnelles. Toutes les données précédemment collectées sont conservées et peuvent toujours être utilisées pour améliorer le service. Pour plus d'informations, consultez FAQ sur la confidentialité des données.

AWS Deadline Cloud CLI et outils

La AWS Deadline Cloud CLI, les soumetteurs et l'agent de travail collectent tous des informations opérationnelles, telles que les cas de plantage et le moment où les tâches sont soumises, afin de nous aider à savoir quand vous rencontrez des problèmes avec ces applications. Pour refuser la collecte de ces informations opérationnelles, utilisez l'une des méthodes suivantes :

- Dans le terminal, entrez**`deadline config set telemetry.opt_out true`**.

  Cela désactivera la CLI, les soumetteurs et l'agent de travail lors de l'exécution en tant qu'utilisateur actuel.
- Lors de l'installation de l' Deadline Cloud agent de travail, ajoutez l'argument de ligne de **`--telemetry-opt-out`** commande. Par exemple, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.
- Avant d'exécuter l'agent de travail, la CLI ou l'émetteur, définissez une variable d'environnement : **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Une fois que vous vous êtes désinscrit, les Deadline Cloud outils n'envoient plus les données opérationnelles. Toutes les données précédemment collectées sont conservées et peuvent toujours

être utilisées pour améliorer le service. Pour plus d'informations, consultez FAQ sur la confidentialité des données.

Identity and Access Management dans Deadline Cloud

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources de Deadline Cloud. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution des problèmes AWS Deadline Identité et accès au cloud)
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir Comment Deadline Cloud fonctionne avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir Identity-based exemples de politiques pour Deadline Cloud)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-based politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Resource-based politiques

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez Politiques de contrôle des ressources (RCP) dans le Guide de l'utilisateur AWS Organizations .

- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment Deadline Cloud fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Deadline Cloud, découvrez quelles fonctionnalités IAM peuvent être utilisées avec Deadline Cloud.

Fonctionnalités IAM que vous pouvez utiliser avec AWS Deadline Cloud

Fonctionnalité IAM	Support de Deadline Cloud
Identity-based politiques	Oui
Resource-based politiques	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Transmission des sessions d'accès (FAS)	Oui
Rôles de service	Oui

Fonctionnalité IAM	Support de Deadline Cloud
Service-linked rôles	Non

Pour obtenir une vue d'ensemble de la façon dont Deadline Cloud et les autres services Services AWS fonctionnent avec la plupart des fonctionnalités IAM, consultez les AWS services compatibles avec IAM dans le guide de l'utilisateur IAM.

Identity-based politiques pour Deadline Cloud

Prend en charge les politiques basées sur l’identité : oui

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous pouvez associer à une identité, telle qu'un utilisateur IAM, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Définition d’autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l’utilisateur IAM.

Avec les politiques IAM basées sur l’identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l’utilisateur IAM.

Identity-based exemples de politiques pour Deadline Cloud

Pour consulter des exemples de politiques basées sur l'identité de Deadline Cloud, consultez. Identity-based exemples de politiques pour Deadline Cloud

Resource-based politiques dans Deadline Cloud

Prend en charge les politiques basées sur les ressources : non

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l’accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique,

cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Actions politiques pour Deadline Cloud

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions de Deadline Cloud, consultez la section Actions définies par AWS Deadline Cloud dans la référence d'autorisation de service.

Les actions politiques dans Deadline Cloud utilisent le préfixe suivant avant l'action :

```
deadline
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Pour consulter des exemples de politiques basées sur l'identité de Deadline Cloud, consultez. Identity-based exemples de politiques pour Deadline Cloud

Ressources relatives aux politiques pour Deadline Cloud

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Deadline Cloud et leurs ARN, consultez la section Ressources définies par AWS Deadline Cloud dans la référence d'autorisation de service. Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez la section Actions définies par AWS Deadline Cloud.

Pour consulter des exemples de politiques basées sur l'identité de Deadline Cloud, consultez. Identity-based exemples de politiques pour Deadline Cloud

Clés de conditions de politique pour Deadline Cloud

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition de Deadline Cloud, voir Clés de condition pour AWS Deadline Cloud dans la référence d'autorisation de service. Pour savoir avec quelles actions et

ressources vous pouvez utiliser une clé de condition, consultez la section Actions définies par AWS Deadline Cloud.

Pour consulter des exemples de politiques basées sur l'identité de Deadline Cloud, consultez. Identity-based exemples de politiques pour Deadline Cloud

ACL dans Deadline Cloud

Prend en charge les ACL : non

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec Deadline Cloud

Prise en charge d'ABAC (balises dans les politiques) : Oui

Attribute-based le contrôle d'accès (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction d'attributs appelés balises. Vous pouvez associer des balises aux entités et aux AWS ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez Définition d'autorisations avec l'autorisation ABAC dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Deadline Cloud

Prend en charge les informations d'identification temporaires : oui

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM et Services AWS compatibles avec IAM dans le Guide de l'utilisateur IAM.

Transférer les sessions d'accès pour Deadline Cloud

Prend en charge les sessions d'accès direct (FAS) : oui

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section Sessions de transmission d'accès.

Rôles de service pour Deadline Cloud

Prend en charge les rôles de service : oui

Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber les fonctionnalités de Deadline Cloud. Modifiez les rôles de service uniquement lorsque Deadline Cloud fournit des instructions à cet effet.

Service-linked rôles pour Deadline Cloud

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut assumer le rôle d'effectuer une action en votre nom. Service-linked les rôles apparaissent dans votre fichier Compte AWS et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne des Service-linked rôles. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Identity-based exemples de politiques pour Deadline Cloud

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources Deadline Cloud. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM (console) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Deadline Cloud, y compris le format des ARN pour chacun des types de ressources, consultez la section Actions, ressources et clés de condition pour AWS Deadline Cloud dans la référence d'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Deadline Cloud
- Politique d'accès à la console
- Politique de soumission des tâches à une file d'attente
- Politique autorisant la création d'un point de terminaison de licence
- Politique autorisant la surveillance d'une file d'attente de ferme spécifique

Bonnes pratiques en matière de politiques

Identity-based les politiques déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Deadline Cloud dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire

davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.

- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Deadline Cloud

Pour accéder à la console AWS Deadline Cloud, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et de consulter les

informations relatives aux ressources de Deadline Cloud présentes dans votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la console Deadline Cloud, associez également le Deadline Cloud *ConsoleAccess* ou la politique *ReadOnly* AWS gérée aux entités. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Politique d'accès à la console

Pour accorder l'accès à toutes les fonctionnalités de la console Deadline Cloud, associez cette politique d'identité à un utilisateur ou à un rôle auquel vous souhaitez bénéficier d'un accès complet.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
```

```
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
            "vpc-lattice:GetResourceConfiguration",
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ValidateDependencies",
```

```
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "RoleSelection",
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles",
"iam:ListAttachedRolePolicies"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
        "Resource": "*"
    },
```

```
    {
        "Sid": "OrganizationAndIdentityCenterIdentification",
        "Effect": "Allow",
        "Action": [
            "sso:ListDirectoryAssociations",
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "sso:DescribeRegisteredRegions",
            "sso:GetManagedApplicationInstance",
            "sso:GetSharedSsoConfiguration",
            "sso:ListInstances",
            "sso:GetApplicationAssignmentConfiguration",
            "sso:GetSSOStatus",
            "sso:ListRegions",
            "sso:DescribeRegion"
    ],
        "Resource": "*"
    },
    {
        "Sid": "ManagedDeadlineCloudIDCApplication",
        "Effect": "Allow",
        "Action": [
            "sso:CreateApplication",
            "sso:PutApplicationAssignmentConfiguration",
            "sso:PutApplicationAuthenticationMethod",
            "sso:PutApplicationGrant",
            "sso:DeleteApplication",
            "sso:UpdateApplication"
    ],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseSecret",
        "Effect": "Allow",
        "Action": ["secretsmanager:ListSecrets"],
        "Resource": "*"
    },
    {
        "Sid": "DeadlineMembershipActions",
        "Effect": "Allow",
        "Action": [
```

```
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
            "deadline:ListFarmMembers",
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
```

```
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
            "deadline:UpdateLimit",
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:GetMonitorSettings",
            "deadline:UpdateMonitorSettings"
        ],
        "Resource": ["*"]
      }]
}
```

Politique de soumission des tâches à une file d'attente

Dans cet exemple, vous créez une politique limitée qui accorde l'autorisation de soumettre des tâches à une file d'attente spécifique dans un parc spécifique.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
}
```

Politique autorisant la création d'un point de terminaison de licence

Dans cet exemple, vous créez une politique délimitée qui accorde les autorisations requises pour créer et gérer les points de terminaison de licence. Utilisez cette politique pour créer le point de terminaison de licence pour le VPC associé à votre parc de serveurs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
```

```
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
    }]
}
```

Politique autorisant la surveillance d'une file d'attente de ferme spécifique

Dans cet exemple, vous créez une politique limitée qui autorise le suivi des tâches dans une file d'attente spécifique pour un parc de serveurs spécifique.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
            "deadline:GetSessionAction"
```

```
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
    }]
}
```

AWS politiques gérées pour Deadline Cloud

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : AWSDeadlineCloud-FleetWorker

Vous pouvez associer la `AWSDeadlineCloud-FleetWorker` politique à vos identités Gestion des identités et des accès AWS (IAM).

Cette politique accorde aux travailleurs de cette flotte les autorisations nécessaires pour se connecter au service et en recevoir des tâches.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet aux directeurs de gérer les travailleurs d'une flotte.

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-FleetWorkerle guide de référence des politiques gérées par AWS.

AWS politique gérée : AWSDeadlineCloud-WorkerHost

Vous pouvez associer la politique `AWSDeadlineCloud-WorkerHost` à vos identités IAM.

Cette politique accorde les autorisations nécessaires pour se connecter initialement au service. Il peut être utilisé comme profil d'instance Amazon Elastic Compute Cloud (Amazon EC2).

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet à l'utilisateur de créer des travailleurs, d'assumer le rôle de flotte pour les travailleurs et d'appliquer des balises aux travailleurs

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-WorkerHostle guide de référence des politiques gérées par AWS.

AWS politique gérée : AWSDeadlineCloud-UserAccessFarms

Vous pouvez associer la politique `AWSDeadlineCloud-UserAccessFarms` à vos identités IAM.

Cette politique permet aux utilisateurs d'accéder aux données des fermes en fonction des fermes dont ils sont membres et de leur niveau d'adhésion.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet à l'utilisateur d'accéder aux données de la ferme.
- `ec2`— Permet aux utilisateurs de consulter les informations relatives aux types d'instances Amazon EC2.
- `identitystore`— Permet aux utilisateurs de voir les noms des utilisateurs et des groupes.

- `kms`— Permet aux utilisateurs de configurer AWS Key Management Service (AWS KMS) des clés gérées par le client pour leur instance AWS IAM Identity Center (IAM Identity Center).

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-UserAccessFarmsle guide de référence des politiques gérées par AWS.

AWS politique gérée : AWSDeadlineCloud-UserAccessFleets

Vous pouvez associer la politique `AWSDeadlineCloud-UserAccessFleets` à vos identités IAM.

Cette politique permet aux utilisateurs d'accéder aux données de la flotte en fonction des fermes dont ils sont membres et de leur niveau d'adhésion.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet à l'utilisateur d'accéder aux données de la ferme.
- `ec2`— Permet aux utilisateurs de consulter les informations relatives aux types d'instances Amazon EC2.
- `identitystore`— Permet aux utilisateurs de voir les noms des utilisateurs et des groupes.

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-UserAccessFleetsle guide de référence des politiques gérées par AWS.

AWS politique gérée : AWSDeadlineCloud-UserAccessJobs

Vous pouvez associer la politique `AWSDeadlineCloud-UserAccessJobs` à vos identités IAM.

Cette politique permet aux utilisateurs d'accéder aux données relatives aux emplois en fonction des fermes dont ils sont membres et de leur niveau d'adhésion.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet à l'utilisateur d'accéder aux données de la ferme.
- `ec2`— Permet aux utilisateurs de consulter les informations relatives aux types d'instances Amazon EC2.
- `identitystore`— Permet aux utilisateurs de voir les noms des utilisateurs et des groupes.

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-UserAccessJobsle guide de référence des politiques gérées par AWS.

AWS politique gérée : AWSDeadlineCloud-UserAccessQueues

Vous pouvez associer la politique `AWSDeadlineCloud-UserAccessQueues` à vos identités IAM.

Cette politique permet aux utilisateurs d'accéder aux données des files d'attente en fonction des fermes dont ils sont membres et de leur niveau d'adhésion.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes :

- `deadline`— Permet à l'utilisateur d'accéder aux données de la ferme.
- `ec2`— Permet aux utilisateurs de consulter les informations relatives aux types d'instances Amazon EC2.
- `identitystore`— Permet aux utilisateurs de voir les noms des utilisateurs et des groupes.

Pour obtenir une liste JSON des détails de la politique, consultez AWSDeadlineCloud-UserAccessQueuesle guide de référence des politiques gérées par AWS.

Mises à jour de Deadline Cloud AWS stratégies gérées

Consultez les détails des mises à jour des politiques AWS gérées pour Deadline Cloud depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique des documents de Deadline Cloud.

Modifier	Description	Date
AWSDeadlineCloud-UserAccessFarms— Modification	Deadline Cloud a ajouté une nouvelle action `kms:Decrypt` afin que vous puissiez utiliser une clé AWS KMS gérée par le client avec votre instance IAM Identity Center.	22 décembre 2025

Modifier	Description	Date
AWSDeadlineCloud-WorkerHost— Modification	Deadline Cloud a ajouté `deadline:ListTagsForResource` de nouvelles actions `deadline:TagResource` et vous permet d'ajouter et de consulter les tags associés aux employés de votre flotte.	30 mai 2025
AWSDeadlineCloud-UserAccessFarms— Modification AWSDeadlineCloud-UserAccessJobs— Modification AWSDeadlineCloud-UserAccessQueues— Modification	Deadline Cloud a ajouté de nouvelles actions `deadline:GetJobTemplate` et vous `deadline:ListJobParameterDefinitions` permet de soumettre à nouveau des tâches.	7 octobre 2024
Deadline Cloud a commencé à suivre les modifications	Deadline Cloud a commencé à suivre les modifications apportées à ses politiques AWS gérées.	2 avril 2024

Rôles du service

Comment Deadline Cloud utilise les rôles de service IAM

Deadline Cloud assume automatiquement les rôles IAM et fournit des informations d'identification temporaires aux employés, aux tâches et au moniteur Deadline Cloud. Cette approche élimine la gestion manuelle des informations d'identification tout en préservant la sécurité grâce à un contrôle d'accès basé sur les rôles.

Lorsque vous créez des moniteurs, des flottes et des files d'attente, vous spécifiez les rôles IAM que Deadline Cloud assume en votre nom. Les travailleurs et le moniteur Deadline Cloud reçoivent

ensuite des informations d'identification temporaires provenant de ces rôles pour y accéder Services AWS.

Rôle de la flotte

Configurez un rôle dans le parc pour donner aux employés de Deadline Cloud les autorisations dont ils ont besoin pour recevoir le travail et rendre compte de l'avancement de ce travail.

Il n'est généralement pas nécessaire de configurer ce rôle vous-même. Ce rôle peut être créé pour vous dans la console Deadline Cloud afin d'inclure les autorisations nécessaires. Utilisez le guide suivant pour comprendre les spécificités de ce rôle à des fins de dépannage.

Lorsque vous créez ou mettez à jour des flottes par programmation, spécifiez l'ARN du rôle de flotte à l'aide des opérations de l'API `CreateFleet` or`UpdateFleet`.

À quoi sert le rôle de la flotte

Le rôle de flotte fournit aux travailleurs les autorisations nécessaires pour :

- Recevez de nouveaux travaux et signalez l'avancement des travaux en cours au service Deadline Cloud
- Gérez le cycle de vie et le statut des employés
- Enregistrer les événements dans Amazon CloudWatch Logs pour les journaux des employés

Configurez la politique de confiance relative aux rôles de la flotte

Votre rôle dans la flotte doit faire confiance au service Deadline Cloud et être limité à votre ferme spécifique.

À titre de bonne pratique, la politique de confiance devrait inclure des conditions de sécurité pour la protection des adjoints confus. Pour en savoir plus sur la protection contre Confused Deputy, consultez Confused Deputy dans le guide de l'utilisateur de Deadline Cloud.

- `aws:SourceAccount`garantit que seules ses ressources Compte AWS peuvent assumer ce rôle.
- `aws:SourceArn`limite l'attribution des rôles à une ferme Deadline Cloud spécifique.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

Associer les autorisations du rôle Fleet

Associez la politique AWS gérée suivante au rôle de votre flotte :

AWSDeadlineCloud-FleetWorker

Cette politique gérée fournit des autorisations pour :

- `deadline:AssumeFleetRoleForWorker`- Permet aux employés d'actualiser leurs informations d'identification.
- `deadline:UpdateWorker`- Permet aux travailleurs de mettre à jour leur statut (par exemple, en le faisant passer à STOPPÉ à la sortie).
- `deadline:UpdateWorkerSchedule`- Pour obtenir du travail et rendre compte des progrès réalisés.
- `deadline:BatchGetJobEntity`- Pour récupérer des informations sur le travail.
- `deadline:AssumeQueueRoleForWorker`- Pour accéder aux informations d'identification du rôle de file d'attente lors de l'exécution de la tâche.

Ajouter des autorisations KMS pour les fermes chiffrées

Si votre ferme a été créée à l'aide d'une clé KMS, ajoutez ces autorisations à votre rôle dans la flotte pour garantir que le travailleur puisse accéder aux données chiffrées de la ferme.

Les autorisations KMS ne sont nécessaires que si votre parc possède une clé KMS associée. La `kms:ViaService` condition doit utiliser le format`deadline.{region}.amazonaws.com`.

Lors de la création d'une flotte, un groupe de CloudWatch journaux Logs est créé pour cette flotte. Les autorisations du travailleur sont utilisées par le service Deadline Cloud pour créer un flux de journal spécifique à ce travailleur en particulier. Une fois le programme de travail configuré et exécuté, il utilisera ces autorisations pour envoyer les événements du journal directement à CloudWatch Logs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
```

```
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

Modifier le rôle de la flotte

Les autorisations relatives au rôle de flotte ne sont pas personnalisables. Les autorisations décrites sont toujours obligatoires et l'ajout d'autorisations supplémentaires n'a aucun effet.

Customer-managed rôle d'hôte de flotte

Configurez un WorkerHost rôle si vous utilisez des flottes gérées par le client sur des instances Amazon EC2 ou des hôtes sur site.

À quoi sert le WorkerHost rôle

Ce WorkerHost rôle permet de dynamiser les travailleurs des hébergeurs de flotte gérés par le client. Il fournit les autorisations minimales nécessaires à un hôte pour :

- Créer un travailleur dans Deadline Cloud
- Assumez le rôle de flotte pour récupérer les informations d'identification opérationnelles
- Étiquetez les travailleurs avec des étiquettes de flotte (si la propagation des balises est activée)

Configurer les autorisations de WorkerHost rôle

Associez la politique AWS gérée suivante à votre WorkerHost rôle :

AWSDeadlineCloud-WorkerHost

Cette politique gérée fournit des autorisations pour :

- `deadline:CreateWorker`- Permet à l'hôte d'enregistrer un nouveau travailleur.

- `deadline:AssumeFleetRoleForWorker`- Permet à l'hôte d'assumer le rôle de flotte.
- `deadline:TagResource`- Permet de baliser les travailleurs lors de la création (si activé).
- `deadline:ListTagsForResource`- Permet de lire les étiquettes des flottes pour la propagation.

Comprendre le processus de bootstrap

Le WorkerHost rôle est uniquement utilisé lors du démarrage initial du worker :

1. L'agent de travail démarre sur l'hôte à l'aide des WorkerHost informations d'identification.
2. Il invite `deadline:CreateWorker` à s'inscrire auprès de Deadline Cloud.
3. Il appelle ensuite `deadline:AssumeFleetRoleForWorker` pour récupérer les informations d'identification des rôles de flotte.
4. À partir de ce moment, le travailleur utilise uniquement les informations d'identification relatives au rôle de flotte pour toutes les opérations.

Le WorkerHost rôle n'est pas utilisé une fois que le programme de travail a commencé à fonctionner. Cette politique n'est pas obligatoire pour les Service-managed flottes. Dans les Service-managed flottes, le bootstrap est effectué automatiquement.

Rôle de file d'attente

Le rôle de file d'attente est assumé par le travailleur lors du traitement d'une tâche. Ce rôle fournit les autorisations nécessaires pour effectuer la tâche.

Lorsque vous créez ou mettez à jour des files d'attente par programmation, spécifiez l'ARN du rôle de file d'attente à l'aide des opérations de l'API `CreateQueue` or`UpdateQueue`.

Configurer la politique de confiance des rôles dans la file d'attente

Votre rôle dans la file d'attente doit faire confiance au service Deadline Cloud.

À titre de bonne pratique, la politique de confiance devrait inclure des conditions de sécurité pour la protection des adjoints confus. Pour en savoir plus sur la protection contre Confused Deputy, consultez Confused Deputy dans le guide de l'utilisateur de Deadline Cloud.

- `aws:SourceAccount`garantit que seules ses ressources Compte AWS peuvent assumer ce rôle.
- `aws:SourceArn`limite l'attribution des rôles à une ferme Deadline Cloud spécifique.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

Comprendre les autorisations des rôles de file d'attente

Le rôle de file d'attente n'utilise aucune politique gérée unique. Au lieu de cela, lorsque vous configurez votre file d'attente dans la console, Deadline Cloud crée une politique personnalisée pour votre file d'attente en fonction de votre configuration.

Cette politique créée automatiquement donne accès à :

Pièces jointes aux offres d'emploi

Accès en lecture et en écriture au compartiment Amazon S3 que vous avez spécifié pour les fichiers d'entrée et de sortie des tâches :

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
```

```
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
    }
  }
}
```

Journaux d'emplois

Accès en lecture aux CloudWatch journaux pour les tâches de cette file d'attente. Chaque file d'attente possède son propre groupe de journaux et chaque session possède son propre flux de journaux :

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Third-party logiciel

Accès au téléchargement de logiciels tiers pris en charge par Deadline Cloud (tels que Maya, Blender, etc.) :

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
```

```
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Ajoutez des autorisations pour vos tâches

Ajoutez des autorisations à votre rôle de file d'attente pour Services AWS que vos tâches puissent accéder. Lorsque vous écrivez des scripts d' OpenJobDescription étape, le SDK AWS CLI and utilise automatiquement les informations d'identification de votre rôle dans la file d'attente. Utilisez-le pour accéder aux services supplémentaires nécessaires à l'exécution de votre travail.

Les exemples de cas d’utilisation incluent :

- pour récupérer des données personnalisées
- Autorisations SSM pour un tunnel vers un serveur de licences personnalisé
- CloudWatch pour émettre des métriques personnalisées
- Deadline Cloud autorise la création de nouvelles tâches pour les flux de travail dynamiques

Comment les informations d'identification du rôle de file d'attente sont utilisées

Deadline Cloud fournit les informations d'identification des rôles de file d'attente pour :

- Travailleurs pendant l'exécution des tâches
- Utilisateurs via la CLI et le moniteur de Deadline Cloud lorsqu'ils interagissent avec les pièces jointes aux tâches et les journaux

Deadline Cloud crée des groupes de CloudWatch journaux de journaux distincts pour chaque file d'attente. Les tâches utilisent les informations d'identification du rôle de file d'attente pour écrire des journaux dans le groupe de journaux de leur file d'attente. La CLI et le moniteur de Deadline Cloud utilisent le rôle de file d'attente (via`deadline:AssumeQueueRoleForRead`) pour lire les journaux des tâches à partir du groupe de journaux de la file d'attente. La CLI et le moniteur de Deadline Cloud utilisent le rôle de file d'attente (via`deadline:AssumeQueueRoleForUser`) pour charger ou télécharger les données des pièces jointes aux tâches.

Rôle du moniteur

Configurez un rôle de moniteur pour permettre aux applications Web et de bureau de surveillance de Deadline Cloud d'accéder à vos ressources Deadline Cloud.

Lorsque vous créez ou mettez à jour des moniteurs par programmation, spécifiez l'ARN du rôle de moniteur à l'aide des opérations de l'`UpdateMonitor`API `CreateMonitor` or.

À quoi sert le rôle de moniteur

Le rôle de moniteur permet au moniteur Deadline Cloud de fournir aux utilisateurs finaux l'accès à :

- Fonctionnalités de base requises pour les soumissionnaires intégrés, la CLI et le moniteur de Deadline Cloud
- Fonctionnalités personnalisées pour les utilisateurs finaux

Configurer la politique de confiance relative aux rôles de surveillance

Votre rôle de moniteur doit faire confiance au service Deadline Cloud.

À titre de bonne pratique, la politique de confiance devrait inclure des conditions de sécurité pour la protection des adjoints confus. Pour en savoir plus sur la protection contre Confused Deputy, consultez Confused Deputy dans le guide de l'utilisateur de Deadline Cloud.

`aws:SourceAccount`garantit que seules ses ressources Compte AWS peuvent assumer ce rôle.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
```

```
}
```

Associer des autorisations aux rôles de surveillance

Associez toutes les politiques AWS gérées suivantes à votre rôle de moniteur pour un fonctionnement de base :

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

Fonctionnement du rôle de moniteur

Lorsque vous utilisez le moniteur Deadline Cloud, un utilisateur du service se connecte à l'aide de AWS IAM Identity Center (IAM Identity Center) et le rôle de moniteur est assumé. Les informations d'identification du rôle assumé sont utilisées par l'application de surveillance pour afficher l'interface utilisateur du moniteur, y compris la liste des fermes, des flottes, des files d'attente et d'autres informations.

Lorsque vous utilisez l'application de bureau Deadline Cloud Monitor, ces informations d'identification sont également mises à disposition sur le poste de travail à l'aide d'un profil AWS d'identification nommé correspondant au nom de profil fourni par l'utilisateur final. Pour en savoir plus sur les profils nommés, consultez le guide de référence du AWS SDK et des outils.

Ce profil nommé permet à la CLI de Deadline et aux soumissionnaires d'accéder aux ressources de Deadline Cloud.

Personnalisation du rôle de moniteur pour les cas d'utilisation avancés

Vous pouvez personnaliser le rôle de moniteur pour modifier ce que les utilisateurs peuvent faire à chaque niveau d'accès (lecteur, contributeur, responsable, propriétaire) ou pour ajouter des autorisations pour les flux de travail avancés.

Personnalisation des autorisations de niveau d'accès

Les quatre politiques AWS gérées associées au rôle de moniteur contrôlent ce que chaque niveau d'accès peut faire. Vous pouvez ajouter des politiques personnalisées au rôle de surveillance afin d'accorder ou de restreindre des autorisations pour des niveaux d'accès spécifiques à l'aide de la clé de `deadline:MembershipLevel` condition.

Par exemple, pour permettre aux contributeurs de mettre à jour et d'annuler des tâches (ce qui est normalement réservé aux gestionnaires et aux propriétaires), ajoutez une politique telle que la suivante :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Grâce à cette politique, les contributeurs peuvent mettre à jour et annuler des offres d'emploi en plus de les soumettre.

Ajouter des autorisations pour les flux de travail avancés

Vous pouvez ajouter des politiques IAM personnalisées au rôle de moniteur pour accorder des autorisations supplémentaires à tous les utilisateurs du moniteur. Cela est utile pour les flux de travail de script avancés où les utilisateurs ont besoin d'accéder à des fonctionnalités Services AWS autres que celles standard de Deadline Cloud.

Suivez ces directives lorsque vous modifiez votre rôle de moniteur :

- Ne supprimez aucune des politiques gérées. La suppression de ces politiques interrompt les fonctionnalités du moniteur.

Comment Deadline Cloud Monitor utilise les informations d'identification des rôles de surveillance

Deadline Cloud Monitor obtient automatiquement les informations d'identification du rôle de moniteur lorsque vous vous authentifiez. Cette fonctionnalité permet à l'application de bureau de fournir des fonctionnalités de surveillance améliorées au-delà de ce qui est disponible dans un navigateur Web standard.

Lorsque vous vous connectez avec le moniteur Deadline Cloud, celui-ci crée automatiquement un profil que vous pouvez utiliser avec cet outil AWS CLI ou tout autre AWS outil. Ce profil utilise les informations d'identification du rôle de moniteur, ce qui vous donne un accès programmatique en Services AWS fonction des autorisations associées à votre rôle de moniteur.

Les expéditeurs de Deadline Cloud fonctionnent de la même manière : ils utilisent le profil créé par le moniteur Deadline Cloud pour y accéder Services AWS avec les autorisations de rôle appropriées.

Personnalisation avancée des rôles de Deadline Cloud

Vous pouvez étendre les rôles de Deadline Cloud avec des autorisations supplémentaires afin de permettre des cas d'utilisation avancés allant au-delà des flux de travail de rendu de base. Cette approche s'appuie sur le système de gestion des accès de Deadline Cloud pour contrôler l'accès à d'autres en Services AWS fonction de l'adhésion à la file d'attente.

Collaboration d'équipe avec AWS CodeCommit

Ajoutez AWS CodeCommit des autorisations à votre rôle de file d'attente pour permettre la collaboration en équipe sur les référentiels de projets. Cette approche utilise le système de gestion des accès de Deadline Cloud pour des cas d'utilisation supplémentaires : seuls les utilisateurs ayant accès à la file d'attente spécifique recevront ces AWS CodeCommit autorisations, ce qui vous permet de gérer l'accès au référentiel par projet via l'adhésion à la file d'attente Deadline Cloud.

Cela est utile pour les scénarios dans lesquels les artistes ont besoin d'accéder à des ressources, à des scripts ou à des fichiers de configuration spécifiques au projet stockés dans des AWS CodeCommit référentiels dans le cadre de leur flux de travail de rendu.

Addition AWS CodeCommit autorisations d'accès au rôle de file d'attente

Ajoutez les autorisations suivantes à votre rôle de file d'attente pour permettre AWS CodeCommit l'accès :

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
```

```
}
```

Configurer le fournisseur d'identifiants sur les postes de travail des artistes

Configurez chaque poste de travail artistique pour utiliser les informations d'identification de la file d'attente Deadline Cloud pour AWS CodeCommit y accéder. Cette configuration est effectuée une fois par poste de travail.

Pour configurer le fournisseur d'informations d'identification

1. Ajoutez un profil de fournisseur d'informations d'identification à votre fichier de AWS configuration (`~/.aws/config`) :

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Configurez Git pour utiliser ce profil pour les AWS CodeCommit référentiels :

   ```
   git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
    '!aws codecommit credential-helper --profile queue-codecommit $@'
   git config --global credential.https://git-
   codecommit.REGION.amazonaws.com.UseHttpPath true
   ```

Remplacez *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* et *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* par vos identifiants de ferme et de file d'attente réels. Remplacez *REGION* par votre AWS région (par exemple,`us-west-2`).

Utilisation AWS CodeCommit avec des informations d'identification de file

Une fois configurées, les opérations Git utiliseront automatiquement les informations d'identification du rôle de file d'attente lors de l'accès aux AWS CodeCommit référentiels. La `deadline queue export-credentials` commande renvoie des informations d'identification temporaires qui ressemblent à ceci :

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
```

```
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Ces informations d'identification sont automatiquement actualisées selon les besoins, et les opérations Git fonctionneront parfaitement :

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Les artistes peuvent désormais accéder aux référentiels de projets en utilisant leurs autorisations de file d'attente sans avoir besoin d'informations d' AWS CodeCommit identification distinctes. Seuls les utilisateurs ayant accès à la file d'attente spécifique pourront accéder au référentiel associé, ce qui permettra un contrôle d'accès précis via le système d'adhésion à la file d'attente de Deadline Cloud.

Résolution des problèmes AWS Deadline Identité et accès au cloud

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Deadline Cloud et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Deadline Cloud
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mes ressources Deadline Cloud

Je ne suis pas autorisé à effectuer une action dans Deadline Cloud

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `deadline:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource *`my-example-widget`* à l'aide de l'action `deadline:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Deadline Cloud.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Deadline Cloud. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mes ressources Deadline Cloud

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Deadline Cloud prend en charge ces fonctionnalités, consultezComment Deadline Cloud fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Validation de conformité pour Deadline Cloud

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Résilience dans Deadline Cloud

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui

basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

AWS Deadline Cloud ne sauvegarde pas les données stockées dans le compartiment S3 de vos pièces jointes aux tâches. Vous pouvez activer les sauvegardes des données de vos pièces jointes à des tâches à l'aide de n'importe quel mécanisme de sauvegarde standard d'Amazon S3, tel que le versionnement S3 ou AWS Backup.

Sécurité de l'infrastructure dans Deadline Cloud

En tant que service géré, AWS Deadline Cloud est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Deadline Cloud via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Deadline Cloud ne prend pas en charge l'utilisation de politiques de point de terminaison de cloud privé AWS PrivateLink virtuel (VPC). Il utilise la politique AWS PrivateLink par défaut, qui accorde un accès complet au point de terminaison. Pour plus d'informations, consultez la section Politique de point de terminaison par défaut dans le guide de AWS PrivateLink l'utilisateur.

Analyse de configuration et de vulnérabilité dans Deadline Cloud

AWS gère les tâches de sécurité de base telles que l'application de correctifs au système d'exploitation client (OS) et aux bases de données, la configuration du pare-feu et la reprise après

sinistre. Ces procédures ont été vérifiées et certifiées par les tiers appropriés. Pour plus de détails, consultez les ressources suivantes :

- Modèle de responsabilité partagée
- Amazon Web Services : Présentation des procédures de sécurité (livre blanc)

AWS Deadline Cloud gère les tâches sur les flottes gérées par les services ou par les clients :

- Pour les flottes gérées par des services, Deadline Cloud gère le système d'exploitation client.
- Pour les flottes gérées par le client, vous êtes responsable de la gestion du système d'exploitation.

Pour plus d'informations sur la configuration et l'analyse des vulnérabilités pour AWS Deadline Cloud, voir

- Bonnes pratiques de sécurité pour Deadline Cloud

Cross-service prévention confuse des adjoints

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS, l'usurpation d'identité interservices peut entraîner un problème de confusion chez les adjoints. Cross-service l'usurpation d'identité peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé à accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services avec des principaux de service qui ont eu accès aux ressources de votre compte.

Nous recommandons d'utiliser les clés de contexte de condition aws:SourceAccountglobale aws:SourceArnet les clés contextuelles dans les politiques de ressources afin de limiter les autorisations qui AWS Deadline Cloud accordent un autre service à la ressource. Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

Le moyen le plus efficace de se protéger contre le problème de l'adjoint confus est d'utiliser la clé de `aws:SourceArn` contexte de condition globale avec le nom de ressource Amazon (ARN) complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource

ou si vous spécifiez plusieurs ressources, utilisez la clé de contexte de condition globale `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:deadline:*:123456789012:*`.

Si la valeur `aws:SourceArn` ne contient pas l'ID du compte, tel qu'un ARN de compartiment Amazon S3, vous devez utiliser les deux clés de contexte de condition globale pour limiter les autorisations.

L'exemple suivant montre comment vous pouvez utiliser les touches de contexte de condition `aws:SourceAccount` globale `aws:SourceArn` et globale Deadline Cloud pour éviter le problème de confusion des adjoints.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

Accès AWS Deadline Cloud en utilisant un point de terminaison d'interface (AWS PrivateLink)

Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et. AWS Deadline Cloud Vous pouvez y accéder Deadline Cloud comme s'il se trouvait dans votre VPC, sans utiliser de passerelle Internet, de périphérique NAT, de connexion VPN ou Direct Connect de connexion. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour y accéder. Deadline Cloud

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par le demandeur qui servent de point d'entrée pour le trafic destiné à Deadline Cloud.

Deadline Cloud propose également des points de terminaison à double pile. Dual-stack les points de terminaison prennent en charge les demandes via IPv6 et IPv4.

Pour plus d'informations, consultez Accès aux Services AWS via AWS PrivateLink dans le Guide AWS PrivateLink .

Considérations relatives à Deadline Cloud

Avant de configurer un point de terminaison d'interface pour Deadline Cloud, consultez la section Accès à un service AWS à l'aide d'un point de terminaison VPC d'interface dans le AWS PrivateLink Guide.

Deadline Cloud prend en charge les appels à toutes ses actions d'API via le point de terminaison de l'interface.

Par défaut, l'accès complet à Deadline Cloud est autorisé via le point de terminaison de l'interface. Vous pouvez également associer un groupe de sécurité aux interfaces réseau du point de terminaison pour contrôler le trafic Deadline Cloud passant par le point de terminaison de l'interface.

Deadline Cloud prend également en charge les politiques de point de terminaison VPC. Pour plus d'informations, consultez Contrôle de l'accès aux points de terminaison d'un VPC à l'aide de politiques de point de terminaison dans le AWS PrivateLink .

Deadline Cloud points de terminaison

Deadline Cloud utilise quatre points de terminaison pour accéder au service en utilisant AWS PrivateLink : deux pour IPv4 et deux pour IPv6.

Les travailleurs utilisent le `scheduling.deadline.region.amazonaws.com` point de terminaison pour récupérer les tâches de la file d'attente Deadline Cloud, rendre compte de la progression et renvoyer les résultats des tâches. Si vous utilisez une flotte gérée par le client, le point de terminaison de planification est le seul point de terminaison que vous devez créer, sauf si vous utilisez des opérations de gestion. Par exemple, si une tâche crée d'autres tâches, vous devez autoriser le point de terminaison de gestion à appeler l'`CreateJob`opération.

Le Deadline Cloud moniteur utilise le `management.deadline.region.amazonaws.com` pour gérer les ressources de votre ferme, par exemple en créant et en modifiant des files d'attente et des flottes ou en obtenant des listes de tâches, d'étapes et de tâches.

Les AWS SDK et la CLI ajoutent automatiquement les `scheduling` préfixes `management` et au point de terminaison. Si vous souhaitez désactiver ce comportement, consultez la section relative à l'injection du préfixe d'hôte dans le Guide de référence AWS des SDK et des outils.

Deadline Cloud nécessite également des points de terminaison pour les points de terminaison AWS de service suivants :

- Si vous configurez votre flotte gérée par le client dans un sous-réseau sans connexion Internet, vous devez créer un point de terminaison VPC pour CloudWatch Amazon Logs afin que les employés puissent écrire des journaux. Pour plus d'informations, consultez la section Surveillance avec CloudWatch.
- Si vous utilisez des pièces jointes à des tâches, vous devez créer un point de terminaison VPC pour Amazon Simple Storage Service (Amazon S3) afin que les employés puissent accéder aux pièces jointes. Pour plus d'informations, consultez la section Pièces jointes aux Job dans Deadline Cloud.

Créez des points de terminaison pour Deadline Cloud

Vous pouvez créer des points de terminaison d'interface pour Deadline Cloud utiliser la console Amazon VPC ou AWS Command Line Interface le AWS CLI(). Pour plus d’informations, consultez Création d’un point de terminaison d’interface dans le Guide AWS PrivateLink .

Créez des points de terminaison de gestion et de planification pour Deadline Cloud utiliser les noms de service suivants. *region*Remplacez-le par celui Région AWS où vous avez été déployé Deadline Cloud.

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud prend en charge les points de terminaison à double pile.

Si vous activez le DNS privé pour les points de terminaison de l'interface, vous pouvez envoyer des demandes d'API à Deadline Cloud l'aide de son nom DNS régional par défaut. Par exemple, `scheduling.deadline.us-east-1.amazonaws.com` pour les opérations des travailleurs ou `management.deadline.us-east-1.amazonaws.com` pour toutes les autres opérations.

Si votre flotte gérée par le client se trouve sur un sous-réseau sans connexion Internet, vous devez créer un point de terminaison CloudWatch Logs en utilisant le nom de service suivant :

```
com.amazonaws.region.logs
```

Si vous utilisez des pièces jointes pour transférer des fichiers, vous devez créer un point de terminaison Amazon S3 en utilisant le nom de service suivant :

```
com.amazonaws.region.s3
```

Environnements réseau restreints

Deadline Cloud fournit des outils utilisés par les artistes ou d'autres utilisateurs sur leurs postes de travail locaux. Ces outils nécessitent un accès à l' AWS API et aux points de terminaison Web pour exécuter leur fonction. Si vous filtrez l'accès à des AWS domaines ou points de terminaison d'URL spécifiques à l'aide d'une solution de filtrage de contenu Web telle que les pare-feux de nouvelle génération (NGFW) ou les passerelles Web sécurisées (SWG), vous devez ajouter les domaines ou points de terminaison d'URL suivants aux listes d'autorisation de votre solution de filtrage de contenu Web.

AWS Points de terminaison d'API à autoriser

Les outils clients de Deadline Cloud, tels que le moniteur AWS Management Console, la CLI et les soumetteurs intégrés, nécessitent un accès aux AWS API en plus de Deadline Cloud. Ces points de terminaison ne prennent en charge que l'IPv4.

- `scheduling.deadline.[Region].amazonaws.com`
- `management.deadline.[Region].amazonaws.com`
- `logs.[Region].amazonaws.com`
- `ec2.[Region].amazonaws.com`
- `s3.[Region].amazonaws.com`
- `sts.[Region].amazonaws.com`
- `identitystore.[Region].amazonaws.com`

Liste des domaines Web à autoriser

Le moniteur Deadline Cloud nécessite l'accès aux domaines suivants pour fonctionner.

Pour plus d'informations sur l'autorisation de répertorier des domaines pour AWS Sign-In, consultez la section Domaines à ajouter à votre liste autorisée dans le Guide de l'AWS Sign-In utilisateur.

- `downloads.deadlinecloud.amazonaws.com`
- `d2ev1rdnjzhmnr.cloudfront.net`
- `prod.log.shortbread.aws.dev`
- `prod.tools.shortbread.aws.dev`
- `prod.log.shortbread.analytics.console.aws.a2z.com`
- `prod.tools.shortbread.analytics.console.aws.a2z.com`
- `global.help-panel.docs.aws.a2z.com`
- `[Region].signin.aws`
- `[Region].signin.aws.amazon.com`
- `sso.[Region].amazonaws.com`
- `portal.sso.[Region].amazonaws.com`
- `oidc.[Region].amazonaws.com`

- `assets.sso-portal.[Region].amazonaws.com`

Environment-specific points de terminaison à autoriser

Ces domaines varient en fonction de la configuration spécifique de Deadline Cloud. Si des moniteurs ou des files d'attente supplémentaires de Deadline Cloud sont créés, des domaines supplémentaires devront être autorisés.

- *[Directory ID or alias]*`.awsapps.com`

  Ce domaine est lié à la configuration du centre d'identité IAM et doit être le même pour toutes les configurations utilisant la même instance de centre d'identité IAM. La valeur exacte peut être trouvée par l'administrateur de l'entreprise dans la console IAM Identity Center sous Paramètres → Portail d'accès AWS URL.
- *[Monitor alias]*`.`*[Region]*`.deadlinecloud.amazonaws.com`

  Ce domaine est destiné à la configuration du moniteur dans Deadline Cloud. Les artistes saisissent ce lien dans leur navigateur ou dans l'application de surveillance Deadline Cloud. Si Deadline Cloud est configuré dans d'autres comptes ou régions à l'avenir, ce domaine changera. Vous pouvez trouver cette valeur dans la console Deadline Cloud dans le tableau de bord → Vue d'ensemble du moniteur → Détails du moniteur → URL.
- *[Bucket name]*`.`*[Region]*`.s3.amazonaws.com`

  Il s'agit du domaine du bucket de pièces jointes aux tâches utilisé par les files d'attente de Deadline Cloud. Chaque file d'attente peut avoir son propre compartiment de pièces jointes aux tâches configuré. Le nom exact du bucket se trouve dans la console Deadline Cloud sous Queues → Détails de la file d'attente → Pièces jointes aux Job. Pour plus d'informations sur les pièces jointes aux tâches, consultez la documentation sur les files d'attente.

Bonnes pratiques de sécurité pour Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

> **Note**
>
> Pour plus d'informations sur l'importance de nombreux sujets liés à la sécurité, consultez le modèle de responsabilité partagée.

Protection des données

Pour des raisons de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des comptes individuels avec Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui vous aident à découvrir et à sécuriser les données personnelles stockées dans Amazon Simple Storage Service (Amazon S3).
- Si vous avez besoin de modules cryptographiques validés FIPS 140-2 lorsque vous accédez à AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour de plus amples informations sur les points de terminaison FIPS disponibles, consultez Federal Information Processing Standard (FIPS) 140-2.

Nous vous recommandons vivement de ne jamais placer d'informations identifiables sensibles, telles que les numéros de compte de vos clients, dans des champs de formulaire comme Name (Nom). Cette recommandation s'applique également lorsque vous travaillez avec AWS Deadline Cloud ou une autre solution Services AWS à l'aide de la console AWS CLI, de l'API ou AWS des SDK. Toutes les données que vous saisissez dans Deadline Cloud ou dans d'autres services peuvent être récupérées pour être incluses dans les journaux de diagnostic. Lorsque vous fournissez une URL à un serveur externe, n'incluez pas les informations d'identification non chiffrées dans l'URL pour valider votre demande adressée au serveur.

Gestion des identités et des accès AWS des autorisations

Gérez l'accès aux AWS ressources à l'aide des utilisateurs, des rôles Gestion des identités et des accès AWS (IAM) et en accordant le moindre privilège aux utilisateurs. Établissez des politiques et des procédures de gestion des informations d'identification pour la création, la distribution, la rotation et la révocation des informations AWS d'accès. Pour plus d'informations, consultez Bonnes pratiques IAM dans le Guide de l'utilisateur IAM.

Exécuter des tâches en tant qu'utilisateurs et en tant que groupes

Lorsque vous utilisez la fonctionnalité de file d'attente dans Deadline Cloud, il est recommandé de spécifier un utilisateur du système d'exploitation (OS) et son groupe principal afin que l'utilisateur du système d'exploitation dispose des autorisations les moins privilégiées pour les tâches de la file d'attente.

Lorsque vous spécifiez un « Exécuter en tant qu'utilisateur » (et un groupe), tous les processus relatifs aux tâches soumises à la file d'attente seront exécutés à l'aide de cet utilisateur du système d'exploitation et hériteront des autorisations de système d'exploitation associées à cet utilisateur.

Les configurations de flotte et de file d'attente se combinent pour établir une posture de sécurité. Du côté de la file d'attente, le rôle « Job exécuté en tant qu'utilisateur » et le rôle IAM peuvent être spécifiés pour utiliser le système d'exploitation et AWS les autorisations pour les tâches de la file d'attente. Le parc définit l'infrastructure (hôtes de travail, réseaux, stockage partagé monté) qui, lorsqu'elle est associée à une file d'attente particulière, exécute les tâches au sein de cette file. Les données disponibles sur les hôtes de travail doivent être accessibles par les jobs depuis une ou plusieurs files d'attente associées. La spécification d'un utilisateur ou d'un groupe permet de protéger les données des tâches contre les autres files d'attente, les autres logiciels installés ou les autres utilisateurs ayant accès aux hôtes de travail. Lorsqu'une file d'attente n'a pas d'utilisateur, elle s'exécute en tant qu'utilisateur agent qui peut se faire passer pour (`sudo`) n'importe quel utilisateur de la file d'attente. Ainsi, une file d'attente sans utilisateur peut transférer des privilèges à une autre file d'attente.

Réseaux

Pour éviter que le trafic ne soit intercepté ou redirigé, il est essentiel de sécuriser comment et où le trafic de votre réseau est acheminé.

Nous vous recommandons de sécuriser votre environnement réseau de la manière suivante :

- Sécurisez les tables de routage du sous-réseau Amazon Virtual Private Cloud (Amazon VPC) pour contrôler le mode de routage du trafic de la couche IP.
- Si vous utilisez Amazon Route 53 (Route 53) comme fournisseur DNS dans la configuration de votre parc ou de votre station de travail, sécurisez l'accès à l'API Route 53.
- Si vous vous connectez à Deadline Cloud en dehors de celui-ci, par AWS exemple en utilisant des postes de travail sur site ou d'autres centres de données, sécurisez toute infrastructure réseau sur site. Cela inclut les serveurs DNS et les tables de routage sur les routeurs, les commutateurs et autres périphériques réseau.

Emplois et données sur les emplois

Les tâches Deadline Cloud s'exécutent dans le cadre de sessions sur des hôtes de travail. Chaque session exécute un ou plusieurs processus sur l'hôte de travail, qui nécessitent généralement que vous saisissiez des données pour produire une sortie.

Pour sécuriser ces données, vous pouvez configurer les utilisateurs du système d'exploitation avec des files d'attente. L'agent de travail utilise l'utilisateur du système d'exploitation de la file d'attente pour exécuter les sous-processus de session. Ces sous-processus héritent des autorisations de l'utilisateur du système d'exploitation de la file d'attente.

Nous vous recommandons de suivre les meilleures pratiques pour sécuriser l'accès aux données auxquelles ces sous-processus accèdent. Pour de plus amples informations, veuillez consulter Modèle de responsabilité partagée.

Structure de la ferme

Vous pouvez organiser les flottes et les files d'attente de Deadline Cloud de nombreuses manières. Cependant, certains arrangements ont des implications en matière de sécurité.

Une ferme possède l'une des limites les plus sécurisées, car elle ne peut pas partager les ressources de Deadline Cloud avec d'autres fermes, notamment les flottes, les files d'attente et les profils de stockage. Cependant, vous pouvez partager AWS des ressources externes au sein d'un parc de serveurs, ce qui compromet les limites de sécurité.

Vous pouvez également établir des limites de sécurité entre les files d'attente d'une même batterie de serveurs en utilisant la configuration appropriée.

Suivez ces bonnes pratiques pour créer des files d'attente sécurisées dans le même parc de serveurs :

- Associez une flotte uniquement aux files d'attente situées dans la même limite de sécurité. Notez ce qui suit :
  - Une fois la tâche exécutée sur l'hôte de travail, les données peuvent rester, par exemple dans un répertoire temporaire ou dans le répertoire personnel de l'utilisateur de la file d'attente.
  - Le même utilisateur du système d'exploitation exécute toutes les tâches sur un hôte de flotte appartenant au service, quelle que soit la file d'attente à laquelle vous soumettez la tâche.
  - Une tâche peut laisser des processus s'exécuter sur un hôte de travail, ce qui permet aux tâches d'autres files d'attente d'observer d'autres processus en cours d'exécution.
- Assurez-vous que seules les files d'attente situées dans la même limite de sécurité partagent un compartiment Amazon S3 pour les pièces jointes aux tâches.
- Assurez-vous que seules les files d'attente situées dans les mêmes limites de sécurité partagent un même utilisateur du système d'exploitation.
- Sécurisez toutes les autres AWS ressources intégrées à la ferme jusqu'à la limite.

Files d'attente pour les offres d'emploi

Les pièces jointes aux tâches sont associées à une file d'attente qui utilise votre compartiment Amazon S3.

- Les pièces jointes aux tâches sont écrites et lues à partir d'un préfixe racine du compartiment Amazon S3. Vous spécifiez ce préfixe racine dans l'appel `CreateQueue` d'API.
- Le bucket a un correspondant`Queue Role`, qui spécifie le rôle qui accorde aux utilisateurs de la file d'attente l'accès au bucket et au préfixe racine. Lorsque vous créez une file d'attente, vous spécifiez l'`Queue Role`Amazon Resource Name (ARN) à côté du compartiment des pièces jointes aux tâches et du préfixe racine.
- Les appels autorisés aux opérations `AssumeQueueRoleForRead`AssumeQueueRoleForUser, et `AssumeQueueRoleForWorker` API renvoient un ensemble d'informations d'identification de sécurité temporaires pour le`Queue Role`.

Si vous créez une file d'attente et que vous réutilisez un compartiment Amazon S3 et un préfixe racine, des informations risquent d'être divulguées à des tiers non autorisés. Par exemple, QueueA et QueueB partagent le même bucket et le même préfixe racine. Dans un flux de travail sécurisé, ArtistA a accès à QueueA mais pas à QueueB. Toutefois, lorsque plusieurs files d'attente partagent un bucket, ArtistA peut accéder aux données contenues dans QueueB car il utilise le même bucket et le même préfixe racine que QueueA.

La console configure des files d'attente sécurisées par défaut. Assurez-vous que les files d'attente comportent une combinaison distincte de compartiment Amazon S3 et de préfixe racine, sauf si elles font partie d'une limite de sécurité commune.

Pour isoler vos files d'attente, vous devez configurer le `Queue Role` pour n'autoriser l'accès aux files d'attente qu'au bucket et au préfixe racine. Dans l'exemple suivant, remplacez chacune par les *`placeholder`* informations spécifiques à votre ressource.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

Vous devez également définir une politique de confiance pour le rôle. Dans l'exemple suivant, remplacez le *placeholder* texte par les informations spécifiques à votre ressource.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
```

```
        }
    ]
}
```

Buckets Amazon S3 logiciels personnalisés

Vous pouvez ajouter l'instruction suivante à votre compte `Queue Role` pour accéder aux logiciels personnalisés de votre compartiment Amazon S3. Dans l'exemple suivant, remplacez-le *SOFTWARE_BUCKET_NAME* par le nom de votre compartiment S3 et *BUCKET_ACCOUNT_OWNER* par l' Compte AWS ID du propriétaire du compartiment.

```
"Statement": [
    {
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

Pour plus d'informations sur les meilleures pratiques de sécurité d'Amazon S3, consultez la section Meilleures pratiques de sécurité pour Amazon S3 dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Hôtes de travail

Sécurisez les hôtes de travail pour garantir que chaque utilisateur ne peut effectuer des opérations que pour le rôle qui lui est assigné.

Nous recommandons les meilleures pratiques suivantes pour sécuriser les hôtes de travail :

- L'utilisation d'un script de configuration d'hôte peut modifier la sécurité et les opérations d'un travailleur. Une configuration incorrecte peut rendre le travailleur instable ou l'arrêter de travailler. Il est de votre responsabilité de corriger ces défaillances.
- N'utilisez pas la même `jobRunAsUser` valeur avec plusieurs files d'attente, sauf si les tâches soumises à ces files d'attente se situent dans la même limite de sécurité.
- Ne définissez pas la file `jobRunAsUser` d'attente sur le nom de l'utilisateur du système d'exploitation sous lequel l'agent de travail s'exécute.
- Accordez aux utilisateurs de la file d'attente les autorisations de système d'exploitation les moins privilégiées requises pour les charges de travail de file d'attente prévues. Assurez-vous qu'ils ne disposent pas d'autorisations d'écriture dans le système de fichiers pour accéder aux fichiers du programme de l'agent de travail ou à d'autres logiciels partagés.
- Assurez-vous que seuls l'utilisateur root Linux et le compte `Administrator` propriétaire sont propriétaires et peuvent modifier les fichiers du programme de l'agent de travail. Windows
- Sur les hôtes de Linux travail, envisagez de configurer une `umask` dérogation permettant à `/etc/sudoers` l'utilisateur de l'agent de travail de lancer des processus en tant qu'utilisateurs de la file d'attente. Cette configuration permet de garantir que les autres utilisateurs ne peuvent pas accéder aux fichiers écrits dans la file d'attente.
- Accordez à des personnes de confiance un accès moins privilégié aux hôtes professionnels.
- Limitez les autorisations aux fichiers de configuration de remplacement du DNS local (activé `/etc/hosts` et activéWindows) Linux et au routage des tables `C:\Windows\system32\etc\hosts` sur les postes de travail et les systèmes d'exploitation des hôtes de travail.
- Limitez les autorisations relatives à la configuration DNS sur les postes de travail et les systèmes d'exploitation hôtes des travailleurs.
- Appliquez régulièrement des correctifs au système d'exploitation et à tous les logiciels installés. Cette approche inclut les logiciels spécifiquement utilisés avec Deadline Cloud, tels que les émetteurs, les adaptateurs, les agents de travail, les OpenJD packages, etc.
- Utilisez des mots de passe forts pour la Windows file d'attente`jobRunAsUser`.
- Changez régulièrement les mots de passe de votre file d'attente`jobRunAsUser`.
- Garantissez l'accès aux secrets du mot de Windows passe avec le moindre privilège et supprimez les secrets non utilisés.
- N'`jobRunAsUser`autorisez pas la file d'attente à exécuter les commandes de planification à l'avenir :
  - ActivéLinux, refusez à ces comptes l'accès à `cron` et`at`.

- ActivéWindows, refusez à ces comptes l'accès au Windows planificateur de tâches.

> **Note**
>
> Pour plus d'informations sur l'importance d'appliquer régulièrement des correctifs au système d'exploitation et aux logiciels installés, consultez le modèle de responsabilité partagée.

Script de configuration de l'hôte

- L'utilisation d'un script de configuration d'hôte peut modifier la sécurité et les opérations d'un travailleur. Une configuration incorrecte peut rendre le travailleur instable ou l'arrêter de travailler. Il est de votre responsabilité de corriger ces défaillances.

Stations de travail

Il est important de sécuriser les postes de travail ayant accès à Deadline Cloud. Cette approche permet de garantir que les tâches que vous soumettez à Deadline Cloud ne peuvent pas exécuter des charges de travail arbitraires facturées à votre compte. Compte AWS

Nous recommandons de suivre les bonnes pratiques suivantes pour sécuriser les postes de travail des artistes. Pour plus d'informations, consultez le Modèle de responsabilité partagée .

- Sécurisez toutes les informations d'identification persistantes donnant accès à Deadline Cloud AWS, y compris. Pour plus d’informations, consultez Gestion des clés d’accès pour les utilisateurs IAM dans le Guide de l’utilisateur IAM.
- Installez uniquement des logiciels fiables et sécurisés.
- Exigez que les utilisateurs se fédérent avec un fournisseur d'identité pour accéder à l' AWS aide d'informations d'identification temporaires.
- Utilisez des autorisations sécurisées sur les fichiers du programme d'envoi de Deadline Cloud pour éviter toute falsification.
- Accordez à des personnes de confiance un accès moins privilégié aux postes de travail des artistes.
- N'utilisez que des émetteurs et des adaptateurs que vous obtenez via le Deadline Cloud Monitor.

- Limitez les autorisations aux fichiers de configuration de remplacement du DNS local (`/etc/hosts`activé Linux et macOS activéWindows) et au routage des tables `C:\Windows\system32\etc\hosts` sur les postes de travail et les systèmes d'exploitation des hôtes de travail.
- Limitez les autorisations `/etc/resolve.conf` aux postes de travail et aux systèmes d'exploitation hôtes des travailleurs.
- Appliquez régulièrement des correctifs au système d'exploitation et à tous les logiciels installés. Cette approche inclut les logiciels spécifiquement utilisés avec Deadline Cloud, tels que les émetteurs, les adaptateurs, les agents de travail, les OpenJD packages, etc.

Vérifier l'authenticité du logiciel téléchargé

Vérifiez l'authenticité de votre logiciel après avoir téléchargé le programme d'installation afin de vous protéger contre la falsification de fichiers. Cette procédure fonctionne pour Windows les deux Linux systèmes.

Windows

Pour vérifier l'authenticité des fichiers que vous avez téléchargés, procédez comme suit.

1. Dans la commande suivante, *`file`* remplacez-le par le fichier que vous souhaitez vérifier. Par exemple, ***`C:\PATH\TO\MY\`*`DeadlineCloudSubmitter-windows-x64-installer.exe`** . Remplacez-le également *`signtool-sdk-version`* par la version du SignTool SDK installée. Par exemple, **`10.0.22000.0`**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Par exemple, vous pouvez vérifier le fichier d'installation de l'expéditeur de Deadline Cloud en exécutant la commande suivante :

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-
   windows-x64-installer.exe
   ```

Linux

Pour vérifier l'authenticité des fichiers téléchargés, utilisez l'outil de ligne de gpg commande.

1. Importez la OpenPGP clé en exécutant la commande suivante :

```
 gpg --import --armor <<EOF
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
/shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. Déterminez s'il faut faire confiance à la OpenPGP clé. Certains facteurs à prendre en compte pour décider de faire confiance à la clé ci-dessus sont les suivants :

   - La connexion Internet que vous avez utilisée pour obtenir la clé GPG sur ce site Web est sécurisée.
   - L'appareil sur lequel vous accédez à ce site Web est sécurisé.
   - AWS a pris des mesures pour sécuriser l'hébergement de la clé OpenPGP publique sur ce site Web.

3. Si vous décidez de faire confiance à la OpenPGP clé, modifiez-la gpg comme dans l'exemple suivant :

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF

    gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.


    pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud example@example.com

    gpg> trust
    pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com

    Please decide how far you trust this user to correctly verify other users'
 keys
    (by looking at passports, checking fingerprints from different sources,
 etc.)

      1 = I don't know or won't say
      2 = I do NOT trust
      3 = I trust marginally
      4 = I trust fully
      5 = I trust ultimately
      m = back to the main menu

    Your decision? 5
    Do you really want to set this key to ultimate trust? (y/N) y

    pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                         trust: ultimate      validity: unknown
    [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
    Please note that the shown key validity is not necessarily correct
    unless you restart the program.

    gpg> quit
```

4. Vérifiez le programme d'installation de Deadline Cloud Submitter

   Pour vérifier le programme d'installation de Deadline Cloud Submitter, procédez comme suit :

   a. Téléchargez le fichier de signature pour le programme d'installation de Deadline Cloud Submitter.

      Télécharger le fichier de signature (.sig)

   b. Vérifiez la signature du programme d'installation de l'émetteur de Deadline Cloud en exécutant :

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. Vérifiez le moniteur Deadline Cloud

   > **Note**
   >
   > Vous pouvez vérifier le téléchargement du moniteur Deadline Cloud à l'aide de fichiers de signature ou de méthodes spécifiques à la plate-forme. Pour les méthodes spécifiques à la plate-forme, consultez l'Linux (Debian)onglet, l'onglet Linux (RPM) ou l'Linux (AppImage)onglet en fonction du type de fichier que vous avez téléchargé.

   Pour vérifier l'application de bureau Deadline Cloud Monitor avec les fichiers de signature, procédez comme suit :

   a. Téléchargez le fichier de signature correspondant pour votre programme d'installation du moniteur Deadline Cloud :

      - Télécharger le fichier de signature .deb
      - Télécharger le fichier de signature .rpm
      - Télécharger. AppImage fichier de signature

   b. Vérifiez la signature :

      Pour le .deb :

      ```
      gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
      monitor_amd64.deb
      ```

Pour .rpm :

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

Pour. AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Vérifiez que le résultat ressemble à ce qui suit :

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Si la sortie contient cette phrase`Good signature from "AWS Deadline Cloud"`, cela signifie que la signature a été vérifiée avec succès et que vous pouvez exécuter le script d'installation du moniteur Deadline Cloud.

Clés historiques

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
```

```
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

Pour vérifier les packages qui utilisent unLinux. AppImage binaire, effectuez d'abord les étapes 1 à 3 dans l'Linuxonglet, puis effectuez les étapes suivantes.

1. À partir de la AppImageUpdate page qui apparaît GitHub, téléchargez le validate-x86_64. AppImagefichier.
2. Après avoir téléchargé le fichier, pour ajouter des autorisations d'exécution, exécutez la commande suivante.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```

3. Pour ajouter des autorisations d'exécution, exécutez la commande suivante.

   ```
   chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

4. Pour vérifier la signature du moniteur Deadline Cloud, exécutez la commande suivante.

   ```
   ./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

   Si la sortie contient cette phrase`Validation successful`, cela signifie que la signature a été vérifiée avec succès et que vous pouvez exécuter le script d'installation du moniteur Deadline Cloud en toute sécurité.

Linux (Debian)

Pour vérifier les packages qui utilisent un binaire Linux .deb, effectuez d'abord les étapes 1 à 3 de l'Linuxonglet.

dpkg est le principal outil de gestion des paquets dans la plupart des Linux distributions debian basées. Vous pouvez vérifier le fichier .deb avec l'outil.

1. Téléchargez le fichier .deb du moniteur Deadline Cloud :

   Télécharger le moniteur Deadline Cloud (.deb)
2. Vérifiez le fichier .deb :

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. Le résultat sera similaire à :

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. Pour vérifier le fichier .deb, vérifiez qu'il `GOODSIG` est présent dans la sortie.

Linux (RPM)

Pour vérifier les packages qui utilisent un binaire Linux .rpm, effectuez d'abord les étapes 1 à 3 de l'Linuxonglet.

1. Téléchargez le fichier .rpm du moniteur Deadline Cloud :

   Télécharger le moniteur Deadline Cloud (.rpm)
2. Vérifiez le fichier .rpm :

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. Le résultat sera similaire à :

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. Pour vérifier le fichier .rpm, vérifiez qu'il `digests signatures OK` figure dans le résultat.

Historique du document

Pour plus d'informations sur les mises à jour de AWS Deadline Cloud, consultez les notes de mise à jour de Deadline Cloud.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.