Guide de l’utilisateur

AWS Proton

AWS Proton: Guide de l’utilisateur

Table of Contents

Avis de fin de support : le 7 octobre 2026, AWS le support de AWS Proton. Après le 7 octobre 2026, vous ne pourrez plus accéder à la AWS Proton console ni aux AWS Proton ressources. Votre infrastructure déployée restera intacte. Pour plus d'informations, consultez le Guide AWS Proton de dépréciation et de migration des services.

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.

Qu'est-ce que c'est AWS Proton ?

AWS Proton est :

- Infrastructure automatisée sous forme de provisionnement de code et de déploiement d'applications sans serveur et basées sur des conteneurs

  Le AWS Proton service est un cadre d'automatisation à deux volets. En tant qu'administrateur, vous créez des modèles de service versionnés qui définissent une infrastructure normalisée et des outils de déploiement pour les applications sans serveur et basées sur des conteneurs. En tant que développeur d'applications, vous pouvez sélectionner parmi les modèles de service disponibles pour automatiser vos déploiements d'applications ou de services.

  AWS Proton identifie pour vous toutes les instances de service existantes qui utilisent une version de modèle obsolète. En tant qu'administrateur, vous pouvez demander AWS Proton à les mettre à niveau en un clic.
- Infrastructure standardisée

  Les équipes de la plateforme peuvent utiliser AWS Proton une infrastructure versionnée comme modèles de code. Ils peuvent utiliser ces modèles pour définir et gérer des piles d'applications standard contenant l'architecture, les ressources d'infrastructure et le pipeline de déploiement des CI/CD logiciels.
- Déploiements intégrés à CI/CD

  Lorsque les développeurs utilisent l'interface en AWS Proton libre-service pour sélectionner un modèle de service, ils sélectionnent une définition de pile d'applications standardisée pour leurs déploiements de code. AWS Proton provisionne automatiquement les ressources, configure le CI/CD pipeline et déploie le code dans l'infrastructure définie.

AWS Proton pour les équipes de la plateforme

En tant qu'administrateur, vous ou les membres de votre équipe de plateforme créez des modèles d'environnement et de service contenant l'infrastructure sous forme de code. Le modèle d'environnement définit l'infrastructure partagée utilisée par plusieurs applications ou ressources. Le modèle de service définit le type d'infrastructure nécessaire pour déployer et gérer une seule application ou un microservice dans un environnement. Un AWS Proton service est une instanciation d'un modèle de service, qui inclut normalement plusieurs instances de service et un pipeline.

Une instance AWS Proton de service est une instanciation d'un modèle de service dans un environnement spécifique. Vous ou les autres membres de votre équipe pouvez spécifier les modèles d'environnement compatibles avec un modèle de service donné. Pour plus d'informations sur les modèles, consultezAWS Proton modèles.

Vous pouvez utiliser l'infrastructure suivante en tant que fournisseur de code avec AWS Proton :

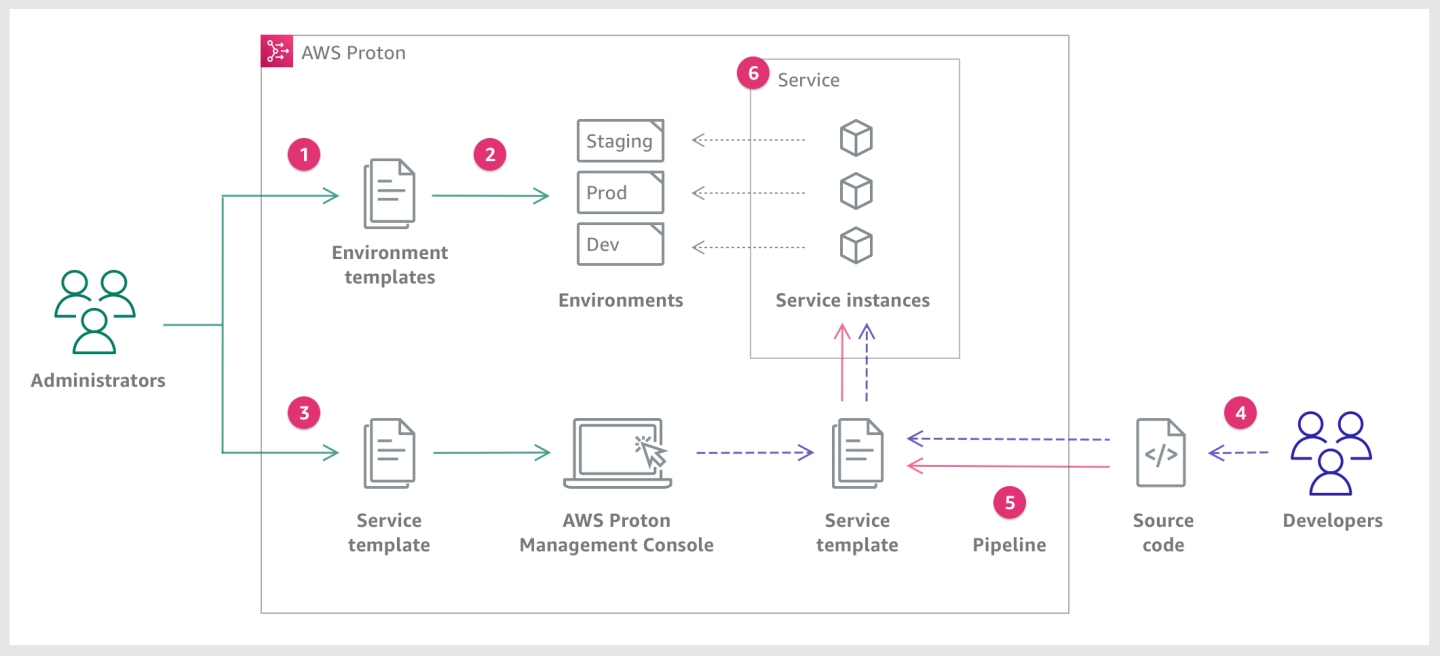
- CloudFormation
- Terraforme

AWS Proton pour les développeurs

En tant que développeur d'applications, vous sélectionnez un modèle de service standardisé qui AWS Proton permet de créer un service qui déploie et gère votre application dans une instance de service. Un AWS Proton service est une instanciation d'un modèle de service, qui inclut normalement plusieurs instances de service et un pipeline.

AWS Proton flux de travail

Le schéma suivant est une visualisation des principaux AWS Proton concepts abordés dans le paragraphe précédent. Il offre également une vue d'ensemble de haut niveau de ce qui constitue un AWS Proton flux de travail simple.



1. En tant qu'administrateur, vous créez et enregistrez un modèle d'environnement avec AWS Proton lequel vous définissez les ressources partagées.
2. AWS Proton déploie un ou plusieurs environnements, sur la base d'un modèle d'environnement.
3. En tant qu'administrateur, vous créez et enregistrez un modèle de service auprès AWS Proton duquel vous définissez l'infrastructure, la surveillance et les CI/CD ressources associées, ainsi que les modèles d'environnement compatibles.
4. En tant que développeur, vous sélectionnez un modèle de service enregistré et vous fournissez un lien vers votre référentiel de code source.
5. AWS Proton fournit au Service un pipeline CI/CD pour vos instances de service.
6. AWS Proton fournit et gère le service et les instances de service qui exécutent le code source tel que défini dans le modèle de service sélectionné. Une instance de service est une instanciation du modèle de service sélectionné dans un environnement pour une seule étape d'un pipeline (par exemple Prod).

AWS Proton Guide de dépréciation et de migration des services

AWS a décidé de mettre fin au support AWS Proton, le support prenant fin le 7 octobre 2026. Les nouveaux clients ne pourront pas s'inscrire après le 7 octobre 2025, mais les clients existants pourront continuer à utiliser le service jusqu'au 7 octobre 2026.

État du service jusqu'à la dépréciation

Jusqu'au 7 octobre 2026, les AWS Proton clients existants peuvent continuer à utiliser le service normalement. Au cours de cette période, AWS vous allez :

1. Fournir des correctifs de sécurité et des corrections de bogues critiques

2. Maintenir la disponibilité et les performances des services
3. Continuer à offrir une assistance par le biais de AWS Support canaux
4. Ne pas ajouter de nouvelles fonctionnalités au service

Informations importantes sur la migration

AWS Proton est avant tout un CI/CD outil de déploiement d'infrastructures. Lorsqu'elle AWS Proton est déconseillée, vos CloudFormation piles déployées et les ressources qu'elles gèrent resteront intactes et continueront de fonctionner. La dépréciation n'affecte que les pipelines de livraison et le AWS Proton service lui-même, et non votre infrastructure déployée.

Solutions alternatives

Nous avons identifié plusieurs alternatives AWS Proton qui peuvent vous aider à maintenir votre infrastructure sous forme de code et les fonctionnalités CI/CD.

CloudFormation Git Sync

Idéal pour : les équipes CloudFormation qui utilisent un GitOps flux de travail

Git Sync permet aux équipes de plateforme de modéliser CloudFormation des modèles dans un référentiel Git que les équipes de développement peuvent créer. Les développeurs mettent à jour les fichiers de paramètres, transmettent les modifications à leur dépôt bifurqué et Git Sync met à jour la pile.

Principaux avantages :

1. Expérience de développeur similaire à AWS Proton
2. Exploite les connaissances existantes CloudFormation
3. Séparation claire entre les équipes de plateforme et de développement

Limites:

1. Absence de concept d'environnement
2. Aucune fonctionnalité de pipeline avancée
3. S'appuie sur GitHub des fonctionnalités qui ne sont peut-être pas disponibles dans d'autres fournisseurs Git

Pour en savoir plus : Git sync

Harmonix activé AWS

Idéal pour : les entreprises qui ont besoin d'un portail interne complet pour les développeurs

Harmonix est une AWS Partner solution basée sur Backstage.io et fournit un AWS plugin qui permet aux équipes de créer des modèles, des environnements et des services similaires à Proton.

Principaux avantages :

1. Fonctionnalité similaire à AWS Proton
2. Construit sur le célèbre framework Backstage
3. Expérience complète du portail pour développeurs

Limites:

1. Non maintenu par une Service AWS équipe
2. Implémentation de référence pouvant nécessiter une personnalisation

Pour en savoir plus : https://harmonixonaws.io/

AWS CodePipeline et AWS CodeBuild

Idéal pour : les équipes ayant besoin d'un maximum de flexibilité et de contrôle

Utilisez les CI/CD services de AWS base pour reproduire les AWS Proton fonctionnalités avec une flexibilité et un contrôle accrus.

Principaux avantages :

1. Flexibilité maximale
2. Intégration approfondie avec les AWS services
3. Maintenance active et nouvelles fonctionnalités

Limites:

1. Nécessite des travaux de mise en œuvre

2. Moins de out-of-box self-service pour les développeurs

En savoir plus :

Qu'est-ce que AWS CodePipeline

Qu'est-ce que AWS CodeBuild

GitHub Actions

Idéal pour : les petites équipes utilisant des GitHub personnes recherchant la simplicité

>Principaux avantages

1. Intégration facile avec les GitHub référentiels
2. Configuration simple pour les GitHub utilisateurs
3. Grand marché d'actions réutilisables

Limites:

1. Lié à GitHub l'écosystème
2. Peut nécessiter plus de travail pour les contrôles de l'équipe de plateforme

En savoir plus :

GitHub Documentation sur les actions

Exemple CI/CD : intégration aux GitHub actions — CI/CD pipeline pour déployer une application Web sur Amazon EC2

Conseils de migration

Le processus de migration dépend de votre implémentation et de l'alternative choisie. Étapes générales :

1. Faites l'inventaire de vos ressources Proton :
2. Sélectionnez une solution alternative :
3. Extrayez les données de votre modèle :

4. Mettez en œuvre l'alternative que vous avez choisie :
5. Migrez les charges de travail de production :

Pour obtenir une assistance spécifique en matière de migration, AWS Support contactez l'équipe chargée de votre compte.

FAQs

Q : Pourquoi est-ce que c'est AWS l'arrêt AWS Proton ? R : Nous avons identifié de meilleures opportunités pour répondre aux besoins des clients en matière d'infrastructure en appliquant les politiques du Code par le biais AWS d'autres AWS Partner solutions.

Q : Mon infrastructure existante continuera-t-elle à fonctionner après la date de dépréciation ? R : Oui. AWS Proton est avant tout un CI/CD outil. Vos CloudFormation stocks déployés et les ressources qu'ils gèrent resteront intacts et continueront de fonctionner. La dépréciation n'affecte que les pipelines de livraison, et non votre infrastructure déployée.

Q : Comment puis-je obtenir de l'aide pour la migration ? R : AWS Support peut vous aider dans votre migration. Veuillez contacter AWS Supportou vous pouvez contacter votre Compte AWS responsable pour obtenir de l'aide.

Q : Quelle alternative dois-je choisir ? R : La meilleure alternative dépend de votre cas d'utilisation spécifique :

1. Pour un GitOps flux de travail simple : CloudFormation Git Sync
2. Pour les entreprises ayant besoin d'un portail pour développeurs : Harmonix On AWS
3. Pour une flexibilité maximale : AWS CodePipeline et AWS CodeBuild
4. Pour les équipes déjà présentes GitHub : GitHub Actions

Q : Que se passera-t-il si je ne migre pas avant le 7 octobre 2026 ? R : Vous ne pourrez plus y accéder AWS Proton. Votre infrastructure existante continuera de fonctionner, mais vous ne pourrez pas l'utiliser AWS Proton pour la gérer ou la mettre à jour.

Q : Pendant combien de temps mes données seront-elles conservées ? R : Jusqu'au 7 octobre 2026. Après cette date, toutes les données seront supprimées.

Si vous avez d'autres questions, veuillez contacter AWS Support.

Configuration

Effectuez les tâches de cette section afin de pouvoir créer et enregistrer des modèles de service et d'environnement. Vous en avez besoin pour déployer des environnements et des services AWS Proton.

> **Note**
>
> Nous offrons AWS Proton sans frais supplémentaires. Vous pouvez créer, enregistrer et gérer des modèles de service et d'environnement gratuitement. Vous pouvez également compter sur l'autogestion AWS Proton de ses propres opérations, telles que le stockage, la sécurité et le déploiement. Les seules dépenses que vous encourez lors de l'utilisation AWS Proton sont les suivantes.
>
> - Coûts liés au déploiement et à l'utilisation AWS Cloud des ressources que vous avez AWS Proton chargées de déployer et de gérer pour vous.
> - Coûts liés au maintien d'une AWS CodeStar connexion à votre référentiel de code.
> - Coûts de maintenance d'un compartiment Amazon S3, si vous utilisez un compartiment pour fournir des entrées à AWS Proton. Vous pouvez éviter ces coûts si vous passez à the section called “Configurations de synchronisation de modèles” l'utilisation des référentiels Git pour votrethe section called “Packs de modèles”.

Rubriques

Configuration avec IAM

Lorsque vous vous inscrivez AWS, vous êtes automatiquement Compte AWS inscrit à tous les services de AWS, y compris AWS Proton. Seuls les services et les ressources que vous utilisez vous sont facturés.

> **Note**
>
> Vous et votre équipe, y compris les administrateurs et les développeurs, devez tous avoir le même compte.

Inscrivez-vous pour AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

Créer un utilisateur IAM

Afin de créer un utilisateur administrateur, choisissez l'une des options suivantes :

Choisissez un moyen de gérer votre administrateur	Pour	En	Vous pouvez également
Dans IAM Identity Center (Recommandé)	Utiliser des identifiants à court terme pour accéder à AWS. C'est conforme aux bonnes pratiques en matière de sécurité. Pour plus d'informations sur les bonnes pratiques, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.	Suivant les instructions fournies dans Mise en route dans le Guide de l'utilisateur AWS IAM Identity Center .	Configurez l'accès par programmation en configurant le AWS CLI à utiliser AWS IAM Identity Center dans le guide de l'AWS Command Line Interface utilisateur.
Dans IAM (Non recommandé)	Utiliser les informations d'identification à long terme pour accéder à AWS.	Suivant les instructions fournies dans Création d'un utilisateur IAM pour l'accès d'urgence dans le Guide de l'utilisateur IAM.	Configurer l'accès par programmation en suivant les instructions fournies dans Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Configuration des rôles AWS Proton de service

Il existe quelques rôles IAM que vous souhaiterez peut-être créer pour différentes parties de votre AWS Proton solution. Vous pouvez les créer à l'avance à l'aide de la console IAM, ou vous pouvez utiliser la AWS Proton console pour les créer pour vous.

Créez des rôles d' AWS Proton environnement AWS Proton pour permettre d'effectuer des appels d'API vers d'autres services de calcul et de stockage Services AWS CloudFormation AWS CodeBuild, similaires et divers, en votre nom afin de vous fournir des ressources. Un rôle de provisionnement AWS géré est requis lorsqu'un environnement ou l'une des instances de service qui y sont exécutées utilise AWS le provisionnement géré. Un CodeBuildrôle est requis lorsqu'un environnement ou l'une de ses instances de service utilise le CodeBuildprovisionnement. Pour en savoir plus sur les rôles AWS Proton environnementaux, voirthe section called “Rôles IAM”. Lorsque vous créez un environnement, vous pouvez utiliser la AWS Proton console pour choisir un rôle existant pour l'un de ces deux rôles ou pour créer un rôle doté de privilèges administratifs pour vous.

De même, créez des rôles de AWS Proton pipeline pour AWS Proton permettre d'effectuer des appels d'API vers d'autres services en votre nom afin de vous fournir un pipeline CI/CD. Pour en savoir plus sur les rôles du AWS Proton pipeline, voirthe section called “Rôles des services de pipeline”. Pour plus d'informations sur la configuration des CI/CD paramètres, consultezthe section called “Configuration des paramètres du CI/CD pipeline de comptes”.

> **Note**
>
> Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, les rôles que vous créez à l'aide de la console disposent d'autorisations étendues et peuvent être utilisés à la fois comme rôles de service de AWS Proton pipeline et de rôles de AWS Proton service. Pour les déploiements de production, nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées en créant des politiques personnalisées pour les rôles de service de AWS Proton pipeline et les rôles de service d' AWS Proton environnement. Vous pouvez créer et personnaliser ces rôles à l'aide de AWS CLI ou IAM. Pour plus d'informations, consultez Rôles de service pour AWS Proton et Créer un service.

Configuration avec AWS Proton

Si vous souhaitez utiliser le AWS CLI pour exécuter AWS Proton APIs, vérifiez que vous l'avez installé. Si vous ne l'avez pas installé, consultezConfiguration du AWS CLI.

AWS Proton configuration spécifique :

- Pour créer et gérer des modèles :

  - Si vous utilisez des configurations de synchronisation de modèles, configurez une AWS CodeStar connexion.
  - Sinon, configurez un compartiment Amazon S3.
- Pour approvisionner l'infrastructure :
  - Pour un provisionnement autogéré, vous devez configurer une AWS CodeStar connexion.
- (Facultatif) Pour approvisionner des pipelines :
  - Pour le provisionnement AWS géré et le provisionnement CodeBuildbasé, configurez les rôles du pipeline.
  - Pour un provisionnement autogéré, configurez un référentiel de pipelines.

Pour plus d'informations sur les méthodes de provisionnement, consultezthe section called "AWS-provisionnement géré".

Configuration d'un compartiment Amazon S3

Pour configurer un compartiment S3, suivez les instructions de la section Créez votre premier compartiment S3 pour configurer un compartiment S3. Placez vos entrées AWS Proton dans le compartiment où AWS Proton vous pourrez les récupérer. Ces entrées sont appelées ensembles de modèles. Vous pouvez en apprendre davantage à leur sujet dans d'autres sections de ce guide.

Configuration d'une AWS CodeStar connexion

Pour vous connecter AWS Proton à un référentiel, vous créez une AWS CodeStar connexion qui active un pipeline lorsqu'un nouveau commit est effectué sur un référentiel de code source tiers.

AWS Proton utilise la connexion pour :

- Activez un pipeline de services lorsqu'un nouveau commit est effectué sur le code source de votre référentiel.
- Effectuez une pull request sur une infrastructure en tant que référentiel de code.
- Créez une nouvelle version mineure ou majeure du modèle chaque fois qu'un commit est envoyé vers un référentiel de modèles qui modifie l'un de vos modèles, si la version n'existe pas déjà.

Vous pouvez vous connecter aux référentiels Bitbucket GitHub, GitHub Enterprise et GitHub Enterprise Server avec. CodeConnections Pour plus d'informations, consultez CodeConnections dans le Guide de l'utilisateur AWS CodePipeline .

Pour configurer une CodeStar connexion.

1. Ouvrez la AWS Proton console.
2. Dans le volet de navigation, sélectionnez Paramètres, puis Connexions au référentiel pour accéder à la page Connexions dans les paramètres des outils de développement. La page affiche la liste des connexions.
3. Choisissez Créer une connexion et suivez les instructions.

Configuration des paramètres du CI/CD pipeline de comptes

AWS Proton peut fournir des CI/CD pipelines pour déployer du code d'application dans vos instances de service. Les AWS Proton paramètres dont vous avez besoin pour le provisionnement du pipeline dépendent de la méthode de provisionnement que vous avez choisie pour votre pipeline.

AWS-provisionnement géré et CodeBuild basé : configuration des rôles de pipeline

Avec le provisionnement et le provisionnement AWS gérés, CodeBuild AWS Proton provisionnez les pipelines pour vous. Par conséquent, AWS Proton nécessite un rôle de service fournissant des autorisations pour le provisionnement des pipelines. Chacune de ces deux méthodes de provisionnement utilise son propre rôle de service. Ces rôles sont partagés entre tous les pipelines de AWS Proton services et vous ne les configurez qu'une seule fois dans les paramètres de votre compte.

Pour créer des rôles de service de pipeline à l'aide de la console

1. Ouvrez la AWS Proton console.
2. Dans le volet de navigation, choisissez Paramètres, puis Paramètres du compte.
3. Sur la page des CI/CD paramètres du compte, choisissez Configurer.
4. Effectuez l'une des actions suivantes :
   - Pour avoir AWS Proton créé un rôle de service de pipeline pour vous

     [Pour activer le provisionnement AWS géré des pipelines] Sur la page Configurer les paramètres du compte, dans la section du rôle du pipeline de provisionnement AWS géré :

     a. Sélectionnez Nouveau rôle de service.

     b. Entrez un nom pour le rôle, par exemple,**`myProtonPipelineServiceRole`**.

c. Cochez la case pour accepter de créer un AWS Proton rôle doté de privilèges administratifs dans votre compte.

[Pour activer le provisionnement CodeBuild basé sur les pipelines] Sur la page Configurer les paramètres du compte, dans la section Rôle de CodeBuild pipeline, choisissez Rôle de service existant, puis choisissez le rôle de service que vous avez créé dans la section Rôle de CloudFormation pipeline. Ou, si vous n'avez attribué aucun rôle de CloudFormation pipeline, répétez les trois étapes précédentes pour créer un nouveau rôle de service.

- Pour choisir les rôles de service de pipeline existants

  [Pour activer le provisionnement AWS géré des pipelines] Sur la page Configurer les paramètres du compte, dans la section rôle du pipeline de provisionnement AWS géré, choisissez Rôle de service existant, puis choisissez un rôle de service dans votre compte. AWS

  [Pour activer le CodeBuild provisionnement des pipelines] Sur la page Configurer les paramètres du compte, dans la section Rôle de provisionnement du CodeBuild pipeline, choisissez Rôle de service existant, puis choisissez un rôle de service dans votre AWS compte.

5. Sélectionnez Enregistrer les modifications.

   Votre nouveau rôle de service de pipeline est affiché sur la page des paramètres du compte.

Approvisionnement autogéré : configurez un référentiel de pipelines

Avec le provisionnement autogéré, AWS Proton envoie une pull request (PR) à un référentiel de provisionnement que vous avez configuré, et votre code d'automatisation est responsable du provisionnement des pipelines. Par conséquent, AWS Proton il n'a pas besoin d'un rôle de service pour approvisionner des pipelines. Il a plutôt besoin d'un référentiel de provisioning enregistré. Votre code d'automatisation dans le référentiel doit assumer un rôle approprié qui fournit des autorisations pour le provisionnement des pipelines.

Pour enregistrer un référentiel de provisionnement de pipelines à l'aide de la console

1. Créez un référentiel de provisionnement de CI/CD pipelines si vous n'en avez pas encore créé un. Pour plus d'informations sur les pipelines dans le cadre du provisionnement autogéré, consultez. the section called “Provisionnement autogéré”

2. Dans le volet de navigation, choisissez Paramètres, puis Paramètres du compte.
3. Sur la page des CI/CD paramètres du compte, choisissez Configurer.
4. Sur la page Configurer les paramètres du compte, dans la section du référentiel du pipeline CI/CD :

   a. Sélectionnez Nouveau référentiel, puis choisissez l'un des fournisseurs de référentiels.

   b. Pour la CodeStar connexion, choisissez l'une de vos connexions.

   > **Note**
   >
   > Si vous n'êtes pas encore connecté au compte du fournisseur de référentiel concerné, choisissez Ajouter une nouvelle CodeStar connexion, terminez le processus de création de connexion, puis cliquez sur le bouton d'actualisation situé à côté du menu de CodeStarconnexion. Vous devriez maintenant pouvoir choisir votre nouvelle connexion dans le menu.

   c. Dans Nom du référentiel, choisissez le référentiel de provisionnement de votre pipeline. Le menu déroulant affiche la liste des référentiels du compte fournisseur.

   d. Dans Nom de la branche, choisissez l'une des branches du référentiel.
5. Sélectionnez Enregistrer les modifications.

   Le référentiel de votre pipeline est affiché sur la page des paramètres du compte.

Configuration du AWS CLI

Pour utiliser le AWS CLI pour effectuer des appels d' AWS Proton API, vérifiez que vous avez installé la dernière version du AWS CLI. Pour plus d'informations, consultez Mise en route avec le AWS CLI dans le AWS Command Line Interface Guide de l'utilisateur. Ensuite, pour commencer à utiliser le AWS CLI with AWS Proton, voirthe section called "Commencer à utiliser la CLI".

Commencer avec AWS Proton

Avant de commencer, configurez-vous pour utiliser AWS Proton et vérifiez que vous avez rempli les conditions requises pour le démarrage.

Commencez AWS Proton par choisir un ou plusieurs des chemins suivants :

- Suivez un exemple de flux de travail guidé par une console ou une CLI via les liens de documentation.
- Exécutez un exemple de flux de travail guidé sur console.
- Exécutez un exemple de AWS CLI flux de travail guidé.

Rubriques

Conditions préalables

Avant de commencer à l'utiliser AWS Proton, assurez-vous que les conditions préalables suivantes sont remplies. Pour de plus amples informations, veuillez consulter Configuration.

- Vous disposez d'un compte IAM avec des autorisations d'administrateur. Pour de plus amples informations, veuillez consulter Configuration avec IAM.
- Vous avez le rôle AWS Proton de service et le rôle de service de AWS Proton pipeline sont associés à votre compte. Pour plus d'informations, consultez Configuration des rôles AWS Proton de service et Rôles de service pour AWS Proton.
- Vous avez un AWS CodeStar lien. Pour de plus amples informations, veuillez consulter Configuration d'une AWS CodeStar connexion.
- Vous êtes habitué à créer des CloudFormation modèles et à paramétrer Jinja. Pour plus d'informations, voir Qu'est-ce que c'est CloudFormation ? dans le guide de l' CloudFormation utilisateur et sur le site Web de Jinja.

- Vous avez une connaissance pratique des services AWS d'infrastructure.
- Vous êtes connecté à votre Compte AWS.

Flux de travail de démarrage

Apprenez à créer des ensembles de modèles, à créer et à enregistrer des modèles, ainsi qu'à créer des environnements et des services en suivant les exemples d'étapes et les liens.

Avant de commencer, vérifiez que vous avez créé un rôle AWS Proton de service.

Si votre modèle de service inclut un pipeline AWS Proton de service, vérifiez que vous avez créé une AWS CodeStar connexion et un rôle de service de AWS Proton pipeline.

Pour plus d'informations, consultez The AWS Proton service API Reference.

Exemple : flux de travail de démarrage

1. Reportez-vous au schéma ci-dessous Comment AWS Proton fonctionne pour une vue de haut niveau des AWS Proton entrées et des sorties.
2. Créez un ensemble d'environnements et un ensemble de modèles de services.
    a. Identifiez les paramètres d'entrée.
    b. Créez un fichier de schéma.
    c. Créez des fichiers d'infrastructure sous forme de code (IaC).
    d. Pour compléter votre ensemble de modèles, créez un fichier manifeste et organisez vos fichiers iAc, vos fichiers manifestes et vos fichiers de schéma dans des répertoires.
    e. Rendez votre ensemble de modèles accessible à AWS Proton.
3. Créez et enregistrez une version de modèle d'environnement avec AWS Proton.

    Lorsque vous utilisez la console pour créer et enregistrer un modèle, une version du modèle est automatiquement créée.

    Lorsque vous utilisez le AWS CLI pour créer et enregistrer un modèle :

    a. Créez un modèle d'environnement.
    b. Créez une version du modèle d'environnement.

Pour plus d'informations, consultez CreateEnvironmentTemplateet CreateEnvironmentTemplateVersiondans la référence de AWS Proton l'API.

4. Publiez votre modèle d'environnement pour le rendre utilisable.

   Pour plus d'informations, consultez UpdateEnvironmentTemplateVersionla référence de AWS Proton l'API.

5. Pour créer un environnement, sélectionnez une version de modèle d'environnement publiée et fournissez des valeurs pour les entrées requises.

   Pour plus d'informations, consultez CreateEnvironmentla référence de AWS Proton l'API.

6. Créez et enregistrez une version de modèle de service avec AWS Proton.

   Lorsque vous utilisez la console pour créer et enregistrer un modèle, une version du modèle est automatiquement créée.

   Lorsque vous utilisez le AWS CLI pour créer et enregistrer un modèle :

   a. Créez un modèle de service.
   b. Créez une version de modèle de service.

   Pour plus d'informations, consultez CreateServiceTemplateet CreateServiceTemplateVersiondans la référence de AWS Proton l'API.

7. Publiez votre modèle de service pour le rendre utilisable.

   Pour plus d'informations, consultez UpdateServiceTemplateVersionla référence de AWS Proton l'API.

8. Pour créer un service, sélectionnez une version de modèle de service publiée et fournissez des valeurs pour les entrées requises.

   Pour plus d'informations, consultez CreateServicela référence de AWS Proton l'API.

Commencer à utiliser le AWS Management Console

Commencez avec AWS Proton

- Créez et visualisez un modèle d'environnement.

- Créez, visualisez et publiez un modèle de service qui utilise le modèle d'environnement que vous venez de créer.
- Créez un environnement et un service (facultatif).
- Supprimez le modèle de service, le modèle d'environnement, l'environnement et le service, s'ils ont été créés.

Étape 1 : ouvrir la AWS Proton console

- Ouvrez la console AWS Proton.

Étape 2 : Préparez-vous à utiliser les modèles d'exemple

1. Créez une connexion Codestar à Github et nommez la connexion. my-proton-connection
2. Accédez à https://github.com/aws-samples/aws-proton-cloudformation-sample-templates
3. Créez un fork du dépôt dans votre compte Github.

Étape 3 : Création d'un modèle d'environnement

Dans le volet de navigation, sélectionnez Modèles d'environnement.

1. Sur la page Modèles d'environnement, choisissez Créer un modèle d'environnement.
2. Sur la page Créer un modèle d'environnement, dans la section Options du modèle, choisissez Créer un modèle pour le provisionnement de nouveaux environnements.
3. Dans la section Source du bundle de modèles, choisissez Synchroniser un ensemble de modèles depuis Git.
4. Dans la section Référentiel de définition du modèle, sélectionnez Choisir un dépôt Git lié.
5. Sélectionnez dans my-proton-connectionla liste des référentiels.
6. Sélectionnez principal dans la liste des branches.
7. Dans la section des détails du modèle d'environnement Proton.

    a. Entrez le nom du modèle sous la forme**`fargate-env`**.

    b. Entrez le nom d'affichage du modèle d'environnement sous la forme**`My Fargate Environment`**.

    c. (Facultatif) Entrez une description pour le modèle d'environnement.

8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez Créer un modèle d'environnement.

   Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de balises gérées par le client AWS et une liste de balises gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des AWS Proton ressources. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.
10. Le statut d'un nouveau modèle d'environnement commence à l'état Brouillon. Vous et les autres personnes `proton:CreateEnvironment` autorisées pouvez le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.
11. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans la bannière d'alerte d'information et ignorer l'étape suivante.
12. Dans la section Versions du modèle, choisissez Publier.
13. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.
14. Dans le volet de navigation, sélectionnez Modèles d'environnement.

    Une nouvelle page affiche la liste de vos modèles d'environnement ainsi que les détails des modèles.

Étape 4 : Création d'un modèle de service

Créez un modèle de service.

1. Dans le volet de navigation, sélectionnez Modèles de services.
2. Sur la page Modèles de service, choisissez Créer un modèle de service.
3. Sur la page Créer un modèle de service, dans la section Source du bundle de modèles, choisissez Synchroniser un ensemble de modèles depuis Git.
4. Dans la section Modèle, sélectionnez Choisir un dépôt Git lié.
5. Sélectionnez dans my-proton-connectionla liste des référentiels.
6. Sélectionnez principal dans la liste des branches.
7. Dans la section des détails du modèle de service Proton.

a. Entrez le nom du modèle de service sous la forme**`backend-fargate-svc`**.

b. Entrez le nom d'affichage du modèle de service sous la forme**`My Fargate Service`**.

c. (Facultatif) Entrez une description pour le modèle de service.

8. Dans la section Modèles d'environnement compatibles.

   - Cochez la case située à gauche du modèle d'environnement My Fargate Environment pour sélectionner le modèle d'environnement compatible pour le nouveau modèle de service.

9. Pour les paramètres de chiffrement, conservez les valeurs par défaut.

10. Dans la section Définition du pipeline.

    - Maintenez le bouton Ce modèle inclut un CI/CD pipeline sélectionné.

11. Choisissez Créer un modèle de service.

    Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle de service, y compris une liste de balises gérées par le client AWS et des balises gérées par le client.

12. Le statut d'un nouveau modèle de service commence à l'état Brouillon. Seuls les administrateurs peuvent le consulter et y accéder. Pour mettre le modèle de service à la disposition des développeurs, procédez à l'étape suivante.

13. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans la bannière d'alerte d'information et ignorer l'étape suivante.

14. Dans la section Versions du modèle, choisissez Publier.

15. Le statut du modèle passe à Publié.

    La première version mineure de votre modèle de service est publiée et peut être utilisée par les développeurs. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.

16. Dans le volet de navigation, sélectionnez Modèles de services.

    Une nouvelle page affiche la liste de vos modèles de services et de leurs détails.

Étape 5 : Création d'un environnement

Dans le panneau de navigation, choisissez Environments (Environnements).

1. Choisissez Create environment.
2. Sur la page Choisir un modèle d'environnement, sélectionnez le modèle que vous venez de créer. Il s'appelle My Fargate Environment. Choisissez ensuite Configurer.
3. Sur la page Configurer l'environnement, dans la section Provisioning, choisissez Provisioning through AWS Proton.
4. Dans la section Compte de déploiement, sélectionnez Ceci Compte AWS.
5. Dans Paramètres d'environnement, entrez le nom de l'environnement sous la forme**`my-fargate-environment`**.
6. Dans la section Rôles environnementaux, sélectionnez Nouveau rôle de service ou, si vous avez déjà créé un rôle de AWS Proton service, sélectionnez Rôle de service existant.

   a. Sélectionnez Nouveau rôle de service pour créer un nouveau rôle.

      i. Entrez le nom du rôle d'environnement sous la forme**`MyProtonServiceRole`**.

      ii. Cochez la case pour accepter de créer un rôle AWS Proton de service avec des privilèges administratifs pour votre compte.

   b. Sélectionnez Rôle de service existant pour utiliser un rôle existant.

      - Sélectionnez votre rôle dans le champ déroulant Nom du rôle environnemental.
7. Choisissez Suivant.
8. Sur la page Configurer les paramètres personnalisés, utilisez les valeurs par défaut.
9. Choisissez Next et passez en revue vos entrées.
10. Choisissez Créer.

    Consultez les détails et l'état de l'environnement, ainsi que les balises AWS gérées et les balises gérées par le client pour votre environnement.
11. Dans le panneau de navigation, choisissez Environments (Environnements).

    Une nouvelle page affiche la liste de vos environnements ainsi que leur statut et d'autres détails relatifs à l'environnement.

Étape 6 : Facultatif - Création d'un service et déploiement d'une application

1. Ouvrez la AWS Proton console.
2. Dans le panneau de navigation, choisissez Services.

3. Sur la page Services, choisissez Créer un service.
4. Sur la page Choisir un modèle de service, sélectionnez le modèle My Fargate Service en cliquant sur le bouton radio situé dans le coin supérieur droit de la fiche modèle.
5. Choisissez Configurer dans le coin inférieur droit de la page.
6. Sur la page Configurer le service, dans la section Paramètres du service, entrez le nom du service**`my-service`**.
7. (Facultatif) Entrez une description du service.
8. Dans la section Paramètres du référentiel de services :
    a. Pour CodeStar la connexion, choisissez votre connexion dans la liste.
    b. Pour Nom du référentiel, choisissez le nom de votre référentiel de code source dans la liste.
    c. Pour Nom de la branche, choisissez le nom de la branche de votre référentiel de code source dans la liste.
9. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client. Ensuite, sélectionnez Suivant.
10. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, dans la section Nouvelle instance, suivez les étapes suivantes pour fournir des valeurs personnalisées pour les paramètres de vos instances de service.
    a. Entrez le nom de l'instance**`my-app-service`**.
    b. Choisissez l'environnement **`my-fargate-environment`** de votre instance de service.
    c. Conservez les valeurs par défaut pour les paramètres d'instance restants.
    d. Conservez les valeurs par défaut pour les entrées du pipeline.
    e. Choisissez Next et passez en revue vos entrées.
    f. Choisissez Créer et consultez l'état et les détails de votre service.
11. Sur la page des détails du service, consultez l'état de votre instance de service et de votre pipeline en choisissant les onglets Vue d'ensemble et Pipeline. Sur ces pages, vous pouvez également consulter les AWS tags gérés par les clients. AWS Proton crée automatiquement des tags AWS gérés pour vous. Choisissez Gérer les balises pour créer et modifier les balises gérées par le client. Pour plus d'informations sur le balisage, consultez AWS Proton ressources et balisage.
12. Une fois le service actif, dans l'onglet Vue d'ensemble, dans la section Instances de service, choisissez le nom de votre instance de service my-app-service.

Vous êtes maintenant sur la page détaillée de l'instance de service.

13. Pour afficher votre application, dans la section Sorties, copiez le ServiceEndpointlien dans votre navigateur.

    Vous voyez un AWS Proton graphique sur la page Web.

14. Une fois le service créé, dans le volet de navigation, choisissez Services pour afficher la liste de vos services.

Étape 7 : Nettoyez

1. Ouvrez la AWS Proton console.
2. Supprimer un service (si vous en avez créé un)
    a. Dans le panneau de navigation, choisissez Services.
    b. Sur la page Services, choisissez le nom du service my-service.

       Vous êtes maintenant sur la page détaillée du service de my-service.
    c. Dans le coin supérieur droit de la page, choisissez Actions, puis Supprimer.
    d. Un modal vous invite à confirmer l'action de suppression.
    e. Suivez les instructions et choisissez Oui, supprimer.
3. Supprimer un environnement
    a. Dans le panneau de navigation, choisissez Environments (Environnements).
    b. Sur la page Environnements, sélectionnez le bouton radio situé à gauche de l'environnement que vous venez de créer.
    c. Choisissez Actions, puis Supprimer.
    d. Un modal vous invite à confirmer l'action de suppression.
    e. Suivez les instructions et choisissez Oui, supprimer.
4. Supprimer un modèle de service
    a. Dans le volet de navigation, sélectionnez Modèles de services.
    b. Sur la page Modèles de service, sélectionnez le bouton radio situé à gauche du modèle de service my-svc-template.
    c. Choisissez Actions, puis Supprimer.

d. Un modal vous invite à confirmer l'action de suppression.

e. Suivez les instructions et choisissez Oui, supprimer. Cela supprime le modèle de service et toutes ses versions.

5. Supprimer un modèle d'environnement

a. Dans le volet de navigation, sélectionnez Modèles d'environnement.

b. Sur la page Modèles d'environnement, sélectionnez le bouton radio situé à gauche de my-env-template.

c. Choisissez Actions, puis Supprimer.

d. Un modal vous invite à confirmer l'action de suppression.

e. Suivez les instructions et choisissez Oui, supprimer. Cela supprime le modèle d'environnement et toutes ses versions.

6. Supprimer votre connexion Codestar

Commencer à utiliser le AWS CLI

Pour commencer à AWS Proton utiliser le AWS CLI, suivez ce didacticiel. Le didacticiel présente un AWS Proton service d'équilibrage de charge destiné au public basé sur. AWS Fargate Le didacticiel fournit également un CI/CD pipeline qui déploie un site Web statique avec une image affichée.

Avant de commencer, assurez-vous que vous êtes correctement configuré. Pour en savoir plus, consultez the section called “Conditions préalables”.

Étape 1 : enregistrer un modèle d'environnement

Au cours de cette étape, en tant qu'administrateur, vous enregistrez un exemple de modèle d'environnement, qui contient un cluster Amazon Elastic Container Service (Amazon ECS) et un Amazon Virtual Private Cloud (Amazon VPC) avec deux sous-réseaux. public/private

Pour enregistrer un modèle d'environnement

1. Intégrez le référentiel AWS Proton d'exemples CloudFormation de modèles à votre GitHub compte ou à votre organisation. Ce référentiel inclut les modèles d'environnement et de service que nous utilisons dans ce didacticiel.

   Enregistrez ensuite votre dépôt bifurqué auprès AWS Proton de. Pour de plus amples informations, veuillez consulter the section called “Création d'un lien de référentiel”.

2. Créez un modèle d'environnement.

   La ressource du modèle d'environnement suit les versions du modèle d'environnement.

```
$ aws proton create-environment-template \
  --name "fargate-env" \
  --display-name "Public VPC Fargate" \
  --description "VPC with public access and ECS cluster"
```

3. Créez une configuration de synchronisation de modèles.

   AWS Proton définit une relation de synchronisation entre votre référentiel et votre modèle d'environnement. Il crée ensuite la version 1.0 du modèle dans `DRAFT` status.

```
$ aws proton create-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "environment-templates/fargate-env"
```

4. Attendez que la version du modèle d'environnement soit correctement enregistrée.

   Lorsque cette commande revient avec un statut de sortie de`0`, l'enregistrement de la version est terminé. Cela est utile dans les scripts pour garantir que vous pouvez exécuter correctement la commande à l'étape suivante.

```
$ aws proton wait environment-template-version-registered \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0"
```

5. Publiez la version du modèle d'environnement pour la rendre disponible pour la création de l'environnement.

```
$ aws proton update-environment-template-version \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

Étape 2 : enregistrer un modèle de service

Au cours de cette étape, en tant qu'administrateur, vous enregistrez un exemple de modèle de service, qui contient toutes les ressources nécessaires pour fournir un service Amazon ECS Fargate via un équilibreur de charge et CI/CD un pipeline qui l'utilise. AWS CodePipeline

Pour enregistrer un modèle de service

1. Créez un modèle de service.

   La ressource de modèle de service suit les versions des modèles de service.

   ```
   $ aws proton create-service-template \
     --name "load-balanced-fargate-svc" \
     --display-name "Load balanced Fargate service" \
     --description "Fargate service with an application load balancer"
   ```

2. Créez une configuration de synchronisation de modèles.

   AWS Proton définit une relation de synchronisation entre votre référentiel et votre modèle de service. Il crée ensuite la version 1.0 du modèle dans `DRAFT` status.

   ```
   $ aws proton create-template-sync-config \
     --template-name "load-balanced-fargate-svc" \
     --template-type "SERVICE" \
     --repository-name "your-forked-repo" \
     --repository-provider "GITHUB" \
     --branch "your-branch" \
     --subdirectory "service-templates/load-balanced-fargate-svc"
   ```

3. Attendez que la version du modèle de service soit correctement enregistrée.

   Lorsque cette commande revient avec un statut de sortie de`0`, l'enregistrement de la version est terminé. Cela est utile dans les scripts pour garantir que vous pouvez exécuter correctement la commande à l'étape suivante.

   ```
   $ aws proton wait service-template-version-registered \
     --template-name "load-balanced-fargate-svc" \
     --major-version "1" \
     --minor-version "0"
   ```

4. Publiez la version du modèle de service pour la rendre disponible pour la création du service.

```
$ aws proton update-service-template-version \
  --template-name "load-balanced-fargate-svc" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

Étape 3 : Déploiement d'un environnement

Au cours de cette étape, en tant qu'administrateur, vous instanciez un AWS Proton environnement à partir du modèle d'environnement.

Pour déployer un environnement

1. Obtenez un exemple de fichier de spécifications pour le modèle d'environnement que vous avez enregistré.

   Vous pouvez télécharger le fichier `environment-templates/fargate-env/spec/spec.yaml` depuis le référentiel d'exemples de modèles. Vous pouvez également récupérer l'intégralité du référentiel localement et exécuter la create-environment commande depuis le `environment-templates/fargate-env` répertoire.
2. Créez un environnement.

   AWS Proton lit les valeurs d'entrée de votre spécification d'environnement, les combine avec votre modèle d'environnement et provisionne les ressources environnementales de votre AWS compte en utilisant votre rôle de AWS Proton service.

   ```
   $ aws proton create-environment \
     --name "fargate-env-prod" \
     --template-name "fargate-env" \
     --template-major-version 1 \
     --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
    \
     --spec "file://spec/spec.yaml"
   ```
3. Attendez que l'environnement soit correctement déployé.

   ```
   $ aws proton wait environment-deployed --name "fargate-env-prod"
   ```

Étape 4 : Déployer un service [développeur d'applications]

Au cours des étapes précédentes, un administrateur a enregistré et publié un modèle de service et a déployé un environnement. En tant que développeur d'applications, vous pouvez désormais créer un AWS Proton service et le déployer dans l' AWS Proton environnement

Pour déployer un service

1. Obtenez un exemple de fichier de spécifications pour le modèle de service enregistré par l'administrateur.

   Vous pouvez télécharger le fichier `service-templates/load-balanced-fargate-svc/spec/spec.yaml` depuis le référentiel d'exemples de modèles. Vous pouvez également récupérer l'intégralité du référentiel localement et exécuter la create-service commande depuis le `service-templates/load-balanced-fargate-svc` répertoire.
2. Intégrez le référentiel AWS Proton Sample Services à votre GitHub compte ou à votre organisation. Ce référentiel inclut le code source de l'application que nous utilisons dans ce didacticiel.
3. Créer un service.

   AWS Proton lit les valeurs d'entrée de votre spécification de service, les combine avec votre modèle de service et fournit les ressources de service de votre AWS compte dans l'environnement spécifié dans la spécification. Un AWS CodePipeline pipeline déploie le code de votre application à partir du référentiel que vous spécifiez dans la commande.

   ```
   $ aws proton create-service \
     --name "static-website" \
     --repository-connection-arn \
       "arn:aws:codestar-connections:us-east-1:123456789012:connection/your-codestar-connection-id" \
     --repository-id "your-GitHub-account/aws-proton-sample-services" \
     --branch-name "main" \
     --template-major-version 1 \
     --template-name "load-balanced-fargate-svc" \
     --spec "file://spec/spec.yaml"
   ```
4. Attendez que le service soit correctement déployé.

   ```
   $ aws proton wait service-created --name "static-website"
   ```

5. Récupérez les résultats et consultez votre nouveau site Web.

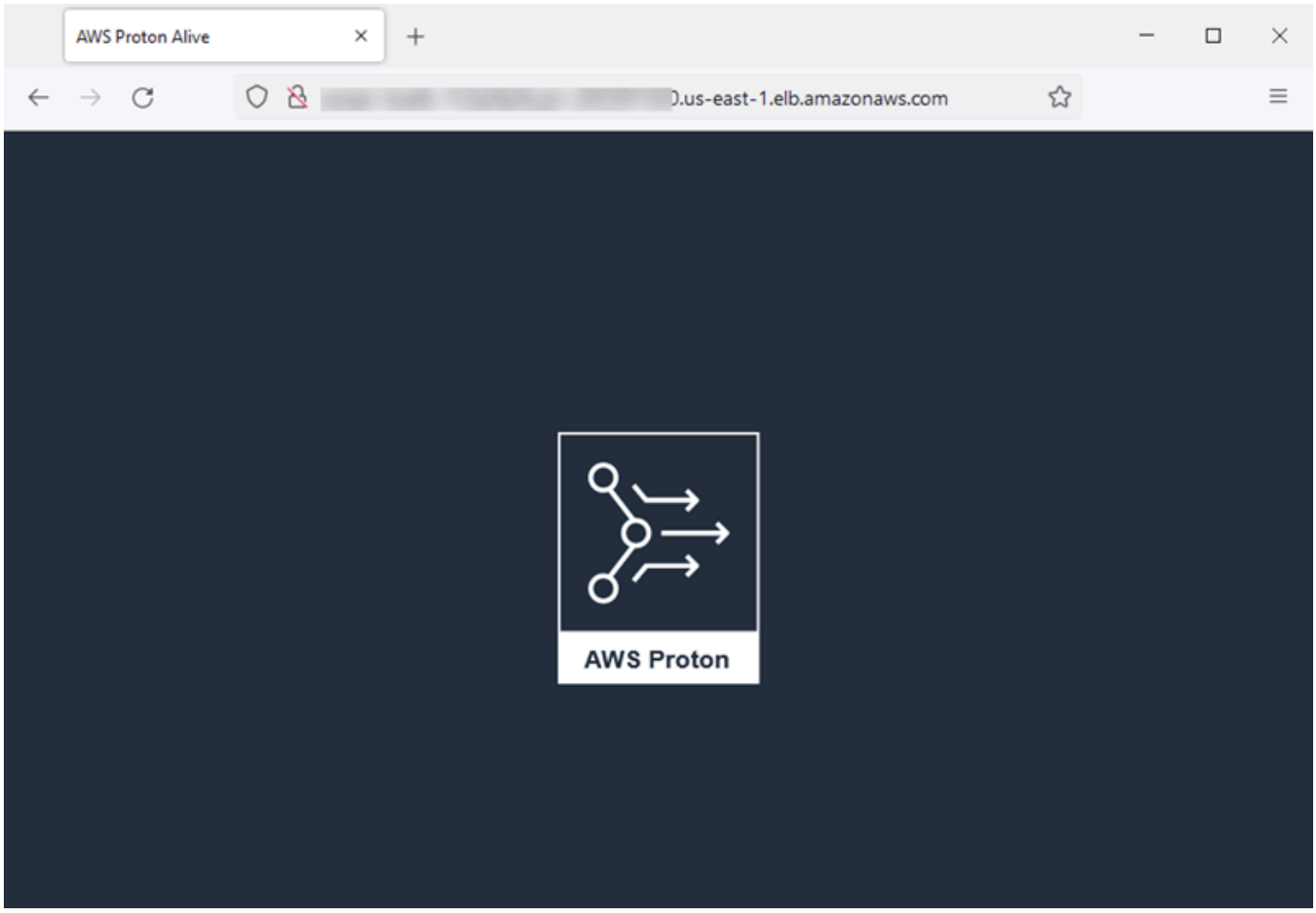
   Exécutez la commande suivante :

   ```
   $ aws proton list-service-instance-outputs \
     --service-name "static-website" \
     --service-instance-name load-balanced-fargate-svc-prod
   ```

   Le résultat de la commande doit être similaire à ce qui suit :

   ```
   {
       "outputs": [
           {
               "key": "ServiceURL",
               "valueString": "http://your-service-endpoint.us-
   east-1.elb.amazonaws.com"
           }
       ]
   }
   ```

   La valeur de la sortie de l'`ServiceURL`instance est le point de terminaison de votre nouveau site Web de service. Utilisez votre navigateur pour y accéder. Vous devriez voir le graphique suivant sur une page statique :



Étape 5 : Nettoyage (facultatif)

Au cours de cette étape, lorsque vous avez terminé d'explorer les AWS ressources que vous avez créées dans le cadre de ce didacticiel, et pour économiser sur les coûts associés à ces ressources, vous les supprimez.

Pour supprimer les ressources du didacticiel

1. Pour supprimer le service, exécutez la commande suivante :

```
$ aws proton delete-service --name "static-website"
```

2. Pour supprimer l'environnement, exécutez la commande suivante :

```
$ aws proton delete-environment --name "fargate-env-prod"
```

3. Pour supprimer le modèle de service, exécutez les commandes suivantes :

```
$ aws proton delete-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE"
$ aws proton delete-service-template --name "load-balanced-fargate-svc"
```

4. Pour supprimer le modèle d'environnement, exécutez les commandes suivantes :

```
$ aws proton delete-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT"
$ aws proton delete-environment-template --name "fargate-env"
```

La bibliothèque AWS Proton de modèles

L' AWS Proton équipe gère une bibliothèque d'exemples de modèles sur GitHub. La bibliothèque inclut des exemples de fichiers d'infrastructure sous forme de code (IaC) pour de nombreux scénarios courants d'environnement et d'infrastructure d'applications.

La bibliothèque de modèles est stockée dans deux GitHub référentiels :

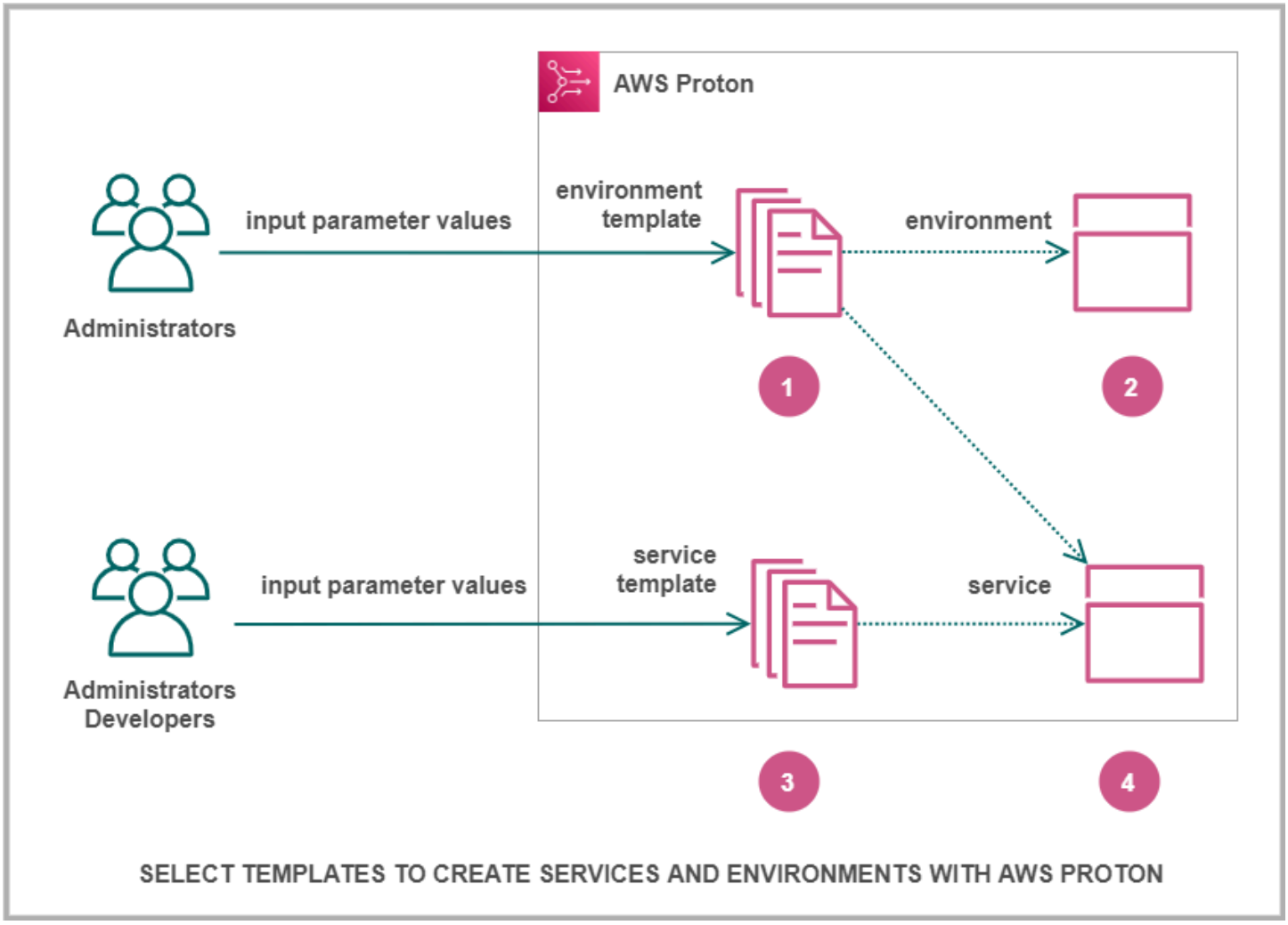
- aws-proton-cloudformation-sample-templates — Exemples de ensembles de modèles qui utilisent AWS CloudFormationJinja comme langage IaC. Vous pouvez utiliser ces exemples pour les AWS-provisionnement géré environnements.
- aws-proton-terraform-sample-templates — Exemples de bundles de modèles qui utilisent Terraform comme langage IaC. Vous pouvez utiliser ces exemples pour les Provisionnement autogéré environnements.

Chacun de ces référentiels possède un `README` fichier contenant des informations complètes sur le contenu et la structure du référentiel. Chaque exemple contient des informations sur le cas d'utilisation couvert par le modèle, son architecture et les paramètres d'entrée utilisés par le modèle.

Vous pouvez utiliser les modèles de cette bibliothèque directement en connectant l'un des référentiels de la bibliothèque à votre GitHub compte. Vous pouvez également utiliser ces exemples comme point de départ pour développer votre environnement et vos modèles de services.

Comment AWS Proton fonctionne

Avec AWS Proton, vous provisionnez des environnements, puis des services exécutés dans ces environnements. Les environnements et les services sont basés sur des modèles d'environnement et de service, respectivement, que vous choisissez dans votre bibliothèque de modèles AWS Proton versionnés.



1 Lorsqu

en tant qu'administrateur, vous sélectionnez un modèle d'environnement avec AWS Proton, vous fournissez des valeurs pour les paramètres d'entrée requis.

2

AWS Proton utilise le modèle d'environnement et les valeurs des paramètres pour approvisionner votre environnement.

3

Lorsqu

en tant que développeur ou administrateur, vous sélectionnez un modèle de service avec AWS Proton, vous fournissez des valeurs pour les paramètres d'entrée requis. Vous sélectionnez également un environnement dans lequel déployer votre application ou votre service.

4

AWS Proton utilise le modèle de service, ainsi que les valeurs de votre service et des paramètres d'environnement sélectionnés, pour fournir votre service.

Vous fournissez des valeurs pour les paramètres d'entrée afin de personnaliser votre modèle en vue de sa réutilisation et de multiples cas d'utilisation, applications ou services.

Pour que cela fonctionne, vous créez des ensembles de modèles d'environnement ou de service et vous les téléchargez respectivement dans des modèles d'environnement ou de service enregistrés.

Les ensembles de modèles contiennent tout ce dont vous AWS Proton avez besoin pour fournir des environnements ou des services.

Lorsque vous créez un modèle d'environnement ou de service, vous téléchargez un ensemble de modèles contenant les fichiers d'infrastructure paramétrée sous forme de code (IaC) AWS Proton utilisés pour fournir des environnements ou des services.

Lorsque vous sélectionnez un modèle d'environnement ou de service pour créer ou mettre à jour un environnement ou un service, vous fournissez des valeurs pour les paramètres du fichier iAC du bundle de modèles.

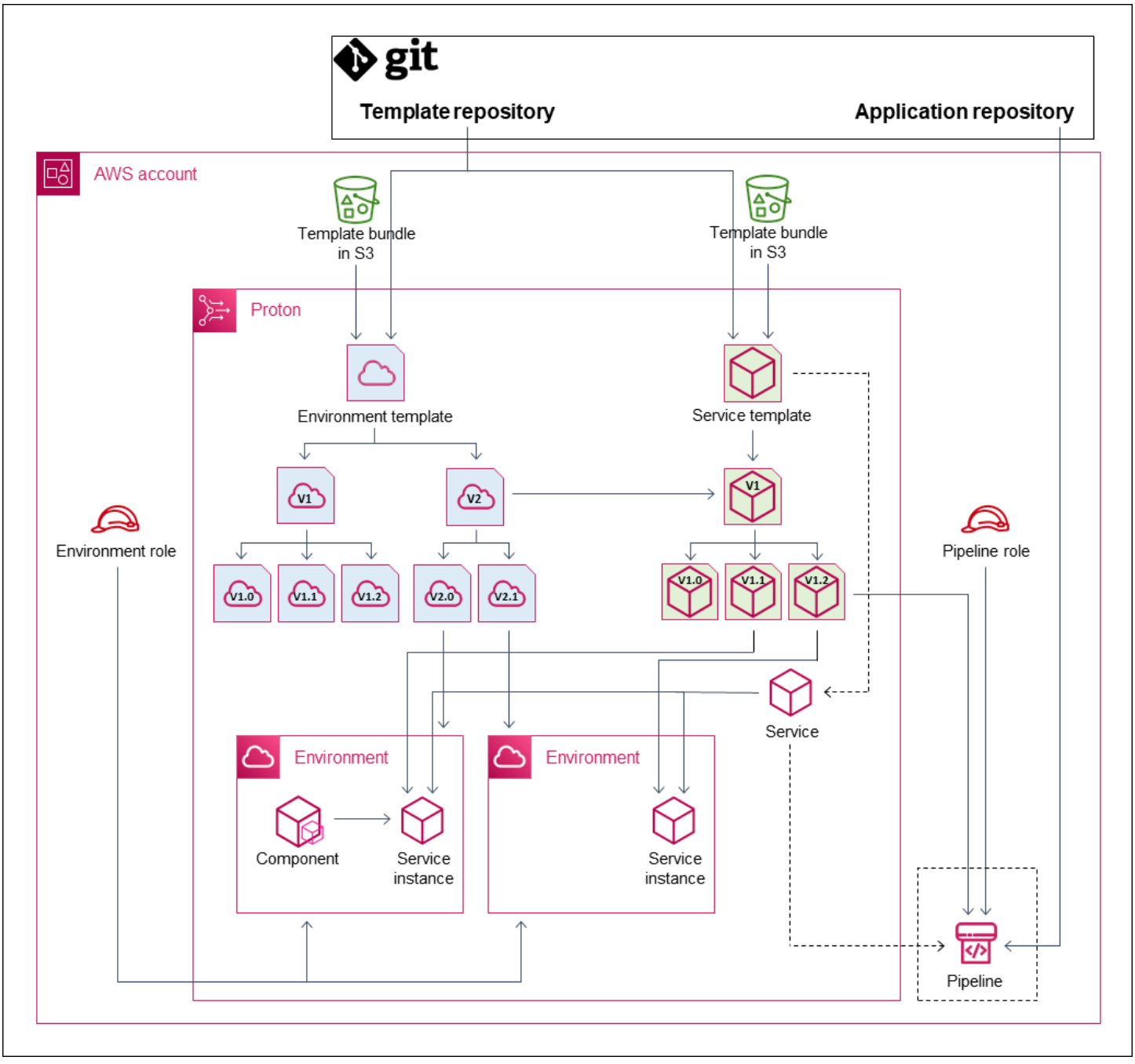
Rubriques

- AWS Proton objets
- Comment AWS Proton approvisionne l'infrastructure
- AWS Proton terminologie

AWS Proton objets

Le schéma suivant montre les principaux AWS Proton objets et leur relation avec d'autres objets AWS et des objets tiers. Les flèches représentent le sens du flux de données (sens inverse de la dépendance).

Nous suivons le schéma avec de brèves descriptions et des liens de référence pour ces AWS Proton objets.



- Modèle d'environnement : collection de versions de modèles d'environnement qui peuvent être utilisées pour créer AWS Proton des environnements.

  Pour plus d'informations, consultez Création de modèles et d'ensembles et Modèles.
- Version du modèle d'environnement : version spécifique d'un modèle d'environnement. Prend un ensemble de modèles en entrée, provenant soit d'un compartiment S3, soit d'un dépôt Git. Le

bundle spécifie l'infrastructure en tant que code (IaC) et les paramètres d'entrée associés pour un AWS Proton environnement.

Pour plus d'informations, consultez the section called "Versions", the section called "Publication" et the section called "Configurations de synchronisation de modèles".

- Environnement : ensemble de ressources d' AWS infrastructure partagées et de politiques d'accès dans AWS Proton lesquelles les services sont déployés. AWS les ressources sont mises en service à l'aide d'une version de modèle d'environnement invoquée avec des valeurs de paramètres spécifiques. Les politiques d'accès sont fournies dans un rôle de service.

  Pour de plus amples informations, veuillez consulter Environnements.

- Modèle de service : collection de versions de modèles de services qui peuvent être utilisées pour créer des AWS Proton services.

  Pour plus d'informations, consultez Création de modèles et d'ensembles et Modèles.

- Version du modèle de service : version spécifique d'un modèle de service. Prend un ensemble de modèles en entrée, provenant soit d'un compartiment S3, soit d'un dépôt Git. Le bundle spécifie l'infrastructure en tant que code (IaC) et les paramètres d'entrée associés pour un AWS Proton service.

  Une version de modèle de service spécifie également les contraintes suivantes sur les instances de service en fonction de la version :

  - Modèles d'environnement compatibles : les instances ne peuvent s'exécuter que dans des environnements basés sur ces modèles d'environnement compatibles.
  - Sources de composants prises en charge : types de composants que les développeurs peuvent associer aux instances.

  Pour plus d'informations, consultez the section called "Versions", the section called "Publication" et the section called "Configurations de synchronisation de modèles".

- Service : ensemble d'instances de service qui exécutent une application à l'aide des ressources spécifiées dans un modèle de service, et éventuellement un CI/CD pipeline qui déploie le code de l'application dans ces instances.

  Dans le diagramme, la ligne pointillée du modèle de service signifie que le service transmet le modèle aux instances de service et au pipeline.

  Pour de plus amples informations, veuillez consulter Services.

- Instance de service : ensemble de ressources d' AWS infrastructure qui exécutent une application dans un AWS Proton environnement spécifique. AWS les ressources sont fournies à l'aide d'une version de modèle de service invoquée avec des valeurs de paramètres spécifiques.

  Pour plus d'informations, consultez Services et the section called "Mettre à jour l'instance".

- Pipeline : CI/CD pipeline facultatif qui déploie une application dans les instances d'un service, avec des politiques d'accès pour approvisionner ce pipeline. Les politiques d'accès sont fournies dans un rôle de service. Un service n'est pas toujours associé à un AWS Proton pipeline. Vous pouvez choisir de gérer les déploiements de code de votre application en dehors de celui-ci. AWS Proton

  Dans le diagramme, la ligne pointillée correspondant à Service et la case en pointillés autour de Pipeline signifient que si vous choisissez de gérer vous-même vos CI/CD déploiements, le AWS Proton pipeline risque de ne pas être créé et le vôtre ne figure peut-être pas dans votre compte. AWS

  Pour plus d'informations, consultez Services et the section called "Mettre à jour le pipeline".

- Composant : extension définie par le développeur pour une instance de service. Spécifie les ressources d' AWS infrastructure supplémentaires dont une application particulière peut avoir besoin, en plus des ressources fournies par l'environnement et l'instance de service. Les équipes de plateforme contrôlent l'infrastructure qu'un composant peut fournir en attribuant un rôle de composant à l'environnement.

  Pour de plus amples informations, veuillez consulter Éléments.

Comment AWS Proton approvisionne l'infrastructure

AWS Proton peut fournir l'infrastructure de plusieurs manières :

- AWS-provisionnement géré : AWS Proton appelle le moteur de provisionnement en votre nom. Cette méthode ne prend en charge que les ensembles de AWS CloudFormation modèles. Pour de plus amples informations, veuillez consulter the section called "CloudFormation fichiers IaC".
- CodeBuild provisionnement : permet AWS Proton AWS CodeBuild d'exécuter les commandes shell que vous fournissez. Vos commandes peuvent lire les entrées qui AWS Proton fournissent et sont responsables du provisionnement ou du déprovisionnement de l'infrastructure et de la génération de valeurs de sortie. Un ensemble de modèles pour cette méthode inclut vos commandes dans un fichier manifeste et tous les programmes, scripts ou autres fichiers dont ces commandes peuvent avoir besoin.

À titre d'exemple d'utilisation du CodeBuild provisionnement, vous pouvez inclure du code qui utilise le AWS Cloud Development Kit (AWS CDK) pour provisionner AWS des ressources, ainsi qu'un manifeste qui installe le CDK et exécute votre code CDK.

Pour de plus amples informations, veuillez consulter the section called “CodeBuild offre groupée”.

> **Note**
>
> Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

- Provisionnement autogéré : AWS Proton envoie une pull request (PR) à un référentiel que vous fournissez, dans lequel votre propre système de déploiement d'infrastructure exécute le processus de provisionnement. Cette méthode ne prend en charge que les ensembles de modèles Terraform. Pour de plus amples informations, veuillez consulter the section called “Fichiers Terraform iAC”.

AWS Proton détermine et définit la méthode de provisionnement pour chaque environnement et service séparément. Lorsque vous créez ou mettez à jour un environnement ou un service, AWS Proton examinez le bundle de modèles que vous fournissez et déterminez la méthode de provisionnement indiquée par le bundle de modèles. Au niveau de l'environnement, vous fournissez les paramètres dont l'environnement et ses services potentiels peuvent avoir besoin pour leurs méthodes de provisionnement : rôles Gestion des identités et des accès AWS (IAM), connexion à un compte d'environnement ou référentiel d'infrastructure.

Les développeurs qui fournissent AWS Proton un service ont la même expérience quelle que soit la méthode de provisionnement. Les développeurs n'ont pas besoin de connaître la méthode de provisionnement et n'ont pas besoin de modifier quoi que ce soit au cours du processus de provisionnement des services. Le modèle de service définit la méthode de provisionnement, et chaque environnement dans lequel un développeur déploie le service fournit les paramètres nécessaires au provisionnement des instances de service.

Le schéma suivant résume les principales caractéristiques des différentes méthodes de provisionnement. Les sections qui suivent le tableau fournissent des détails sur chaque méthode.

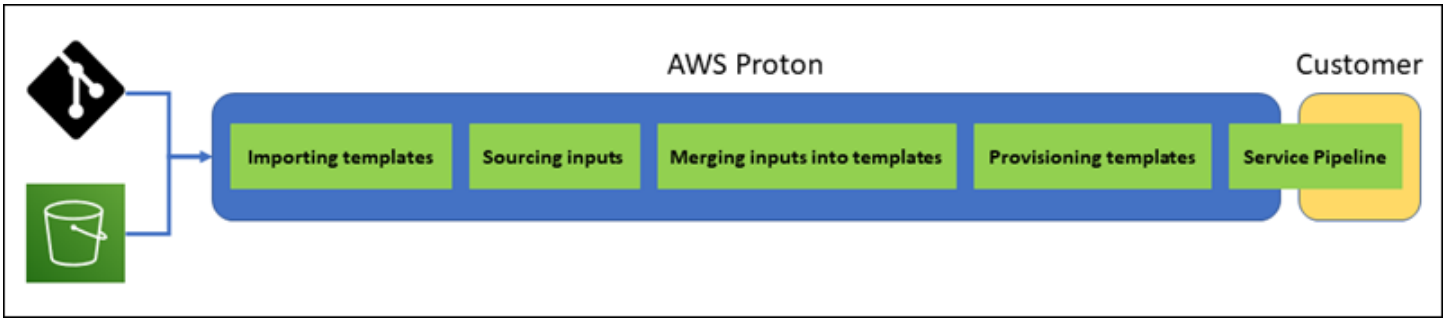
Méthode de provision nement	Modèles	Provisionné par	État suivi par
Géré par AWS	manifeste, schéma, fichier IaC (CloudFor mation)	AWS Proton (à travers CloudFormation)	AWS Proton (à travers CloudFormation)
CodeBuild	manifeste (avec commandes), schéma, dépendances des commandes (par exemple, AWS CDK code)	AWS Proton (à travers CodeBuild)	AWS Proton (vos commandes renvoient le statut CodeBuild)
autogéré	manifeste, schéma, fichiers IaC (Terraform)	Votre code (via des actions Git)	Votre code (transmis AWS via un appel d'API)

Comment fonctionne AWS le provisionnement géré

Lorsqu'un environnement ou un service utilise le provisionnement AWS géré, l'infrastructure est provisionnée comme suit :

1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour plus d'informations, consultez la section suivante,the section called "Considérations relatives au AWS provisionnement géré".
2. AWS Proton affiche un CloudFormation modèle complet pour le provisionnement de la ressource.
3. AWS Proton appels CloudFormation pour démarrer le provisionnement à l'aide du modèle rendu.
4. AWS Proton surveille en permanence le CloudFormation déploiement.
5. Une fois le provisionnement terminé, AWS Proton signale les erreurs en cas d'échec et capture les résultats du provisionnement, tels que l'identifiant Amazon VPC, en cas de succès.

Le schéma suivant montre que AWS Proton la plupart de ces étapes sont effectuées directement.



Considérations relatives au AWS provisionnement géré

- Rôle de provisionnement de l'infrastructure : lorsqu'un environnement ou l'une des instances de service qui y sont exécutées peut utiliser le provisionnement AWS géré, un administrateur doit configurer un rôle IAM (directement ou dans le cadre d'une connexion à un compte d' AWS Proton environnement). AWS Proton utilise ce rôle pour approvisionner l'infrastructure de ces ressources de provisionnement AWS gérées. Le rôle doit disposer des autorisations nécessaires CloudFormation pour créer toutes les ressources incluses dans les modèles de ces ressources.

  Pour plus d'informations, consultez the section called "Rôles IAM" et the section called "Exemples de politiques relatives aux rôles de service".
- Provisionnement de services : lorsqu'un développeur déploie une instance de service qui utilise le provisionnement AWS géré dans l'environnement, AWS Proton utilise le rôle fourni à cet environnement pour provisionner l'infrastructure de l'instance de service. Les développeurs ne voient pas ce rôle et ne peuvent pas le modifier.
- Service avec pipeline — Un modèle de service qui utilise le provisionnement AWS géré peut inclure une définition de pipeline écrite dans le schéma CloudFormation YAML. AWS Proton crée également le pipeline en appelant CloudFormation. Le rôle AWS Proton utilisé pour créer un pipeline est distinct du rôle de chaque environnement individuel. Ce rôle est fourni AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour approvisionner et gérer tous les pipelines AWS gérés. Ce rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

  Les procédures suivantes montrent comment attribuer le rôle de pipeline à AWS Proton.

  AWS Proton console

  Pour fournir le rôle de pipeline

  1. Dans le volet de navigation de la AWS Proton console, sélectionnez Paramètres > Paramètres du compte, puis sélectionnez Configurer.

2. Utilisez la section Rôle AWS géré par le pipeline pour configurer un rôle de pipeline nouveau ou existant pour le provisionnement AWS géré.

AWS Proton API

Pour fournir le rôle de pipeline

1. Utilisez l'action UpdateAccountSettingsAPI.
2. Indiquez le nom de ressource Amazon (ARN) de votre rôle de service de pipeline dans le `pipelineServiceRoleArn` paramètre.

AWS CLI

Pour fournir le rôle de pipeline

Exécutez la commande suivante :

```
$ aws proton update-account-settings \
    --pipeline-service-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

Comment fonctionne CodeBuild le provisionnement

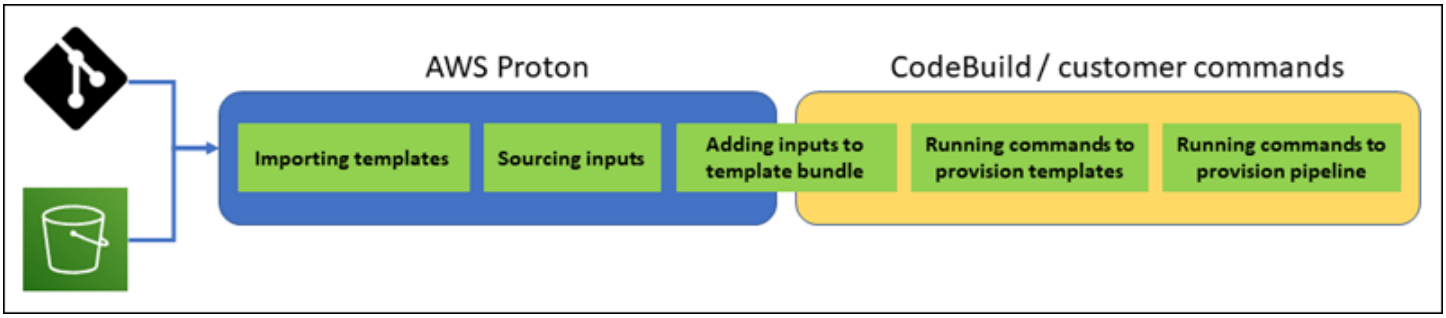
Lorsqu'un environnement ou un service utilise le CodeBuild provisionnement, l'infrastructure est provisionnée comme suit :

1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour plus d'informations, consultez la section suivante,the section called "Considérations relatives au CodeBuild provisionnement".
2. AWS Proton affiche un fichier d'entrée avec les valeurs des paramètres d'entrée pour le provisionnement de la ressource.
3. AWS Proton appels CodeBuild pour démarrer un travail. La CodeBuild tâche exécute les commandes shell du client spécifiées dans le modèle. Ces commandes fournissent l'infrastructure souhaitée, tout en lisant éventuellement les valeurs d'entrée.

4. Lorsque le provisionnement est terminé, la commande finale du client renvoie l'état de provisionnement CodeBuild et appelle l'action d'NotifyResourceDeploymentStatusChange AWS Proton API pour fournir des résultats, tels que l'identifiant Amazon VPC, s'il en existe un.

> **Important**
>
> Assurez-vous que vos commandes renvoient correctement l'état de provisionnement CodeBuild et fournissent les sorties. Dans le cas contraire, ils ne AWS Proton pourront pas suivre correctement l'état du provisionnement et ne pourront pas fournir de résultats corrects aux instances de service.

Le schéma suivant illustre les étapes exécutées et AWS Proton les étapes que vos commandes exécutent dans le cadre d'une CodeBuild tâche.



Considérations relatives au CodeBuild provisionnement

- Rôle de provisionnement de l'infrastructure : lorsqu'un environnement ou l'une des instances de service qui y sont exécutées peut utiliser le provisionnement CodeBuild basé, un administrateur doit configurer un rôle IAM (directement ou dans le cadre d'une connexion à un compte d' AWS Proton environnement). AWS Proton utilise ce rôle pour approvisionner l'infrastructure de ces ressources de CodeBuild provisionnement. Le rôle doit disposer des autorisations nécessaires pour CodeBuild créer toutes les ressources que vous commandez dans les modèles de mise à disposition de ces ressources.

  Pour plus d'informations, consultez the section called "Rôles IAM" et the section called "Exemples de politiques relatives aux rôles de service".
- Provisionnement de services : lorsqu'un développeur déploie une instance de service qui utilise le CodeBuild provisionnement dans l'environnement, AWS Proton utilise le rôle fourni à cet environnement pour provisionner l'infrastructure de l'instance de service. Les développeurs ne voient pas ce rôle et ne peuvent pas le modifier.

- Service avec pipeline : un modèle de service qui utilise le CodeBuild provisionnement peut inclure des commandes pour approvisionner un pipeline. AWS Proton crée également le pipeline en appelant CodeBuild. Le rôle AWS Proton utilisé pour créer un pipeline est distinct du rôle de chaque environnement individuel. Ce rôle est fourni AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour approvisionner et gérer tous les pipelines CodeBuild basés. Ce rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

  Les procédures suivantes montrent comment attribuer le rôle de pipeline à AWS Proton.

  AWS Proton console

  Pour fournir le rôle de pipeline

  1. Dans le volet de navigation de la AWS Proton console, sélectionnez Paramètres > Paramètres du compte, puis sélectionnez Configurer.
  2. Utilisez la section Rôle de provisionnement du pipeline Codebuild pour configurer un rôle de pipeline nouveau ou existant pour CodeBuild le provisionnement.

  AWS Proton API

  Pour fournir le rôle de pipeline

  1. Utilisez l'action UpdateAccountSettingsAPI.
  2. Indiquez le nom de ressource Amazon (ARN) de votre rôle de service de pipeline dans le `pipelineCodebuildRoleArn` paramètre.

  AWS CLI

  Pour fournir le rôle de pipeline

  Exécutez la commande suivante :

```
$ aws proton update-account-settings \
    --pipeline-codebuild-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

Comment fonctionne le provisionnement autogéré

Lorsqu'un environnement est configuré pour utiliser le provisionnement autogéré, l'infrastructure est provisionnée comme suit :

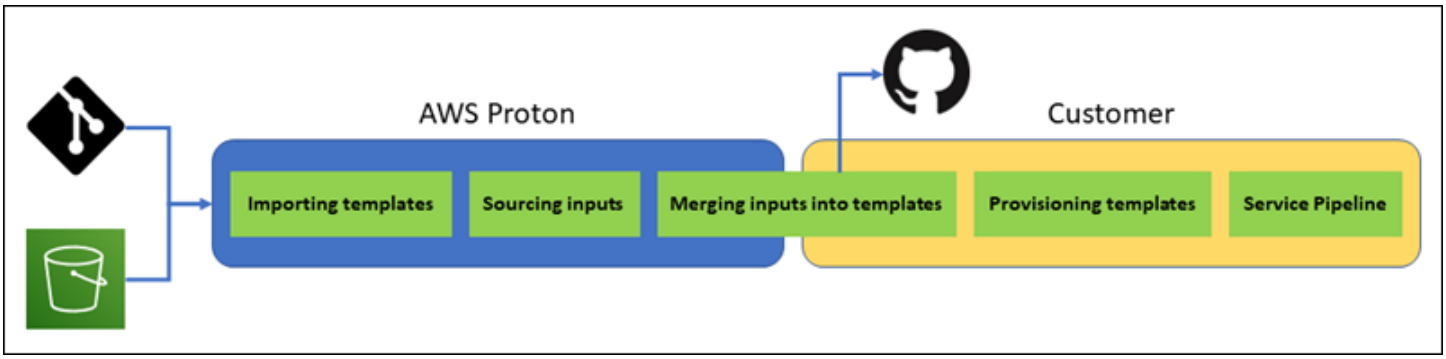
1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour un environnement, le client fournit également un référentiel d'infrastructure lié. Pour plus d'informations, consultez la section suivante,the section called "Considérations relatives au provisionnement autogéré".
2. AWS Proton affiche un modèle Terraform complet. Il se compose d'un ou de plusieurs fichiers Terraform, potentiellement dans plusieurs dossiers, et d'un fichier de `.tfvars` variables. AWS Proton écrit les valeurs des paramètres fournies lors de l'appel de création de ressources dans ce fichier de variables.
3. AWS Proton soumet un PR au référentiel d'infrastructure avec le modèle Terraform rendu.
4. Lorsque le client (administrateur ou développeur) fusionne le PR, l'automatisation du client déclenche le démarrage du provisionnement de l'infrastructure à l'aide du modèle fusionné.

   > **Note**
   >
   > Si le client (administrateur ou développeur) ferme le PR, AWS Proton reconnaît le PR comme étant fermé et marque le déploiement comme annulé.

5. Lorsque le provisionnement est terminé, l'automatisation du client appelle l'action de l'NotifyResourceDeploymentStatusChange AWS Proton API pour indiquer l'achèvement, fournir le statut (réussite ou échec) et fournir des résultats, tels que l'identifiant Amazon VPC, le cas échéant.

   > **Important**
   >
   > Assurez-vous que votre code d'automatisation indique l'état et AWS Proton les sorties de provisionnement. Si ce n'est pas le cas, vous AWS Proton pouvez considérer le provisionnement comme en attente plus longtemps qu'il ne le devrait et continuer à afficher le statut En cours.

Le schéma suivant illustre les étapes exécutées AWS Proton et les étapes exécutées par votre propre système de provisionnement.



Considérations relatives au provisionnement autogéré

- Référentiel d'infrastructure : lorsqu'un administrateur configure un environnement pour un provisionnement autogéré, il doit fournir un référentiel d'infrastructure lié. AWS Proton PRs se soumet à ce référentiel pour approvisionner l'infrastructure de l'environnement et toutes les instances de service qui y sont déployées. L'action d'automatisation appartenant au client dans le référentiel doit assumer un rôle IAM avec les autorisations nécessaires pour créer toutes les ressources incluses dans votre environnement et vos modèles de services, ainsi qu'une identité qui reflète le compte de destination. AWS Pour un exemple GitHub d'action qui assume un rôle, voir Assumer un rôle dans la documentation « Configurer les AWS informations d'identification » Action for GitHub Actions.
- Autorisations : votre code d'approvisionnement doit s'authentifier auprès d'un compte si nécessaire (par exemple, s'authentifier auprès d'un AWS compte) et fournir une autorisation de mise en service des ressources (par exemple, fournir un rôle).
- Provisionnement de services : lorsqu'un développeur déploie une instance de service qui utilise le provisionnement autogéré dans l'environnement, AWS Proton soumet un PR au référentiel associé à l'environnement afin de fournir l'infrastructure de l'instance de service. Les développeurs ne voient pas le dépôt et ne peuvent pas le modifier.

> **Note**
>
> Les développeurs qui créent des services utilisent le même processus, quelle que soit la méthode de provisionnement, et la différence en est déduite. Cependant, avec l'autogestion du provisionnement, les développeurs peuvent avoir une réponse plus lente, car ils doivent attendre que quelqu'un (qui n'est peut-être pas eux-mêmes) fusionne le PR dans le référentiel d'infrastructure avant que le provisionnement puisse commencer.

- Service avec pipeline — Un modèle de service pour un environnement avec un provisionnement autogéré peut inclure une définition de pipeline (par exemple, un AWS CodePipeline pipeline), écrite en Terraform HCL. Pour AWS Proton permettre le provisionnement de ces pipelines, un administrateur fournit un référentiel de pipelines lié à AWS Proton. Lors du provisionnement d'un pipeline, l'action d'automatisation appartenant au client dans le référentiel doit assumer un rôle IAM avec les autorisations nécessaires pour approvisionner le pipeline et une identité qui reflète le compte de destination. AWS Le référentiel et le rôle du pipeline sont distincts de ceux utilisés pour chaque environnement individuel. Le référentiel lié est fourni AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour approvisionner et gérer tous les pipelines. Le rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

  Les procédures suivantes montrent comment fournir le référentiel et le rôle du pipeline à AWS Proton.

  AWS Proton console

  Pour fournir le rôle de pipeline

  1. Dans le volet de navigation de la AWS Proton console, sélectionnez Paramètres > Paramètres du compte, puis sélectionnez Configurer.
  2. Utilisez la section du référentiel du pipeline CI/CD pour configurer un lien de référentiel nouveau ou existant.

  AWS Proton API

  Pour fournir le rôle de pipeline

  1. Utilisez l'action UpdateAccountSettingsAPI.
  2. Indiquez le fournisseur, le nom et la branche de votre référentiel de pipeline dans le `pipelineProvisioningRepository` paramètre.

  AWS CLI

  Pour fournir le rôle de pipeline

  Exécutez la commande suivante :

  ```
  $ aws proton update-account-settings \
      --pipeline-provisioning-repository \
  ```

```
"provider=GITHUB,name=my-pipeline-repo-name,branch=my-branch"
```

- Suppression des ressources provisionnées autogérées — Les modules Terraform peuvent inclure des éléments de configuration nécessaires au fonctionnement de Terraform, en plus des définitions de ressources. Par conséquent, il n'est pas AWS Proton possible de supprimer tous les fichiers Terraform d'une instance d'environnement ou de service. AWS Proton Marquez plutôt les fichiers à supprimer et mettez à jour un indicateur dans les métadonnées PR. Votre automatisation peut lire cet indicateur et l'utiliser pour déclencher une commande de destruction Terraform.

AWS Proton terminologie

Modèle d'environnement

Définit l'infrastructure partagée, telle qu'un VPC ou un cluster, utilisée par plusieurs applications ou ressources.

Ensemble de modèles d'environnement

Ensemble de fichiers que vous chargez pour créer et enregistrer un modèle d'environnement AWS Proton. Un ensemble de modèles d'environnement contient les éléments suivants :

1. Fichier de schéma qui définit l'infrastructure en tant que paramètres d'entrée de code.
2. Fichier d'infrastructure sous forme de code (IaC) qui définit une infrastructure partagée, telle qu'un VPC ou un cluster, utilisée par plusieurs applications ou ressources.
3. Un fichier manifeste qui répertorie le fichier IaC.

Environnement

Infrastructure partagée provisionnée, telle qu'un VPC ou un cluster, utilisée par plusieurs applications ou ressources.

Modèle de service

Définit le type d'infrastructure nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.

Ensemble de modèles de services

Ensemble de fichiers que vous chargez pour créer et enregistrer un modèle de service AWS Proton. Un ensemble de modèles de services contient les éléments suivants :

1. Fichier de schéma qui définit l'infrastructure en tant que paramètres d'entrée de code (IaC).

2. Fichier IaC qui définit l'infrastructure nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.
3. Un fichier manifeste qui répertorie le fichier IaC.
4. Facultatif
   a. Fichier IaC qui définit l'infrastructure du pipeline de services.
   b. Un fichier manifeste qui répertorie le fichier IaC.

Service

Infrastructure provisionnée nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.

Instance de service

Infrastructure provisionnée qui prend en charge une application ou un microservice dans un environnement.

Pipeline de services

Infrastructure provisionnée qui prend en charge un pipeline.

Version du modèle

Version majeure ou mineure d'un modèle. Pour de plus amples informations, veuillez consulter Modèles versionnés.

Paramètres d’entrée

Défini dans un fichier de schéma et utilisé dans un fichier d'infrastructure en tant que code (IaC) afin que le fichier IaC puisse être utilisé de manière répétitive et pour divers cas d'utilisation.

Fichier de schéma

Définit l'infrastructure en tant que paramètres d'entrée du fichier de code.

Fichier de spécifications

Spécifie les valeurs de l'infrastructure sous forme de paramètres d'entrée de fichier de code, tels que définis dans un fichier de schéma.

Fichier manifeste

Répertorie une infrastructure sous forme de fichier de code.

Création de modèles et création de packs pour AWS Proton

AWS Proton fournit des ressources pour vous sur la base de fichiers d'infrastructure sous forme de code (IaC). Vous décrivez l'infrastructure dans des fichiers IaC réutilisables. Pour que les fichiers soient réutilisables pour différents environnements et applications, vous les créez sous forme de modèles, vous définissez les paramètres d'entrée et vous utilisez ces paramètres dans les définitions IaC. Lorsque vous créez ultérieurement une ressource de provisionnement (environnement, instance de service ou composant), elle AWS Proton utilise un moteur de rendu qui combine les valeurs d'entrée avec un modèle pour créer un fichier IaC prêt à être provisionné.

Les administrateurs créent la plupart des modèles sous forme de lots de modèles, puis les téléchargent et les enregistrent dans AWS Proton. Le reste de cette page traite de ces ensembles AWS Proton de modèles. Les composants directement définis constituent une exception : les développeurs les créent et fournissent directement des fichiers modèles IaC. Pour plus d'informations sur les composants, consultezÉléments.

Rubriques

Packs de modèles

En tant qu'administrateur, vous créez et enregistrez des modèles auprès de AWS Proton. Vous utilisez ces modèles pour créer des environnements et des services. Lorsque vous créez un service, AWS Proton provisionnez et déployez des instances de service dans des environnements sélectionnés. Pour de plus amples informations, veuillez consulter AWS Proton pour les équipes de la plateforme.

Pour créer et enregistrer un modèle AWS Proton, vous devez télécharger un ensemble de modèles contenant les fichiers d'infrastructure sous forme de code (IaC) AWS Proton nécessaires à la fourniture d'un environnement ou d'un service.

Un ensemble de modèles contient les éléments suivants :

- Un fichier d'infrastructure en tant que code (IaC) avec un fichier manifeste YAML répertoriant le fichier IaC.
- Un fichier de schéma YAML pour les définitions des paramètres d'entrée de votre fichier IaC.

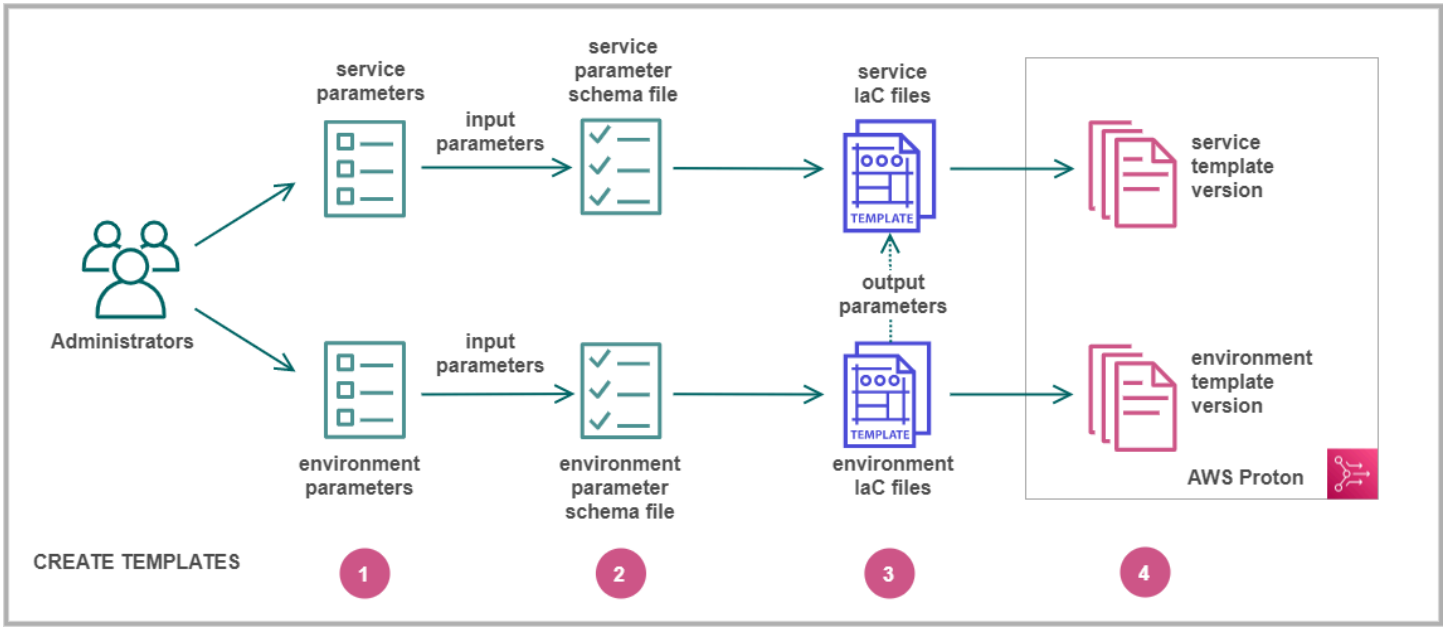
Un ensemble de modèles d' CloudFormation environnement contient un fichier IaC.

Un ensemble CloudFormation de modèles de services contient un fichier IaC pour les définitions d'instances de service et un autre fichier IaC facultatif pour une définition de pipeline.

Les ensembles de modèles d'environnement et de service Terraform peuvent chacun contenir plusieurs fichiers IaC.

AWS Proton nécessite un fichier de schéma de paramètres d'entrée. Lorsque vous créez vos fichiers IaC, vous utilisez la syntaxe Jinja pour référencer vos paramètres d'entrée. AWS CloudFormation AWS Proton fournit des espaces de noms de paramètres que vous pouvez utiliser pour référencer les paramètres de vos fichiers IaC.

Le schéma suivant montre un exemple d'étapes que vous pouvez suivre pour créer un modèle AWS Proton.



1 Identifi

les paramètres d'entrée.

2 Créez un fichier de schéma pour définir vos paramètres d'entrée.

3 Créez des fichiers IaC qui font référence à vos paramètres d'entrée. Vous pouvez référencer les sorties des fichiers iAc de l'environnement en tant qu'entrées pour les fichiers iAc de votre service.

4 Enregis une version de modèle auprès de votre bundle de modèles AWS Proton et téléchargez-le.

AWS Proton paramètres

Vous pouvez définir et utiliser les paramètres de votre infrastructure sous forme de fichiers de code (IaC) pour les rendre flexibles et réutilisables. Vous pouvez lire la valeur d'un paramètre dans vos fichiers IaC en vous référant au nom du paramètre dans l'espace de noms des AWS Proton paramètres. AWS Proton injecte des valeurs de paramètres dans les fichiers iAc rendus qu'il génère lors du provisionnement des ressources. Pour traiter les AWS CloudFormation paramètres IaC, AWS Proton utilise Jinja. Pour traiter les paramètres Terraform IaC, AWS Proton génère un fichier de valeurs de paramètres Terraform et s'appuie sur la capacité de paramétrage intégrée à HCL.

AvecCodeBuild approvisionnement, AWS Proton génère un fichier d'entrée que votre code peut importer. Le fichier est un fichier JSON ou HCL, selon une propriété du manifeste de votre modèle. Pour de plus amples informations, veuillez consulter the section called “CodeBuild paramètres de provisionnement”.

Vous pouvez faire référence aux paramètres de votre environnement, de vos services et de vos composants, aux fichiers iAc ou au code de provisionnement en respectant les exigences suivantes :

- La longueur de chaque nom de paramètre ne dépasse pas 100 caractères.
- La longueur de l'espace de noms des paramètres et du nom de ressource combinés ne dépasse pas la limite de caractères du nom de la ressource.

AWS Proton le provisionnement échoue si ces quotas sont dépassés.

Types de paramètres

Les types de paramètres suivants sont à votre disposition à titre de référence dans les fichiers AWS Proton IaC :

Paramètre d'entrée

Les environnements et les instances de service peuvent prendre des paramètres d'entrée que vous définissez dans un fichier de schéma que vous associez à l'environnement ou au modèle de service. Vous pouvez vous référer aux paramètres d'entrée d'une ressource dans le fichier IaC de la ressource. Les fichiers iAc du composant peuvent faire référence aux paramètres d'entrée de l'instance de service à laquelle le composant est attaché.

AWS Proton vérifie les noms des paramètres d'entrée par rapport à votre fichier de schéma et les associe aux paramètres référencés dans vos fichiers iAc pour injecter les valeurs d'entrée que vous fournissez dans un fichier de spécifications lors du provisionnement des ressources.

Paramètre de sortie

Vous pouvez définir des sorties dans n'importe lequel de vos fichiers iAc. Une sortie peut être, par exemple, le nom, l'ID ou l'ARN de l'une des ressources fournies par le modèle, ou elle peut être un moyen de passer par l'une des entrées du modèle. Vous pouvez faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Dans les CloudFormation fichiers IaC, définissez les paramètres de sortie dans le `Outputs:` bloc. Dans un fichier Terraform iAC, définissez chaque paramètre de sortie à l'aide d'une `output` instruction.

Paramètre de ressource

AWS Proton crée automatiquement les paramètres AWS Proton des ressources. Ces paramètres exposent les propriétés de l'objet de AWS Proton ressource. Voici un exemple de paramètre de ressource`environment.name`.

Utilisation de AWS Proton paramètres dans vos fichiers iAc

Pour lire la valeur d'un paramètre dans un fichier IaC, vous devez vous référer au nom du paramètre dans l'espace de noms des AWS Proton paramètres. Pour les fichiers AWS CloudFormation iAC, vous devez utiliser la syntaxe Jinja et entourer le paramètre de paires d'accolades et de guillemets.

Le tableau suivant indique la syntaxe de référence pour chaque langage de modèle pris en charge, avec un exemple.

Langue du modèle	Syntaxe	Exemple : entrée d'environnement nommée « VPC »
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
Terraform	`var.parameter-name`	`var.environment.inputs.VPC` Définitions de variables Terraform générées

> **Note**
>
> Si vous utilisez des paramètres CloudFormation dynamiques dans votre fichier IaC, vous devez y échapper pour éviter les erreurs d'interprétation par Jinja. Pour de plus amples informations, consultez Résolution des problèmes AWS Proton.

Le tableau suivant répertorie les noms d'espaces de noms pour tous les paramètres de AWS Proton ressources. Chaque type de fichier modèle peut utiliser un sous-ensemble différent de l'espace de noms des paramètres.

Fichier modèle	Type de paramètre	Nom du paramètre	Description
Environnement	ressource	`environment.` **`name`**	Nom de l'environnement
	input	`environment.inputs.` *`input-name`*	Entrées d'environnement définies par le schéma
Service	ressource	`environment.` **`name`** `environment.` **`account_id`**	Nom et Compte AWS ID de l'environnement

Fichier modèle	Type de paramètre	Nom du paramètre	Description
	output	`environment.outputs.` *`output-name`*	Sorties de fichiers iAC de l'environnement
	ressource	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nom du service et référentiel de code
	ressource	`service_instance.` **`name`**	Nom de l'instance de service
	input	`service_instance.inputs.` *`input-name`*	Entrées d'instance de service définies par le schéma
	ressource	`service_instance.components.default.` **`name`**	Nom du composant par défaut joint
	output	`service_instance.components.default.outputs.` *`output-name`*	Sorties du fichier iAC du composant par défaut joint
Pipeline	ressource	`service_instance.environment.` **`name`** `service_instance.environment.` **`account_id`**	Nom et Compte AWS ID de l'environnement de l'instance de service
	output	`service_instance.environment.outputs.` *`output-name`*	Sorties de fichiers iAC de l'environnement d'instance de service

Fichier modèle	Type de paramètre	Nom du paramètre	Description
Composant	input	`pipeline.inputs.` *`input-name`*	Entrées de pipeline définies par le schéma
	ressource	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nom du service et référentiel de code
	input	`service_instance.inputs.` *`input-name`*	Entrées d'instance de service définies par le schéma
	collection	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	Ensemble d'instances de service que vous pouvez parcourir en boucle
	ressource	`environment.` **`name`** `environment.` **`account_id`**	Nom de l'environnement et ID de Compte AWS compte
	output	`environment.outputs.` *`output-name`*	Sorties de fichiers iAC de l'environnement
	ressource	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nom du service et référentiel de code (composants attachés)

Fichier modèle	Type de paramètre	Nom du paramètre	Description
	ressource	`service_instance.` **`name`**	Nom de l'instance de service (composants attachés)
	input	`service_instance.inputs.` *`input-name`*	Entrées d'instance de service définies par le schéma (composants attachés)
	ressource	`component.` **`name`**	Nom du composant

Pour plus d'informations et des exemples, consultez les sous-rubriques relatives aux paramètres des fichiers modèles IaC pour différents types de ressources et différents langages de modèles.

Rubriques

- Détails et exemples des paramètres du fichier CloudFormation iAC d'environnement
- Détails et exemples des paramètres du fichier CloudFormation IaC du service
- Détails et exemples des paramètres du fichier CloudFormation iAC du composant
- Filtres de paramètres pour les fichiers CloudFormation IaC
- CodeBuild détails et exemples des paramètres de provisionnement
- Détails et exemples des paramètres du fichier d'infrastructure en tant que code (IaC) Terraform

Détails et exemples des paramètres du fichier CloudFormation iAC d'environnement

Vous pouvez définir et référencer les paramètres de votre infrastructure d'environnement sous forme de fichiers de code (IaC). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultezthe section called “Parameters”.

Définir les paramètres de l'environnement

Vous pouvez définir des paramètres d'entrée et de sortie pour les fichiers iAc d'environnement.

- Paramètres d'entrée : définissez les paramètres d'entrée de l'environnement dans votre fichier de schéma.

  La liste suivante inclut des exemples de paramètres d'entrée d'environnement pour des cas d'utilisation typiques.

  - Valeurs CIDR VPC
  - Paramètres de l'équilibreur de charge
  - Paramètres de base de données
  - Un délai d'expiration pour le bilan de santé

  En tant qu'administrateur, vous pouvez fournir des valeurs pour les paramètres d'entrée lorsque vous créez un environnement :

  - Utilisez la console pour remplir un formulaire basé sur un schéma qui AWS Proton fournit.
  - Utilisez la CLI pour fournir une spécification qui inclut les valeurs.
- Paramètres de sortie — Définissez les sorties d'environnement dans les fichiers iAc de votre environnement. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Lire les valeurs des paramètres dans les fichiers iAC de l'environnement

Vous pouvez lire les paramètres relatifs à l'environnement dans les fichiers iAc de l'environnement. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'un environnement en la référençantenvironment.inputs.*input-name*.
- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels queenvironment.name.

> **Note**
>
> Aucun paramètre de sortie d'autres ressources n'est disponible pour les fichiers iAc de l'environnement.

Exemples de fichiers IaC d'environnement et de service avec paramètres

L'exemple suivant illustre la définition et la référence de paramètres dans un fichier IaC d'environnement. L'exemple montre ensuite comment les paramètres de sortie d'environnement définis dans le fichier IaC d'environnement peuvent être référencés dans un fichier IaC de service.

Example Fichier CloudFormation iAc d'environnement

Notez ce qui suit dans cet exemple :

- L'espace de `environment.inputs.` noms fait référence aux paramètres d'entrée de l'environnement.
- Le `StoreInputValue` paramètre Amazon EC2 Systems Manager (SSM) concatène les entrées de l'environnement.
- La `MyEnvParameterValue` sortie expose la même concaténation de paramètres d'entrée qu'un paramètre de sortie. Trois paramètres de sortie supplémentaires exposent également les paramètres d'entrée individuellement.
- Six paramètres de sortie supplémentaires exposent les ressources fournies par l'environnement.

```
Resources:
  StoreInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ environment.inputs.my_sample_input }}
 {{ environment.inputs.my_other_sample_input}}
 {{ environment.inputs.another_optional_input }}"
              # input parameter references

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'environment.outputs' namespace, for example,
 service_instance.environment.outputs.ClusterName.
```

```
Outputs:
  MyEnvParameterValue:                                    # output definition
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                     # output definition
    Value: "{{ environment.inputs.my_sample_input }}"         #   input parameter
 reference
  MyOtherSampleInputValue:                                # output definition
    Value: "{{ environment.inputs.my_other_sample_input }}"   #   input parameter
 reference
  AnotherOptionalInputValue:                              # output definition
    Value: "{{ environment.inputs.another_optional_input }}"  #   input parameter
 reference
  ClusterName:                                            # output definition
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'                              #   provisioned resource
  ECSTaskExecutionRole:                                   # output definition
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'             #   provisioned resource
  VpcId:                                                  # output definition
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'                                     #   provisioned resource
  PublicSubnetOne:                                        # output definition
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'                         #   provisioned resource
  PublicSubnetTwo:                                        # output definition
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'                         #   provisioned resource
  ContainerSecurityGroup:                                 # output definition
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'                  #   provisioned resource
```

Example Fichier CloudFormation IaC du service

L'espace de `environment.outputs.` noms fait référence aux sorties d'environnement d'un fichier IaC d'environnement. Par exemple, le nom `environment.outputs.ClusterName` indique la valeur du paramètre de sortie de `ClusterName` l'environnement.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
```

```
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}' # resource parameter
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
 parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}'  # output
 reference to an environment infrastructure code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'  # resource parameter
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
```

```
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}' # resource parameter
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}' # resource parameter

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'  # resource parameter
      Cluster: '{{environment.outputs.ClusterName}}' # output reference to an
 environment infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'# input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output reference to an
 environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}' # output reference to an
 environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}' # output reference to an
 environment infrastructure as code file
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'  # resource parameter
          ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          TargetGroupArn: !Ref 'TargetGroup'
[...]
```

Détails et exemples des paramètres du fichier CloudFormation IaC du service

Vous pouvez définir et référencer les paramètres de votre infrastructure de service et de pipeline sous forme de fichiers de code (iAc). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultezthe section called "Parameters".

Définir les paramètres du service

Vous pouvez définir des paramètres d'entrée et de sortie pour les fichiers IaC du service.

- Paramètres d'entrée : définissez les paramètres d'entrée de l'instance de service dans votre fichier de schéma.

  La liste suivante inclut des exemples de paramètres d'entrée de service pour des cas d'utilisation typiques.

  - Port
  - Taille de la tâche
  - Image
  - Nombre souhaité
  - fichier Docker
  - Commande de test unitaire

  Vous fournissez des valeurs pour les paramètres d'entrée lorsque vous créez un service :

  - Utilisez la console pour remplir un formulaire basé sur un schéma qui AWS Proton fournit.
  - Utilisez la CLI pour fournir une spécification qui inclut les valeurs.
- Paramètres de sortie — Définissez les sorties des instances de service dans vos fichiers IaC de service. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Lire les valeurs des paramètres dans les fichiers IaC du service

Vous pouvez lire les paramètres relatifs au service et aux autres ressources dans les fichiers IaC du service. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'une instance de service en la référençantservice_instance.inputs.*input-name*.
- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels que service.nameservice_instance.name, etenvironment.name.
- Paramètres de sortie — Lisez les sorties d'autres ressources en faisant référence à environment.outputs.*output-name* ouservice_instance.components.default.outputs.*output-name*.

Exemple de fichier IaC de service avec paramètres

L'exemple suivant est un extrait d'un fichier IaC de service CloudFormation . L'espace de environment.outputs. noms fait référence aux sorties du fichier IaC de l'environnement. L'espace de service_instance.inputs. noms fait référence aux paramètres d'entrée de l'instance de service. La service_instance.name propriété fait référence à un paramètre de AWS Proton ressource.

```
Resources:
  StoreServiceInstanceInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ service.name }} {{ service_instance.name }}
 {{ service_instance.inputs.my_sample_service_instance_required_input }}
 {{ service_instance.inputs.my_sample_service_instance_optional_input }}
 {{ environment.outputs.MySampleInputValue }}
 {{ environment.outputs.MyOtherSampleInputValue }}"
              #  resource parameter references               # input parameter
 references
                                          # output references to an environment
 infrastructure as code file
Outputs:
  MyServiceInstanceParameter:                                                  #
 output definition
    Value: !Ref StoreServiceInstanceInputValue
  MyServiceInstanceRequiredInputValue:                                         #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_required_input }}"  #
 input parameter reference
  MyServiceInstanceOptionalInputValue:                                         #
 output definition
```

```
    Value: "{{ service_instance.inputs.my_sample_service_instance_optional_input }}"  #
input parameter reference
 MyServiceInstancesEnvironmentSampleOutputValue:                                      #
output definition
    Value: "{{ environment.outputs.MySampleInputValue }}"                             #
output reference to an environment IaC file
 MyServiceInstancesEnvironmentOtherSampleOutputValue:                                 #
output definition
    Value: "{{ environment.outputs.MyOtherSampleInputValue }}"                        #
output reference to an environment IaC file
```

Détails et exemples des paramètres du fichier CloudFormation iAC du composant

Vous pouvez définir et référencer les paramètres de votre infrastructure de composants sous forme de fichiers de code (IaC). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultezthe section called “Parameters”. Pour plus d'informations sur les composants, consultezÉléments.

Définir les paramètres de sortie des composants

Vous pouvez définir les paramètres de sortie dans les fichiers iAc de vos composants. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC du service.

> **Note**
>
> Vous ne pouvez pas définir d'entrées pour les fichiers iAc des composants. Les composants attachés peuvent obtenir des entrées de l'instance de service à laquelle ils sont attachés. Les composants détachés n'ont pas d'entrées.

Lire les valeurs des paramètres dans les fichiers iAc des composants

Vous pouvez lire les paramètres relatifs au composant et à d'autres ressources dans les fichiers iAc du composant. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'une instance de service attachée en la référençant`service_instance.inputs.`*`input-name`*.

- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels que `component.name`ou `service_instance.name`, et`environment.name`.
- Paramètres de sortie — Lisez les sorties de l'environnement en les référençant`environment.outputs.`*`output-name`*.

Exemples de fichiers IaC de composants et de services avec paramètres

L'exemple suivant montre un composant qui approvisionne un bucket Amazon Simple Storage Service (Amazon S3) et la politique d'accès associée, et qui expose les Amazon Resource ARNs Names () des deux ressources sous forme de sorties de composants. Un modèle de service IaC ajoute les sorties des composants en tant que variables d'environnement de conteneur d'une tâche Amazon Elastic Container Service (Amazon ECS) afin de rendre les sorties disponibles pour le code exécuté dans le conteneur, et ajoute la politique d'accès au bucket au rôle de la tâche. Le nom du compartiment est basé sur les noms de l'environnement, du service, de l'instance de service et du composant, ce qui signifie que le compartiment est couplé à une instance spécifique du modèle de composant étendant une instance de service spécifique. Les développeurs peuvent créer plusieurs composants personnalisés sur la base de ce modèle de composants, afin de fournir des compartiments Amazon S3 pour différentes instances de service et différents besoins fonctionnels.

L'exemple montre comment vous utilisez la `{{ ... }}` syntaxe Jinja pour faire référence aux paramètres des composants et autres ressources dans votre fichier IaC de service. Vous pouvez utiliser `{% if ... %}` des instructions pour ajouter des blocs d'instructions uniquement lorsqu'un composant est attaché à l'instance de service. Les `proton_cfn_*` mots clés sont des filtres que vous pouvez utiliser pour nettoyer et formater les valeurs des paramètres de sortie. Pour plus d'informations sur les filtres, consultez the section called “CloudFormation filtres de paramètres”.

En tant qu'administrateur, vous créez le fichier modèle du service iAC.

Example fichier CloudFormation IaC de service utilisant un composant

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
```

```
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

En tant que développeur, vous créez le fichier modèle du composant iAc.

Example fichier CloudFormation iAC du composant

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
```

```
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

Lors du AWS Proton rendu d'un CloudFormation modèle pour votre instance de service et du remplacement de tous les paramètres par des valeurs réelles, le modèle peut ressembler au fichier suivant.

Example fichier IaC CloudFormation rendu par une instance de service

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          Environment:
            - Name: BucketName
              Value: arn:aws:s3:us-
east-1:123456789012:environment_name-service_name-service_instance_name-component_name
            - Name: BucketAccessPolicyArn
              Value: arn:aws:iam::123456789012:policy/cfn-generated-policy-name
  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name
```

```
  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Filtres de paramètres pour les fichiers CloudFormation IaC

Lorsque vous faites référence à des AWS Proton paramètres dans vos fichiers AWS CloudFormation iAc, vous pouvez utiliser des modificateurs Jinja appelés filtres pour valider, filtrer et formater les valeurs des paramètres avant qu'elles ne soient insérées dans le modèle de rendu. Les validations de filtres sont particulièrement utiles lorsqu'il s'agit de faire référence aux paramètres de sortie des composants, car la création et l'attachement des composants sont effectués par les développeurs, et un administrateur utilisant les sorties des composants dans un modèle d'instance de service peut souhaiter vérifier leur existence et leur validité. Cependant, vous pouvez utiliser des filtres dans n'importe quel fichier Jinja iAc.

Les sections suivantes décrivent et définissent les filtres de paramètres disponibles, et fournissent des exemples. AWS Proton définit la plupart de ces filtres. Le `default` filtre est un filtre intégré à Jinja.

Formater les propriétés de l'environnement pour les tâches Amazon ECS

Déclaration

```
dict # proton_cfn_ecs_task_definition_formatted_env_vars (raw: boolean = True) # YAML
 list of dicts
```

Description

Ce filtre met en forme une liste de sorties à utiliser dans une propriété d'environnement dans la `ContainerDefinition` section d'une définition de tâche Amazon Elastic Container Service (Amazon ECS).

Définissez `raw` sur `False` pour valider également la valeur du paramètre. Dans ce cas, la valeur doit correspondre à l'expression régulière`^[a-zA-Z0-9_-]*$`. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Et le modèle de service suivant :

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            {{ service_instance.components.default.outputs
              | proton_cfn_ecs_task_definition_formatted_env_vars }}
```

Le modèle de service rendu est le suivant :

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            - Name: Output1
              Value: hello
            - Name: Output2
              Value: world
```

Propriétés de l'environnement de formatage pour les fonctions Lambda

Déclaration

```
dict # proton_cfn_lambda_function_formatted_env_vars (raw: boolean = True) # YAML dict
```

Description

Ce filtre met en forme une liste de sorties à utiliser dans une propriété d'environnement dans la `Properties` section de définition d'une AWS Lambda fonction.

Définissez `raw` sur `False` pour valider également la valeur du paramètre. Dans ce cas, la valeur doit correspondre à l'expression régulière`^[a-zA-Z0-9_-]*$`. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Et le modèle de service suivant :

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          {{ service_instance.components.default.outputs
            | proton_cfn_lambda_function_formatted_env_vars }}
```

Le modèle de service rendu est le suivant :

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          Output1: hello
          Output2: world
```

Extraire la politique IAM ARNs à inclure dans les rôles IAM

Déclaration

```
dict # proton_cfn_iam_policy_arns # YAML list
```

Description

Ce filtre met en forme une liste de sorties à utiliser dans une ManagedPolicyArns propriété dans la `Properties` section de définition d'un rôle Gestion des identités et des accès AWS (IAM). Le filtre utilise l'expression régulière `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` pour extraire la politique IAM valide ARNs de la liste des paramètres de sortie. Vous pouvez utiliser ce filtre pour ajouter des politiques dans les valeurs des paramètres de sortie à une définition de rôle IAM dans un modèle de service.

Exemple

Avec le modèle de composant personnalisé suivant :

```
Resources:
  # ...
  ExamplePolicy1:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
  ExamplePolicy2:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...

  # ...
```

```
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
  PolicyArn1:
    Description: "ARN of policy 1"
    Value: !Ref ExamplePolicy1
  PolicyArn2:
    Description: "ARN of policy 2"
    Value: !Ref ExamplePolicy2
```

Et le modèle de service suivant :

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Le modèle de service rendu est le suivant :

```
Resources:

  # ...

  TaskRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-1
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-2

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Nettoyez les valeurs des propriétés

Déclaration

```
string # proton_cfn_sanitize # string
```

Description

Il s'agit d'un filtre à usage général. Utilisez-le pour valider la sécurité d'une valeur de paramètre. Le filtre vérifie que la valeur correspond à l'expression régulière `^[a-zA-Z0-9_-]*$` ou qu'il s'agit d'un Amazon Resource Name (ARN) valide. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example of valid output"
    Value: "This-is_valid_37"
  Output2:
    Description: "Example incorrect output"
    Value: "this::is::incorrect"
  SomeArn:
```

```
    Description: "Example ARN"
    Value: arn:aws:some-service::123456789012:some-resource/resource-name
```

- La référence suivante dans un modèle de service :

```
# ...
  {{ service_instance.components.default.outputs.Output1
    | proton_cfn_sanitize }}
```

  S'affiche comme suit :

```
# ...
  This-is_valid_37
```

- La référence suivante dans un modèle de service :

```
# ...
  {{ service_instance.components.default.outputs.Output2
    | proton_cfn_sanitize }}
```

  Résultats avec l'erreur de rendu suivante :

```
Illegal character(s) detected in "this::is::incorrect". Must match regex ^[a-zA-
Z0-9_-]*$ or be a valid ARN
```

- La référence suivante dans un modèle de service :

```
# ...
  {{ service_instance.components.default.outputs.SomeArn
    | proton_cfn_sanitize }}
```

  S'affiche comme suit :

```
# ...
  arn:aws:some-service::123456789012:some-resource/resource-name
```

Fournir des valeurs par défaut pour les références inexistantes

Description

Le `default` filtre fournit une valeur par défaut lorsqu'il n'existe aucune référence à un espace de noms. Utilisez-le pour écrire des modèles robustes qui peuvent être rendus sans échec même lorsque le paramètre auquel vous faites référence est manquant.

Exemple

La référence suivante dans un modèle de service entraîne l'échec du rendu du modèle si l'instance de service n'a pas de composant directement défini (par défaut) attaché, ou si le composant attaché n'a pas de sortie nomméetest.

```
# ...
  {{ service_instance.components.default.outputs.test }}
```

Pour éviter ce problème, ajoutez le `default` filtre.

```
# ...
  {{ service_instance.components.default.outputs.test | default("[optional-value]") }}
```

CodeBuild détails et exemples des paramètres de provisionnement

Vous pouvez définir des paramètres dans vos modèles pour les AWS Proton ressources CodeBuild basées et référencer ces paramètres dans votre code de provisionnement. Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultezthe section called "Parameters".

> **Note**
>
> Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

Paramètres d'entrée

Lorsque vous créez une AWS Proton ressource, comme un environnement ou un service, vous fournissez des valeurs pour les paramètres d'entrée qui sont définis dans le fichier de schéma de votre modèle. Lorsque la ressource que vous créez utiliseCodeBuild approvisionnement, AWS Proton affiche ces valeurs d'entrée dans un fichier d'entrée. Votre code d'approvisionnement peut importer et obtenir des valeurs de paramètres à partir de ce fichier.

Pour un exemple de CodeBuild modèle, voirthe section called "CodeBuild offre groupée". Pour plus d'informations sur les fichiers manifeste, consultez the section called "Manifeste et résumé".

L'exemple suivant est un fichier d'entrée JSON généré lors du provisionnement CodeBuild basé sur une instance de service.

Exemple : utilisation du AWS CDK with CodeBuild provisioning

```
{
  "service_instance": {
    "name": "my-service-staging",
    "inputs": {
      "port": "8080",
      "task_size": "medium"
    }
  },
  "service": {
    "name": "my-service"
  },
  "environment": {
    "account_id": "123456789012",
    "name": "my-env-staging",
    "outputs": {
      "vpc-id": "hdh2323423"
    }
  }
}
```

Paramètres de sortie

Pour renvoyer les résultats de mise à disposition des ressources AWS Proton, votre code de provisionnement peut générer un fichier JSON nommé `proton-outputs.json` avec les valeurs des paramètres de sortie définis dans le fichier de schéma de votre modèle. Par exemple, l'`--outputs-file`argument de la `cdk deploy` commande indique de générer un fichier JSON avec des sorties de provisionnement. AWS CDK Si votre ressource utilise le AWS CDK, spécifiez la commande suivante dans votre CodeBuild modèle de manifeste :

```
aws proton notify-resource-deployment-status-change
```

AWS Proton recherche ce fichier JSON. Si le fichier existe une fois que votre code d'approvisionnement a été correctement terminé, il AWS Proton lit les valeurs des paramètres de sortie à partir de celui-ci.

Détails et exemples des paramètres du fichier d'infrastructure en tant que code (IaC) Terraform

Vous pouvez inclure des variables d'entrée Terraform dans les `variable.tf` fichiers de votre ensemble de modèles. Vous pouvez également créer un schéma pour créer des variables AWS Proton gérées. AWS Proton crée une variable `.tf files` à partir de votre fichier de schéma. Pour de plus amples informations, veuillez consulter the section called “Fichiers Terraform iAC”.

Pour référencer les AWS Proton variables définies par votre schéma dans votre infrastructure`.tf files`, vous utilisez les AWS Proton espaces de noms indiqués dans la table Paramètres et espaces de noms pour Terraform iAC. Par exemple, vous pouvez utiliser `var.environment.inputs.vpc_cidr`. Entre guillemets, entourez ces variables de crochets simples et ajoutez un signe dollar devant la première accolade (par exemple,`“${var.environment.inputs.vpc_cidr}”`).

L'exemple suivant montre comment utiliser les espaces de noms pour inclure des AWS Proton paramètres dans un environnement. `.tf file`

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
```

```
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

AWS Proton infrastructure sous forme de fichiers de code

Les principaux éléments du bundle de modèles sont des fichiers d'infrastructure sous forme de code (IaC) qui définissent les ressources et les propriétés de l'infrastructure que vous souhaitez provisionner. AWS CloudFormation et d'autres infrastructures, car les moteurs de code utilisent ces types de fichiers pour provisionner les ressources de l'infrastructure.

> **Note**
>
> Un fichier IaC peut également être utilisé indépendamment des ensembles de modèles, en tant qu'entrée directe vers des composants directement définis. Pour plus d'informations sur les composants, consultezÉléments.

AWS Proton supporte actuellement deux types de fichiers IaC :

- CloudFormationfiles — Utilisé pour le provisionnement AWS géré. AWS Proton utilise Jinja en plus du format de fichier CloudFormation modèle pour le paramétrage.
- Fichiers Terraform HCL : utilisés pour le provisionnement autogéré. HCL prend en charge le paramétrage de manière native.

Vous ne pouvez pas provisionner AWS Proton des ressources à l'aide d'une combinaison de plusieurs méthodes de provisionnement. Vous devez utiliser l'un ou l'autre. Vous ne pouvez pas déployer un service de provisionnement AWS géré dans un environnement de provisionnement autogéré, ou vice versa.

Pour plus d'informations, consultez the section called "Méthodes de provisionnement", Environnements, Services et Éléments.

CloudFormation fichiers IaC

Découvrez comment utiliser l' AWS CloudFormation infrastructure sous forme de fichiers de code avec AWS Proton. CloudFormation est un service d'infrastructure sous forme de code (IaC) qui vous aide à modéliser et à configurer vos AWS ressources. Vous définissez les ressources de votre infrastructure dans des modèles, en utilisant Jinja en plus du format de fichier CloudFormation modèle pour le paramétrage. AWS Proton développe les paramètres et affiche le CloudFormation modèle complet. CloudFormation provisionne les ressources définies sous forme de CloudFormation pile. Pour plus d'informations, consultez la section Contenu CloudFormation du guide de CloudFormation l'utilisateur.

AWS Proton prend en charge AWS le provisionnement géré pour CloudFormation IaC.

Commencez avec votre propre infrastructure existante sous forme de fichiers de code

Vous pouvez adapter votre propre infrastructure existante sous forme de fichiers de code (IaC) pour les utiliser avec AWS Proton.

Les CloudFormation exemples suivants, l'exemple 1 et l'exemple 2, représentent vos propres fichiers CloudFormation IaC existants. CloudFormation peut utiliser ces fichiers pour créer deux CloudFormation piles différentes.

Dans l'exemple 1, le fichier CloudFormation IaC est configuré pour provisionner l'infrastructure à partager entre les applications de conteneur. Dans cet exemple, des paramètres d'entrée sont ajoutés afin que vous puissiez utiliser le même fichier IaC pour créer plusieurs ensembles d'infrastructures provisionnées. Chaque ensemble peut avoir des noms différents ainsi qu'un ensemble différent de valeurs de VPC et de CIDR de sous-réseau. En tant qu'administrateur ou développeur, vous fournissez des valeurs pour ces paramètres lorsque vous utilisez un fichier IaC pour provisionner des ressources d'infrastructure CloudFormation. Pour votre commodité, ces paramètres d'entrée sont marqués de commentaires et référencés à plusieurs reprises dans l'exemple. Les sorties sont définies à la fin du modèle. Ils peuvent être référencés dans d'autres fichiers CloudFormation IaC.

Dans l'exemple 2, le fichier CloudFormation iAC est configuré pour déployer une application sur l'infrastructure provisionnée à partir de l'exemple 1. Les paramètres sont commentés pour votre commodité.

Exemple 1 : fichier CloudFormation IaC

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery namespaces.
Parameters:
   VpcCIDR:       # input parameter
        Description: CIDR for VPC
        Type: String
        Default: "10.0.0.0/16"
   SubnetOneCIDR: # input parameter
        Description: CIDR for SubnetOne
        Type: String
        Default: "10.0.0.0/24"
   SubnetTwoCIDR: # input parameters
        Description: CIDR for SubnetTwo
        Type: String
        Default: "10.0.1.0/24"
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock:
        Ref: 'VpcCIDR'

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
         Ref: 'SubnetOneCIDR'
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
```

```
        Fn::Select:
        - 1
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
        Ref: 'SubnetTwoCIDR'
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
```

```
  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values will be available to other templates to use.
Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSCluster"
  ECSTaskExecutionRole:                                  # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSTaskExecutionRole"
  VpcId:                                                 # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-VPC"
  PublicSubnetOne:                                       # output
```

```
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetOne"
  PublicSubnetTwo:                                          # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetTwo"
  ContainerSecurityGroup:                                   # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ContainerSecurityGroup"
```

Exemple 2 : fichier CloudFormation IaC

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Parameters:
    ContainerPortInput:  # input parameter
        Description: The port to route traffic to
        Type: Number
        Default: 80
    TaskCountInput:      # input parameter
        Description: The default number of Fargate tasks you want running
        Type: Number
        Default: 1
    TaskSizeInput:       # input parameter
        Description: The size of the task you want to run
        Type: String
        Default: x-small
    ContainerImageInput: # input parameter
        Description: The name/url of the container image
        Type: String
        Default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
    TaskNameInput:       # input parameter
        Description: Name for your task
        Type: String
```

```
        Default: "my-fargate-instance"
    StackName:           # input parameter
        Description: Name of the environment stack to deploy to
        Type: String
        Default: "my-fargate-environment"
Mappings:
  TaskSizeMap:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName:
        Ref: 'TaskNameInput' # input parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref 'TaskNameInput'
      Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
      Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn:
        Fn::ImportValue:
```

```
          !Sub "${StackName}-ECSTaskExecutionRole"    # output parameter from another
CloudFormation template
             awslogs-region: !Ref 'AWS::Region'
             awslogs-stream-prefix: !Ref 'TaskNameInput'


 # The service_instance. The service is a resource which allows you to run multiple
 # copies of a type of task, and gather up their logs and metrics, as well
 # as monitor the number of running tasks and replace any that have crashed
 Service:
   Type: AWS::ECS::Service
   DependsOn: LoadBalancerRule
   Properties:
     ServiceName: !Ref 'TaskNameInput'
     Cluster:
       Fn::ImportValue:
         !Sub "${StackName}-ECSCluster"  # output parameter from another
CloudFormation template
     LaunchType: FARGATE
     DeploymentConfiguration:
       MaximumPercent: 200
       MinimumHealthyPercent: 75
     DesiredCount: !Ref 'TaskCountInput'
     NetworkConfiguration:
       AwsvpcConfiguration:
         AssignPublicIp: ENABLED
         SecurityGroups:
           - Fn::ImportValue:
               !Sub "${StackName}-ContainerSecurityGroup"    # output parameter from
another CloudFormation template
         Subnets:
           - Fn::ImportValue:r CloudFormation template
     TaskRoleArn: !Ref "AWS::NoValue"
     ContainerDefinitions:
       - Name: !Ref 'TaskNameInput'
         Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
         Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
         Image: !Ref 'ContainerImageInput'             # input parameter
         PortMappings:
           - ContainerPort: !Ref 'ContainerPortInput'  # input parameter

         LogConfiguration:
           LogDriver: 'awslogs'
           Options:
```

```
              awslogs-group: !Ref 'TaskNameInput'
                !Sub "${StackName}-PublicSubnetOne"    # output parameter from another
CloudFormation template
           - Fn::ImportValue:
                !Sub "${StackName}-PublicSubnetTwo"    # output parameter from another
CloudFormation template
     TaskDefinition: !Ref 'TaskDefinition'
     LoadBalancers:
       - ContainerName: !Ref 'TaskNameInput'
         ContainerPort: !Ref 'ContainerPortInput'  # input parameter
         TargetGroupArn: !Ref 'TargetGroup'

 # A target group. This is used for keeping track of all the tasks, and
 # what IP addresses / port numbers they have. You can query it yourself,
 # to use the addresses yourself, but most often this target group is just
 # connected to an application load balancer, or network load balancer, so
 # it can automatically distribute traffic across all the targets.
 TargetGroup:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     HealthCheckIntervalSeconds: 6
     HealthCheckPath: /
     HealthCheckProtocol: HTTP
     HealthCheckTimeoutSeconds: 5
     HealthyThresholdCount: 2
     TargetType: ip
     Name: !Ref 'TaskNameInput'
     Port: !Ref 'ContainerPortInput'
     Protocol: HTTP
     UnhealthyThresholdCount: 2
     VpcId:
       Fn::ImportValue:
         !Sub "${StackName}-VPC"    # output parameter from another CloudFormation
template

 # Create a rule on the load balancer for routing traffic to the target group
 LoadBalancerRule:
   Type: AWS::ElasticLoadBalancingV2::ListenerRule
   Properties:
     Actions:
       - TargetGroupArn: !Ref 'TargetGroup'
         Type: 'forward'
     Conditions:
       - Field: path-pattern
```

```
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"  # output parameter from another
 CloudFormation template
```

```
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60
```

```
# Create alarms to trigger these policies
LowCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - low-cpu
          - !Ref 'TaskNameInput'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "Low CPU utilization for service"
          - !Ref 'TaskNameInput'
    MetricName: CPUUtilization
    Namespace: AWS/ECS
    Dimensions:
      - Name: ServiceName
        Value: !Ref 'TaskNameInput'
      - Name: ClusterName
        Value:
          Fn::ImportValue:
            !Sub "${StackName}-ECSCluster"
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 20
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleDownPolicy

HighCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - high-cpu
          - !Ref 'TaskNameInput'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "High CPU utilization for service"
```

```
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId:
        Fn::ImportValue:
          !Sub "${StackName}-ContainerSecurityGroup"
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1
```

```
  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetOne"
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetTwo"
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
# These output values will be available to other templates to use.
Outputs:
  ServiceEndpoint:        # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

Vous pouvez adapter ces fichiers pour les utiliser avec AWS Proton.

Transférez votre infrastructure sous forme de code à AWS Proton

Avec de légères modifications, vous pouvez utiliser l'exemple 1 en tant que fichier d'infrastructure en tant que code (IaC) pour un ensemble de modèles d'environnement AWS Proton utilisé pour déployer un environnement (comme indiqué dans l'exemple 3).

Au lieu d'utiliser les CloudFormation paramètres, vous utilisez la syntaxe Jinja pour référencer les paramètres que vous avez définis dans un fichier de schéma basé sur une API ouverte. Ces paramètres d'entrée sont commentés pour votre commodité et référencés plusieurs fois dans le fichier IaC. De cette façon, AWS Proton vous pouvez auditer et vérifier les valeurs des paramètres. Il peut également faire correspondre et insérer les valeurs des paramètres de sortie d'un fichier IaC aux paramètres d'un autre fichier IaC.

En tant qu'administrateur, vous pouvez ajouter l'espace de AWS Proton `environment.inputs.` noms aux paramètres d'entrée. Lorsque vous référencez les sorties d'un fichier IaC de l'environnement dans un fichier IaC de service, vous pouvez ajouter l'`environment.outputs.`espace de noms aux sorties (par exemple,`environment.outputs.ClusterName`). Enfin, vous les entourez d'accolades et de guillemets.

Avec ces modifications, vos fichiers CloudFormation iAc peuvent être utilisés par AWS Proton.

Exemple 3 : infrastructure d' AWS Proton environnement sous forme de fichier de code

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery prefixes.
Mappings:
  # The VPC and subnet configuration is passed in via the environment spec.
  SubnetConfig:
    VPC:
      CIDR: '{{ environment.inputs.vpc_cidr}}'        # input parameter
    PublicOne:
      CIDR: '{{ environment.inputs.subnet_one_cidr}}' # input parameter
    PublicTwo:
      CIDR: '{{ environment.inputs.subnet_two_cidr}}' # input parameter
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock: !FindInMap ['SubnetConfig', 'VPC', 'CIDR']

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicOne', 'CIDR']
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 1
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicTwo', 'CIDR']
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
```

```
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster

  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'service_instance.environment.outputs.' namespace, for example,
 service_instance.environment.outputs.ClusterName.

Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:                                 # output
```

```
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:                                                # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
  PublicSubnetOne:                                      # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
  PublicSubnetTwo:                                      # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
  ContainerSecurityGroup:                               # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
```

Les fichiers IaC de l'exemple 1 et de l'exemple 3 produisent des CloudFormation piles légèrement différentes. Les paramètres sont affichés différemment dans les fichiers modèles de pile. Le fichier de modèle de CloudFormation pile Example 1 affiche les étiquettes de paramètres (clés) dans la vue du modèle de pile. Le fichier modèle de pile AWS Proton CloudFormation d'infrastructure de l'exemple 3 affiche les valeurs des paramètres. AWS Proton les paramètres d'entrée n'apparaissent pas dans la vue des paramètres de la CloudFormation pile de consoles.

Dans l'exemple 4, le fichier AWS Proton IaC du service correspond à l'exemple 2.

Exemple 4 : fichier IaC de l'instance de AWS Proton service

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
```

```
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}'
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
 parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}' # output from an
 environment infrastructure as code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}'
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}'

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
```

```
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'
      Cluster: '{{environment.outputs.ClusterName}}' # output from an environment
infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'       # input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output from an
environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}'        # output from an
environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}'
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'
          ContainerPort: '{{service_instance.inputs.port}}'
          TargetGroupArn: !Ref 'TargetGroup'

  # A target group. This is used for keeping track of all the tasks, and
  # what IP addresses / port numbers they have. You can query it yourself,
  # to use the addresses yourself, but most often this target group is just
  # connected to an application load balancer, or network load balancer, so
  # it can automatically distribute traffic across all the targets.
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Name: '{{service_instance.name}}'
      Port: '{{service_instance.inputs.port}}'
      Protocol: HTTP
      UnhealthyThresholdCount: 2
```

```
      VpcId: '{{environment.outputs.VpcId}}' # output from an environment
 infrastructure as code file

  # Create a rule on the load balancer for routing traffic to the target group
  LoadBalancerRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      Conditions:
        - Field: path-pattern
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}' # output from an environment
 infrastructure as code file
            - '{{service_instance.name}}'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
```

```
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
```

```
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60

  # Create alarms to trigger these policies
  LowCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - low-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "Low CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 20
      ComparisonOperator: LessThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleDownPolicy

  HighCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
```

```
        Fn::Join:
          - '-'
          - - high-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: '{{environment.outputs.ContainerSecurityGroup}}'
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: '{{environment.outputs.VpcId}}'
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
```

```
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - '{{environment.outputs.PublicSubnetOne}}'
        - '{{environment.outputs.PublicSubnetTwo}}'
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
Outputs:
  ServiceEndpoint:         # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

Dans l'exemple 5, le fichier iAC du AWS Proton pipeline approvisionne l'infrastructure du pipeline pour prendre en charge les instances de service fournies par l'exemple 4.

Exemple 5 : fichier AWS Proton IaC du pipeline de services

```
Resources:
  ECRRepo:
    Type: AWS::ECR::Repository
    DeletionPolicy: Retain
```

```
  BuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: true
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: repo_name
          Type: PLAINTEXT
          Value: !Ref ECRRepo
        - Name: service_name
          Type: PLAINTEXT
          Value: '{{ service.name }}'    # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - PublishRole
          - Arn
      Source:
        BuildSpec:
          Fn::Join:
            - ""
            - - >-
                {
                  "version": "0.2",
                  "phases": {
                    "install": {
                      "runtime-versions": {
                        "docker": 18
                      },
                      "commands": [
                        "pip3 install --upgrade --user awscli",
                        "echo
 'f6bd1536a743ab170b35c94ed4c7c4479763356bd543af5d391122f4af852460  yq_linux_amd64' >
 yq_linux_amd64.sha",
                        "wget https://github.com/mikefarah/yq/releases/download/3.4.0/
yq_linux_amd64",
                        "sha256sum -c yq_linux_amd64.sha",
                        "mv yq_linux_amd64 /usr/bin/yq",
                        "chmod +x /usr/bin/yq"
                      ]
```

```
                    },
                    "pre_build": {
                      "commands": [
                        "cd $CODEBUILD_SRC_DIR",
                        "$(aws ecr get-login --no-include-email --region
 $AWS_DEFAULT_REGION)",
                        "{{ pipeline.inputs.unit_test_command }}",    # input parameter
                      ]
                    },
                    "build": {
                      "commands": [
                        "IMAGE_REPO_NAME=$repo_name",
                        "IMAGE_TAG=$CODEBUILD_BUILD_NUMBER",
                        "IMAGE_ID=
              - Ref: AWS::AccountId
              - >-
                .dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG",
                        "docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG -f
 {{ pipeline.inputs.dockerfile }} .",     # input parameter
                        "docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_ID;",
                        "docker push $IMAGE_ID"
                      ]
                    },
                    "post_build": {
                      "commands": [
                        "aws proton --region $AWS_DEFAULT_REGION get-service --name
 $service_name | jq -r .service.spec > service.yaml",
                        "yq w service.yaml 'instances[*].spec.image' \"$IMAGE_ID\" >
 rendered_service.yaml"
                      ]
                    }
                  },
                  "artifacts": {
                    "files": [
                      "rendered_service.yaml"
                    ]
                  }
                }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
```

```
{% for service_instance in service_instances %}
  Deploy{{loop.index}}Project:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: false
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: service_name
          Type: PLAINTEXT
          Value:  '{{service.name}}'          # resource parameter
        - Name: service_instance_name
          Type: PLAINTEXT
          Value: '{{service_instance.name}}'  # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - DeploymentRole
          - Arn
      Source:
        BuildSpec: >-
          {
            "version": "0.2",
            "phases": {
              "build": {
                "commands": [
                  "pip3 install --upgrade --user awscli",
                  "aws proton --region $AWS_DEFAULT_REGION update-service-instance
 --deployment-type CURRENT_VERSION --name $service_instance_name --service-name
 $service_name --spec file://rendered_service.yaml",
                  "aws proton --region $AWS_DEFAULT_REGION wait service-instance-
deployed --name $service_instance_name --service-name $service_name"
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
```

```
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
```

```
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
```

```
            - s3:Abort*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: PublishRoleDefaultPolicy
    Roles:
      - Ref: PublishRole

DeploymentRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
```

```
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
```

```
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/Deploy*Project
                - -*
          - Action:
              - proton:UpdateServiceInstance
              - proton:GetServiceInstance
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
```

```
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
              - kms:Enable*
              - kms:List*
              - kms:Put*
              - kms:Update*
              - kms:Revoke*
              - kms:Disable*
              - kms:Get*
              - kms:Delete*
              - kms:ScheduleKeyDeletion
              - kms:CancelKeyDeletion
              - kms:GenerateDataKey
              - kms:TagResource
              - kms:UntagResource
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
```

```
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
```

```
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
      Version: "2012-10-17"
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineArtifactsBucket:
  Type: AWS::S3::Bucket
  Properties:
    VersioningConfiguration:
      Status: Enabled
    BucketEncryption:
      ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            KMSMasterKeyID:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
            SSEAlgorithm: aws:kms
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
  UpdateReplacePolicy: Retain
  DeletionPolicy: Retain
PipelineArtifactsBucketEncryptionKeyAlias:
  Type: AWS::KMS::Alias
  Properties:
    AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'
    TargetKeyId:
      Fn::GetAtt:
        - PipelineArtifactsBucketEncryptionKey
        - Arn
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineRole:
  Type: AWS::IAM::Role
  Properties:
```

```
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
```

```
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'
                FullRepositoryId: '{{ service.repository_id }}'
                BranchName: '{{ service.branch_name }}'
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
```

```
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
```

```
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
```

```
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"

                ]
              }
            }
```

```
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
```

```
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :log-group:/aws/codebuild/
                - Ref: BuildProject
                - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
```

```
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole
```

```
  DeploymentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
```

```
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/Deploy*Project
                  - -*
          - Action:
              - proton:UpdateServiceInstance
              - proton:GetServiceInstance
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
```

```
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: DeploymentRoleDefaultPolicy
    Roles:
      - Ref: DeploymentRole
PipelineArtifactsBucketEncryptionKey:
  Type: AWS::KMS::Key
  Properties:
    KeyPolicy:
      Statement:
        - Action:
            - kms:Create*
            - kms:Describe*
            - kms:Enable*
            - kms:List*
            - kms:Put*
            - kms:Update*
            - kms:Revoke*
            - kms:Disable*
            - kms:Get*
            - kms:Delete*
            - kms:ScheduleKeyDeletion
            - kms:CancelKeyDeletion
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
          Resource: "*"
        - Action:
            - kms:Decrypt
```

```
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
```

```
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
      Version: "2012-10-17"
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
 PipelineArtifactsBucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketEncryption:
      ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            KMSMasterKeyID:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
            SSEAlgorithm: aws:kms
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
  UpdateReplacePolicy: Retain
  DeletionPolicy: Retain
 PipelineArtifactsBucketEncryptionKeyAlias:
  Type: AWS::KMS::Alias
  Properties:
    AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'     # resource
parameter
    TargetKeyId:
      Fn::GetAtt:
        - PipelineArtifactsBucketEncryptionKey
        - Arn
```

```
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
```

```
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'   # resource
parameter
                FullRepositoryId: '{{ service.repository_id }}'            # resource
parameter
                BranchName: '{{ service.branch_name }}'                    # resource
parameter
              Name: Checkout
              OutputArtifacts:
```

```
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'         # resource parameter
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
```

```
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
```

```
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
```

```
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"
```

CodeBuild ensemble de modèles de provisionnement

Avec le CodeBuild provisionnement, au lieu d'utiliser des modèles IaC pour afficher les fichiers iAc et les exécuter à l'aide d'un moteur de provisionnement iAc, AWS Proton il suffit d'exécuter vos commandes shell. Pour ce faire, AWS Proton créez un AWS CodeBuild projet pour l'environnement, dans le compte d'environnement, et lancez une tâche pour exécuter vos commandes pour chaque création ou mise à jour de AWS Proton ressource. Lorsque vous créez un ensemble de modèles, vous fournissez un manifeste qui spécifie les commandes de provisionnement et de déprovisionnement de l'infrastructure, ainsi que tous les programmes, scripts et autres fichiers dont ces commandes peuvent avoir besoin. Vos commandes peuvent lire les entrées qui AWS Proton fournissent et sont responsables du provisionnement ou du déprovisionnement de l'infrastructure et de la génération de valeurs de sortie.

Le manifeste indique également comment AWS Proton doit être affiché le fichier d'entrée que votre code peut saisir et à partir duquel les valeurs d'entrée doivent être obtenues. Il peut être rendu au format JSON ou HCL. Pour plus d'informations sur les paramètres d'entrée, consultez la section the section called "CodeBuild paramètres de provisionnement". Pour plus d'informations sur les fichiers manifeste, consultez the section called "Manifeste et résumé".

> **Note**
>
> Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

Exemple : utilisation du AWS CDK with CodeBuild provisioning

À titre d'exemple d'utilisation du CodeBuild provisionnement, vous pouvez inclure du code qui utilise les AWS ressources AWS Cloud Development Kit (AWS CDK) pour provisionner (déployer) et déprovisionner (détruire), ainsi qu'un manifeste qui installe le CDK et exécute votre code CDK.

Les sections suivantes répertorient des exemples de fichiers que vous pouvez inclure dans un ensemble de modèles de CodeBuild provisionnement qui approvisionne un environnement à l'aide du AWS CDK.

Manifeste

Le fichier manifeste suivant spécifie le CodeBuild provisionnement et inclut les commandes nécessaires pour installer et utiliser le AWS CDK, le traitement des fichiers de sortie et les rapports sur les résultats vers AWS Proton.

Example infrastructure/manifest.yaml

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Schema

Le fichier de schéma suivant définit les paramètres de l'environnement. Votre AWS CDK code peut faire référence aux valeurs de ces paramètres lors du déploiement.

Example schéma/schema.yaml

```
schema:
```

```
  format:
    openapi: "3.0.0"
  environment_input_type: "MyEnvironmentInputType"
  types:
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:
          type: string
          description: "Another sample input"
      required:
        - my_other_sample_input
```

AWS CDK fichiers

Les fichiers suivants constituent un exemple de projet CDK Node.js.

Example infrastructure/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironmente": "bin/ProtonEnvironment.js"
  },
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk"
  },
  "devDependencies": {
    "@types/jest": "^28.1.7",
    "@types/node": "18.7.6",
    "jest": "^28.1.3",
    "ts-jest": "^28.0.8",
    "aws-cdk": "2.37.1",
    "ts-node": "^10.9.1",
```

```
    "typescript": "~4.7.4"
  },
  "dependencies": {
    "aws-cdk-lib": "2.37.1",
    "constructs": "^10.1.77",
    "source-map-support": "^0.5.21"
  }
}
```

Example infrastructure/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2018",
    "module": "commonjs",
    "lib": [
      "es2018"
    ],
    "declaration": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noImplicitThis": true,
    "alwaysStrict": true,
    "noUnusedLocals": false,
    "noUnusedParameters": false,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": false,
    "inlineSourceMap": true,
    "inlineSources": true,
    "experimentalDecorators": true,
    "strictPropertyInitialization": false,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "typeRoots": [
      "./node_modules/@types"
    ]
  },
  "exclude": [
    "node_modules",
    "cdk.out"
  ]
}
```

Example infrastructure/cdk.json

```
{
  "app": "npx ts-node --prefer-ts-exts bin/ProtonEnvironment.ts",
  "outputsFile": "proton-outputs.json",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
      "yarn.lock",
      "node_modules",
      "test"
    ]
  },
  "context": {
    "@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId": true,
    "@aws-cdk/core:stackRelativeExports": true,
    "@aws-cdk/aws-rds:lowercaseDbIdentifier": true,
    "@aws-cdk/aws-lambda:recognizeVersionProps": true,
    "@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2_2021": true,
    "@aws-cdk-containers/ecs-service-extensions:enableDefaultLogDriver": true,
    "@aws-cdk/aws-ec2:uniqueImdsv2TemplateName": true,
    "@aws-cdk/core:target-partitions": [
      "aws",
      "aws-cn"
    ]
  }
}
```

Example infrastructure/bin/ProtonEnvironment.ts

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { ProtonEnvironmentStack } from '../lib/ProtonEnvironmentStack';
```

```
const app = new cdk.App();
new ProtonEnvironmentStack(app, 'ProtonEnvironmentStack', {});
```

Example infrastructure/lib/ProtonEnvironmentStack.ts

```
import { Stack, StackProps } from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import input from '../proton-inputs.json';

export class ProtonEnvironmentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, { ...props, stackName: process.env.STACK_NAME });

    const ssmParam = new ssm.StringParameter(this, "ssmParam", {
      stringValue: input.environment.inputs.my_sample_input,
      parameterName: `${process.env.STACK_NAME}-Param`,
      tier: ssm.ParameterTier.STANDARD
    })

    new cdk.CfnOutput(this, 'ssmParamOutput', {
      value: ssmParam.parameterName,
      description: 'The name of the ssm parameter',
      exportName: `${process.env.STACK_NAME}-Param`
    });
  }
}
```

Fichier d'entrée rendu

Lorsque vous créez un environnement à l'aide d'un modèle de provisionnement CodeBuild basé, AWS Proton affiche un fichier d'entrée avec les valeurs de paramètres d'entrée que vous avez fournies. Votre code peut faire référence à ces valeurs. Le fichier suivant est un exemple de fichier d'entrée rendu.

Example infrastructure/proton-inputs.json

```
{
  "environment": {
    "name": "myenv",
    "inputs": {
      "my_sample_input": "10.0.0.0/16",
```

```
      "my_other_sample_input": "11.0.0.0/16"
    }
  }
}
```

Fichiers Terraform iAC

Apprenez à utiliser l'infrastructure Terraform sous forme de fichiers de code (IaC) avec. AWS ProtonTerraform est un moteur IaC open source largement utilisé qui a été développé par. HashiCorp Les modules Terraform sont développés dans le langage HashiCorp HCL et prennent en charge plusieurs fournisseurs d'infrastructures dorsales, dont Amazon Web Services.

AWS Proton prend en charge le provisionnement autogéré pour Terraform iAc.

Pour un exemple complet de référentiel de provisionnement qui répond aux pull requests et implémente le provisionnement de l'infrastructure, voir le modèle d'automatisation Terraform OpenSource GitHub Actions pour on. AWS Proton GitHub

Comment fonctionne le provisionnement autogéré avec les fichiers groupés de modèles Terraform iAc :

1. Lorsque vous créez un environnement à partir de ensembles de modèles Terraform, AWS Proton compilez vos `.tf` fichiers avec des paramètres de console ou d'entrée. `spec file`
2. Il effectue une pull request pour fusionner les fichiers iAc compilés avec le référentiel auprès duquel vous vous êtes enregistré AWS Proton.
3. Si la demande est approuvée, AWS Proton attend le statut d'approvisionnement que vous fournissez.
4. Si la demande est rejetée, la création de l'environnement est annulée.
5. Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.

AWS Proton avec les considérations relatives à Terraform IaC :

- AWS Proton ne gère pas votre approvisionnement en Terraform.
- Vous devez enregistrer un référentiel de provisioning auprès AWS Proton de. AWS Proton effectue des pull requests sur ce dépôt.
- Vous devez créer une CodeStar connexion pour vous connecter AWS Proton à votre référentiel de provisioning.

- Pour effectuer un provisionnement à partir de fichiers IaC AWS Proton compilés, vous devez répondre aux AWS Proton pull requests. AWS Proton effectue des pull requests après la création et la mise à jour des actions de l'environnement et du service. Pour plus d'informations, consultez AWS Proton environnements et AWS Proton services.
- Pour approvisionner un pipeline à partir de fichiers iAc AWS Proton compilés, vous devez créer un référentiel de CI/CD pipelines.
- Votre automatisation du provisionnement basée sur les pull requests doit inclure des étapes pour signaler tout AWS Proton changement d'état des AWS Proton ressources provisionnées. Vous pouvez utiliser l' AWS Proton NotifyResourceDeploymentStatusChange API.
- Vous ne pouvez pas déployer de services, de pipelines et de composants créés à partir de fichiers CloudFormation iAC dans des environnements créés à partir de fichiers Terraform iAC.
- Vous ne pouvez pas déployer de services, de pipelines et de composants créés à partir de fichiers IaC Terraform dans des environnements créés à partir de fichiers CloudFormation iAC.

Lorsque vous préparez vos fichiers Terraform iAC pour AWS Proton, vous attachez des espaces de noms à vos variables d'entrée, comme indiqué dans les exemples suivants. Pour plus d'informations, consultez Parameters.

Exemple 1 : AWS Proton environnement : fichier IaC Terraform

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
```

```
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

Infrastructure compilée sous forme de code

Lorsque vous créez un environnement ou un service, AWS Proton compilez votre infrastructure sous forme de fichiers de code avec console ou `spec file` entrées. Il crée `proton.`*`resource-type`*`.variables.tf` des `proton.auto.tfvars.json` fichiers pour vos entrées qui peuvent être utilisés par Terraform, comme indiqué dans les exemples suivants. Ces fichiers se trouvent dans un référentiel spécifié dans un dossier qui correspond au nom de l'environnement ou de l'instance de service.

L'exemple montre comment AWS Proton inclure des balises dans la définition et les valeurs des variables, et comment vous pouvez propager ces AWS Proton balises aux ressources provisionnées. Pour de plus amples informations, veuillez consulter the section called “Propagation des balises vers les ressources provisionnées”.

Exemple 2 : fichiers IaC compilés pour un environnement nommé « dev ».

dev/environment.tf :

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
```

```
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

dev/proton.environment.variables.tf :

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

dev/proton.auto.tfvars.json :

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }
```

```
  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

Chemins du référentiel

AWS Proton utilise les entrées de console ou de spécification provenant d'actions de création d'environnements ou de services pour trouver le référentiel et le chemin où se trouvent les fichiers iAc compilés. Les valeurs d'entrée sont transmises aux paramètres d'entrée avec espace de noms.

AWS Proton prend en charge deux configurations de chemin de dépôt. Dans les exemples suivants, les chemins sont nommés selon les paramètres de ressources avec espace de noms provenant de deux environnements. Chaque environnement possède des instances de service de deux services, et les instances de service de l'un des services ont des composants directement définis.

Type de ressource	Paramètre de nom	=	Nom de la ressource
Environnement	`environment.name`		`"env-prod"`
Environnement	`environment.name`		`"env-staged"`
Service	`service.name`	=	`"service-one"`
Instance de service	`service_instance.name`		`"instance-one-prod"`
Instance de service	`service_instance.name`		`"instance-one-staged"`

Type de ressource	Paramètre de nom	=	Nom de la ressource
Service	`service.name`		`"service-two"`
Instance de service	`service_instance.name`		`"instance-two-prod"`
Composant	`service_instance.components.default.name`		`"component-prod"`
Instance de service	`service_instance.name`		`"instance-two-staged"`
Composant	`service_instance.components.default.name`		`"component-staged"`

Layout 1

S'il AWS Proton trouve le référentiel spécifié avec un `environments` dossier, il crée un dossier qui inclut les fichiers iAc compilés et porte le nom`environment.name`.

S'il AWS Proton trouve le référentiel spécifié avec un `environments` dossier contenant un nom de dossier correspondant à un nom d'environnement compatible avec une instance de service, il crée un dossier qui inclut les fichiers iAC de l'instance compilée et porte le nom de`service_instance.name`.

```
/repo
    /environments
        /env-prod                               # environment folder
            main.tf
            proton.environment.variables.tf
            proton.auto.tfvars.json

            /service-one-instance-one-prod    # instance folder
```

```
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                   # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                           # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                 # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Layout 2

S'il AWS Proton trouve le référentiel spécifié sans `environments` dossier, il crée un `environment.name` dossier dans lequel il localise les fichiers iAc de l'environnement compilé.

S'il AWS Proton trouve le référentiel spécifié dont le nom de dossier correspond à un nom d'environnement compatible avec une instance de service, il crée un `service_instance.name` dossier dans lequel il localise les fichiers iAc de l'instance compilée.

```
/repo
    /env-prod                             # environment folder
        main.tf
        proton.environment.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                   # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                           # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                 # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Fichier de schéma

En tant qu'administrateur, lorsque vous utilisez la section Modèles de données d'API ouvertes (schémas) pour définir un fichier YAML de schéma de paramètres pour votre ensemble de modèles, vous AWS Proton pouvez valider les entrées de valeurs de paramètres par rapport aux exigences que vous avez définies dans votre schéma.

Pour plus d'informations sur les formats et les mots clés disponibles, consultez la section relative aux objets Schema de l'OpenAPI.

Exigences relatives au schéma pour les ensembles de modèles d'environnement

Votre schéma doit suivre la section Modèles de données (schémas) de l'OpenAPI au format YAML. Il doit également faire partie de votre ensemble de modèles d'environnement.

Pour votre schéma d'environnement, vous devez inclure les en-têtes formatés pour indiquer que vous utilisez la section Modèles de données (schémas) de l'Open API. Dans les exemples de schéma d'environnement suivants, ces en-têtes apparaissent dans les trois premières lignes.

Un `environment_input_type` doit être inclus et défini avec un nom que vous fournissez. Dans les exemples suivants, cela est défini à la ligne 5. En définissant ce paramètre, vous l'associez à une ressource d' AWS Proton environnement.

Pour suivre le modèle de schéma Open API, vous devez inclure`types`. Dans l'exemple suivant, il s'agit de la ligne 6.

Ensuite`types`, vous devez définir un `environment_input_type` type. Vous définissez les paramètres d'entrée de votre environnement en tant que propriétés du`environment_input_type`. Vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans le fichier de code (IAc) de l'infrastructure environnementale associé au schéma.

Lorsque vous créez un environnement et que vous fournissez des valeurs de paramètres AWS Proton personnalisées, utilisez le fichier de schéma pour les associer, les valider et les injecter dans les paramètres frisés du fichier CloudFormation iAC associé. Pour chaque propriété (paramètre), indiquez un `name` et`type`. Facultativement, fournissez également un `description``default`, et`pattern`.

Les paramètres définis pour l'exemple de schéma de modèle d'environnement standard suivant incluent `vpc_cidr``subnet_one_cidr`, et `subnet_two_cidr` avec le `default` mot-clé et les

valeurs par défaut. Lorsque vous créez un environnement avec ce schéma de bundle de modèles d'environnement, vous pouvez accepter les valeurs par défaut ou fournir les vôtres. Si un paramètre n'a pas de valeur par défaut et est répertorié en tant que `required` propriété (paramètre), vous devez lui fournir des valeurs lorsque vous créez un environnement.

Le deuxième exemple de schéma de modèle d'environnement standard répertorie le `required` paramètre`my_other_sample_input`.

Vous pouvez créer un schéma pour deux types de modèles d'environnement. Pour de plus amples informations, veuillez consulter Enregistrez et publiez des modèles.

- Modèles d'environnement standard

  Dans l'exemple suivant, un type d'entrée d'environnement est défini avec une description et des propriétés d'entrée. Cet exemple de schéma peut être utilisé avec le fichier AWS Proton CloudFormation IaC présenté dans l'exemple 3.

  Exemple de schéma pour un modèle d'environnement standard :

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "PublicEnvironmentInput"
  types:                          # required
    # defined by administrator
    PublicEnvironmentInput:
      type: object
      description: "Input properties for my environment"
      properties:
        vpc_cidr:                  # parameter
          type: string
          description: "This CIDR range for your VPC"
          default: 10.0.0.0/16
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_one_cidr:           # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.0.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_two_cidr:           # parameter
          type: string
```

```
          description: "The CIDR range for subnet one"
          default: 10.0.1.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
```

Exemple de schéma pour un modèle d'environnement standard qui inclut un `required` paramètre :

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "MyEnvironmentInputType"
  types:                          # required
    # defined by administrator
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:          # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:    # parameter
          type: string
          description: "Another sample input"
        another_optional_input:   # parameter
          type: string
          description: "Another optional input"
          default: "!"
      required:
        - my_other_sample_input
```

- Modèles d'environnement gérés par le client

  Dans l'exemple suivant, le schéma inclut uniquement une liste de sorties qui reproduisent les sorties de l'iAc que vous avez utilisé pour approvisionner votre infrastructure gérée par le client. Vous devez définir les types de valeurs de sortie sous forme de chaînes uniquement (et non sous forme de listes, de tableaux ou d'autres types). Par exemple, l'extrait de code suivant montre la section des sorties d'un modèle externe CloudFormation . Cela provient du modèle présenté dans l'exemple 1. Il peut être utilisé pour créer une infrastructure externe gérée par le client pour un service AWS Proton Fargate créé à partir de l'exemple 4.

> **Important**
>
> En tant qu'administrateur, vous devez vous assurer que votre infrastructure provisionnée et gérée ainsi que tous les paramètres de sortie sont compatibles avec les modèles d'environnement gérés par le client associés. AWS Proton Je ne peux pas comptabiliser les modifications en votre nom, car elles ne sont pas visibles par AWS Proton. Les incohérences entraînent des échecs.

Exemples de sorties de fichiers CloudFormation IaC pour un modèle d'environnement géré par le client :

```
// Cloudformation Template Outputs
[...]
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

Le schéma du bundle de modèles d'environnement géré par le AWS Proton client correspondant est illustré dans l'exemple suivant. Chaque valeur de sortie est définie sous forme de chaîne.

Exemple de schéma pour un modèle d'environnement géré par le client :

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "EnvironmentOutput"
  types:                          # required
    # defined by administrator
    EnvironmentOutput:
      type: object
```

```
      description: "Outputs of the environment"
      properties:
        ClusterName:               # parameter
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:      # parameter
          type: string
          description: "The ARN of the ECS role"
        VpcId:                     # parameter
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Exigences relatives au schéma pour les ensembles de modèles de services

Votre schéma doit suivre la section Modèles de données (schémas) de l'OpenAPI au format YAML, comme indiqué dans les exemples suivants. Vous devez fournir un fichier de schéma dans votre ensemble de modèles de services.

Dans les exemples de schéma de service suivants, vous devez inclure les en-têtes formatés. Dans l'exemple suivant, cela se trouve dans les trois premières lignes. Cela permet de vérifier que vous utilisez la section Modèles de données (schémas) de l'Open API.

A `service_input_type` doit être inclus et défini avec un nom que vous fournissez. Dans l'exemple suivant, il s'agit de la ligne 5. Cela associe les paramètres à une ressource AWS Proton de service.

Un pipeline AWS Proton de services est inclus par défaut lorsque vous utilisez la console ou la CLI pour créer un service. Lorsque vous incluez un pipeline de services pour votre service, vous devez l'inclure `pipeline_input_type` avec un nom que vous fournissez. Dans l'exemple suivant, il s'agit de la ligne 7. N'incluez pas ce paramètre si vous n'incluez pas de pipeline AWS Proton de services. Pour de plus amples informations, veuillez consulter Enregistrez et publiez des modèles.

Pour suivre le modèle de schéma Open API, vous devez inclure `types` dans l'exemple suivant, cela se trouve dans la ligne 9.

Ensuite`types`, vous devez définir un `service_input_type` type. Vous définissez les paramètres d'entrée de votre service en tant que propriétés du`service_input_type`. Vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans le fichier d'infrastructure de service sous forme de code (IAc) associé au schéma.

Pour définir un pipeline de services, en dessous de votre `service_input_type` définition, vous devez définir un`pipeline_input_type`. Comme ci-dessus, vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans un fichier iAc de pipeline associé au schéma. N'incluez pas cette définition si vous n'incluez pas de pipeline AWS Proton de services.

Lorsque, en tant qu'administrateur ou développeur, vous créez un service et que vous fournissez des valeurs de paramètres personnalisées, vous AWS Proton utilisez le fichier de schéma pour les faire correspondre, les valider et les injecter dans les paramètres frisés du fichier CloudFormation IaC associé. Pour chaque propriété (paramètre), indiquez a `name` et `type` a. Facultativement, fournissez également un `description``default`, et`pattern`.

Les paramètres définis pour l'exemple de schéma incluent `port``desired_count`, `task_size` et `image` avec le `default` mot-clé et les valeurs par défaut. Lorsque vous créez un service avec ce schéma de bundle de modèles de services, vous pouvez accepter les valeurs par défaut ou fournir les vôtres. Le paramètre `unique_name` est également inclus dans l'exemple et ne possède pas de valeur par défaut. Il est répertorié en tant que `required` propriété (paramètre). En tant qu'administrateur ou développeur, vous devez fournir des valeurs pour `required` les paramètres lorsque vous créez des services.

Si vous souhaitez créer un modèle de service avec un pipeline de services, incluez-le `pipeline_input_type` dans votre schéma.

Exemple de fichier de schéma de service pour un service qui inclut un pipeline AWS Proton de services.

Cet exemple de schéma peut être utilisé avec les AWS Proton fichiers IaC présentés dans les exemples 4 et 5. Un pipeline de services est inclus.

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required           defined by administrator
  service_input_type: "LoadBalancedServiceInput"
  # only include if including AWS Proton service pipeline, defined by administrator
  pipeline_input_type: "PipelineInputs"

  types:                          # required
    # defined by administrator
    LoadBalancedServiceInput:
      type: object
```

```
      description: "Input properties for a loadbalanced Fargate service"
      properties:
        port:                       # parameter
          type: number
          description: "The port to route traffic to"
          default: 80
          minimum: 0
          maximum: 65535
        desired_count:              # parameter
          type: number
          description: "The default number of Fargate tasks you want running"
          default: 1
          minimum: 1
        task_size:                  # parameter
          type: string
          description: "The size of the task you want to run"
          enum: ["x-small", "small", "medium", "large", "x-large"]
          default: "x-small"
        image:                      # parameter
          type: string
          description: "The name/url of the container image"
          default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
          minLength: 1
          maxLength: 200
        unique_name:                # parameter
          type: string
          description: "The unique name of your service identifier. This will be used
to name your log group, task definition and ECS service"
          minLength: 1
          maxLength: 100
      required:
        - unique_name
    # defined by administrator
    PipelineInputs:
      type: object
      description: "Pipeline input properties"
      properties:
        dockerfile:                 # parameter
          type: string
          description: "The location of the Dockerfile to build"
          default: "Dockerfile"
          minLength: 1
          maxLength: 100
        unit_test_command:          # parameter
```

```
          type: string
          description: "The command to run to unit test the application code"
          default: "echo 'add your unit test command here'"
          minLength: 1
          maxLength: 200
```

Si vous souhaitez créer un modèle de service sans pipeline de services, ne l'incluez pas `pipeline_input_type` dans votre schéma, comme indiqué dans l'exemple suivant.

Exemple de fichier de schéma de service pour un service qui n'inclut pas de pipeline AWS Proton de services

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required            defined by administrator
  service_input_type: "MyServiceInstanceInputType"

  types:                          # required
    # defined by administrator
    MyServiceInstanceInputType:
      type: object
      description: "Service instance input properties"
      required:
        - my_sample_service_instance_required_input
      properties:
        my_sample_service_instance_optional_input:   # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_sample_service_instance_required_input:   # parameter
          type: string
          description: "Another sample input"
```

Résumez les fichiers modèles pour AWS Proton

Après avoir préparé les fichiers de votre environnement et de votre infrastructure de services sous forme de code (IaC) et leurs fichiers de schéma respectifs, vous devez les organiser dans des répertoires. Vous devez également créer un fichier manifeste YAML. Le fichier manifeste répertorie les fichiers iAc d'un répertoire, le moteur de rendu et le langage de modèle utilisé pour développer l'iAc dans ce modèle.

> **Note**
>
> Un fichier manifeste peut également être utilisé indépendamment des ensembles de modèles, en tant qu'entrée directe vers des composants directement définis. Dans ce cas, il spécifie toujours un seul fichier modèle IaC, à la fois pour Terraform CloudFormation et pour Terraform. Pour plus d'informations sur les composants, consultezÉléments.

Le fichier manifeste doit respecter le format et le contenu présentés dans l'exemple suivant.

CloudFormation format de fichier manifeste :

Avec CloudFormation, vous ne listez qu'un seul fichier.

```
infrastructure:
  templates:
    - file: "cloudformation.yaml"
      rendering_engine: jinja
      template_language: cloudformation
```

Format de fichier manifeste Terraform :

Avec Terraform, vous pouvez répertorier explicitement un seul fichier ou utiliser le caractère générique * pour répertorier chacun des fichiers d'un répertoire.

> **Note**
>
> Le caractère générique inclut uniquement les fichiers dont le nom se termine par `.tf`. Les autres fichiers sont ignorés.

```
infrastructure:
  templates:
    - file: "*"
      rendering_engine: hcl
      template_language: terraform
```

CodeBuildformat de fichier manifeste de provisionnement basé sur :

Avec le provisionnement CodeBuild basé, vous spécifiez les commandes shell de provisionnement et de déprovisionnement.

> **Note**
>
> Outre le manifeste, votre bundle doit inclure tous les fichiers dont dépendent vos commandes.

L'exemple de manifeste suivant utilise le provisionnement CodeBuild basé pour provisionner (déployer) et déprovisionner (détruire) les ressources à l'aide du AWS Cloud Development Kit (AWS CDK) (AWS CDK). Le bundle de modèles doit également inclure le code CDK.

Pendant le provisionnement, AWS Proton crée un fichier d'entrée contenant les valeurs des paramètres d'entrée que vous avez définis dans le schéma du modèle avec le nom`proton-input.json`.

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
```

```
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Après avoir configuré les répertoires et les fichiers manifestes pour votre environnement ou votre ensemble de modèles de services, vous compressez les répertoires dans une archive tar et vous les téléchargez dans un bucket Amazon Simple Storage Service (Amazon S3) AWS Proton où vous pouvez les récupérer, ou dans un référentiel Git de synchronisation de modèles.

Lorsque vous créez une version mineure d'un environnement ou d'un modèle de service auprès duquel vous vous êtes inscrit AWS Proton, vous indiquez le chemin d'accès à votre environnement ou à l'archive tar de votre ensemble de modèles de services qui se trouve dans votre compartiment S3. AWS Proton l'enregistre avec la nouvelle version mineure du modèle. Vous pouvez sélectionner la nouvelle version mineure du modèle pour créer ou mettre à jour des environnements ou des services AWS Proton.

Résumé du bundle de modèles d'environnement

Il existe deux types d'ensembles de modèles d'environnement pour AWS Proton lesquels vous pouvez créer.

- Pour créer un ensemble de modèles d'environnement pour un modèle d'environnement standard, organisez le schéma, les fichiers d'infrastructure sous forme de code (IaC) et le fichier manifeste dans des répertoires, comme indiqué dans la structure de répertoires du bundle de modèles d'environnement suivante.
- Pour créer un ensemble de modèles d'environnement pour un modèle d'environnement géré par le client, fournissez uniquement le fichier et le répertoire du schéma. N'incluez pas le répertoire et les fichiers de l'infrastructure. AWS Proton renvoie une erreur si le répertoire et les fichiers de l'infrastructure sont inclus.

Pour de plus amples informations, veuillez consulter Enregistrez et publiez des modèles.

CloudFormation structure du répertoire du bundle de modèles d'environnement :

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  cloudformation.yaml
```

Structure du répertoire du bundle de modèles d'environnement Terraform :

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  environment.tf
```

Résumé de l'offre groupée de modèles de services

Pour créer un ensemble de modèles de services, vous devez organiser le schéma, les fichiers d'infrastructure en tant que code (IaC) et les fichiers manifestes dans des répertoires, comme indiqué dans l'exemple de structure de répertoire du bundle de modèles de services.

Si vous n'incluez pas de pipeline de services dans votre ensemble de modèles, n'incluez pas le répertoire et les fichiers du pipeline et ne définissez pas le modèle de service à associer à ce bundle de modèles `"pipelineProvisioning": "CUSTOMER_MANAGED"` lorsque vous créez le modèle de service.

> **Note**
>
> Vous ne pouvez pas le modifier une `pipelineProvisioning` fois le modèle de service créé.

Pour de plus amples informations, veuillez consulter Enregistrez et publiez des modèles.

CloudFormation structure du répertoire du bundle de modèles de services :

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  cloudformation.yaml
/pipeline_infrastructure
  manifest.yaml
  cloudformation.yaml
```

Structure du répertoire du bundle de modèles de services Terraform :

```
/schema
  schema.yaml
```

```
/instance_infrastructure
  manifest.yaml
  instance.tf
/pipeline_infrastructure
  manifest.yaml
  pipeline.tf
```

Considérations relatives aux ensembles de modèles

- Fichiers d'infrastructure sous forme de code (IaC)

  AWS Proton modèles d'audits pour le format de fichier correct. Cependant, AWS Proton ne vérifie pas les erreurs de développement, de dépendance et de logique du modèle. Supposons, par exemple, que vous ayez spécifié la création d'un compartiment Amazon S3 dans votre fichier CloudFormation IaC dans le cadre de votre modèle de service ou d'environnement. Un service est créé sur la base de ces modèles. Supposons maintenant que vous souhaitiez supprimer le service à un moment donné. Si le compartiment S3 spécifié n'est pas vide et que le fichier CloudFormation iAC ne le marque pas comme étant `Retain` dans le`DeletionPolicy`, AWS Proton échoue lors de l'opération de suppression du service.
- Limites de taille et format des fichiers groupés
  - Les limites de taille, de nombre et de taille de nom des fichiers groupés sont disponibles à l'adresseAWS Proton quotas.
  - Les répertoires de fichiers des ensembles de modèles sont compressés dans une archive tar et situés dans un compartiment Amazon Simple Storage Service (Amazon S3).
  - Chaque fichier du bundle doit être un fichier YAML formaté valide.
- Chiffrement groupé de modèles de compartiments S3

  Si vous souhaitez chiffrer les données sensibles de vos ensembles de modèles au repos dans votre compartiment S3, utilisez les clés SSE-S3 ou SSE-KMS pour permettre de les récupérer. AWS Proton

AWS Proton modèles

Pour ajouter votre ensemble de modèles à votre bibliothèque de AWS Proton modèles, créez une version mineure du modèle et enregistrez-la auprès de celle-ci AWS Proton. Lors de la création du modèle, indiquez le nom du compartiment Amazon S3 et le chemin de votre bundle de modèles. Une fois les modèles publiés, ils peuvent être sélectionnés par les membres de l'équipe de la plateforme et les développeurs. Une fois qu'ils ont été sélectionnés, ils AWS Proton utilisent le modèle pour créer et provisionner l'infrastructure et les applications.

En tant qu'administrateur, vous pouvez créer et enregistrer un modèle d'environnement auprès de AWS Proton. Ce modèle d'environnement peut ensuite être utilisé pour déployer plusieurs environnements. Par exemple, il peut être utilisé pour déployer des environnements « dev », « staging » et « prod ». L'environnement « de développement » peut inclure un VPC avec des sous-réseaux privés et une politique d'accès restrictive à toutes les ressources. Les sorties de l'environnement peuvent être utilisées comme entrées pour les services.

Vous pouvez créer et enregistrer des modèles d'environnement pour créer deux types d'environnements différents. Vous et les développeurs pouvez AWS Proton déployer des services pour les deux types.

- Enregistrez et publiez un modèle d'environnement standard qui permet de créer un environnement standard qui AWS Proton provisionne et gère l'infrastructure de l'environnement.
- Enregistrez et publiez un modèle d'environnement géré par le client AWS Proton qui permet de créer un environnement géré par le client qui se connecte à votre infrastructure provisionnée existante. AWS Proton ne gère pas votre infrastructure provisionnée existante.

Vous pouvez créer et enregistrer des modèles de services AWS Proton pour déployer des services dans des environnements. Un AWS Proton environnement doit être créé avant qu'un service puisse y être déployé.

La liste suivante décrit comment créer et gérer des modèles avec AWS Proton.

- (Facultatif) Préparez un rôle IAM pour contrôler l'accès des développeurs aux appels d' AWS Proton API et aux rôles de service AWS Proton IAM. Pour de plus amples informations, veuillez consulter the section called “Rôles IAM”.
- Composez un ensemble de modèles. Pour de plus amples informations, veuillez consulter Packs de modèles.

- Créez et enregistrez un modèle une AWS Proton fois que le bundle de modèles a été composé, compressé et enregistré dans un compartiment Amazon S3. Vous pouvez le faire soit dans la console, soit en utilisant le AWS CLI.
- Testez et utilisez le modèle pour créer et gérer les ressources AWS Proton provisionnées après son enregistrement auprès AWS Proton de.
- Créez et gérez des versions majeures et secondaires du modèle tout au long de sa durée de vie.

Vous pouvez gérer les versions des modèles manuellement ou à l'aide de configurations de synchronisation des modèles :

- Utilisez la AWS Proton console AWS CLI pour créer une nouvelle version mineure ou majeure.
- Créez une configuration de synchronisation de modèles qui permet de créer AWS Proton automatiquement une nouvelle version mineure ou majeure lorsqu'elle détecte une modification de votre ensemble de modèles dans un référentiel que vous définissez.

Pour plus d'informations, consultez le manuel The AWS Proton Service API Reference.

Rubriques

Modèles versionnés

En tant qu'administrateur ou membre d'une équipe de plateforme, vous définissez, créez et gérez une bibliothèque de modèles versionnés utilisés pour fournir des ressources d'infrastructure. Il existe deux types de versions de modèles : les versions secondaires et les versions majeures.

- Versions mineures : modifications apportées au modèle dont le schéma est rétrocompatible. Ces modifications n'obligent pas le développeur à fournir de nouvelles informations lors de la mise à jour vers la nouvelle version du modèle.

  Lorsque vous essayez d'apporter une modification de version mineure AWS Proton , faites de votre mieux pour déterminer si le schéma de la nouvelle version est rétrocompatible avec les versions mineures précédentes du modèle. Si le nouveau schéma n'est pas rétrocompatible, l'enregistrement de la nouvelle version mineure AWS Proton échoue.

  > **Note**
  >
  > La compatibilité est déterminée uniquement en fonction du schéma. AWS Proton ne vérifie pas si le fichier d'infrastructure en tant que code (IaC) du bundle de modèles est rétrocompatible avec les versions mineures précédentes. Par exemple, AWS Proton ne vérifie pas si le nouveau fichier iAC entraîne des modifications importantes pour les applications qui s'exécutent sur l'infrastructure provisionnée par une version mineure précédente du modèle.

- Versions principales : modifications apportées au modèle qui ne sont peut-être pas rétrocompatibles. Ces modifications nécessitent généralement de nouvelles contributions de la part du développeur et impliquent souvent des modifications du schéma du modèle.

  Vous pouvez parfois choisir de désigner une modification rétrocompatible comme version majeure en fonction du modèle opérationnel de votre équipe.

La méthode utilisée pour AWS Proton déterminer si une demande de version de modèle concerne une version mineure ou majeure dépend de la manière dont les modifications du modèle sont suivies :

- Lorsque vous demandez explicitement la création d'une nouvelle version de modèle, vous demandez une version majeure en spécifiant un numéro de version principale, et vous demandez une version secondaire en ne spécifiant pas de numéro de version principal.
- Lorsque vous utilisez la synchronisation de modèles (et que vous ne faites donc pas de demandes de version de modèle explicites), AWS Proton tente de créer de nouvelles versions mineures pour les modifications de modèle qui se produisent dans le fichier YAML existant. AWS Proton crée une version majeure lorsque vous créez un nouveau répertoire pour le nouveau changement de modèle (par exemple, passez de la v1 à la v2).

> **Note**
>
> L'enregistrement d'une nouvelle version mineure basé sur la synchronisation des modèles échoue toujours s'il est AWS Proton déterminé que la modification n'est pas rétrocompatible.

Lorsque vous publiez une nouvelle version d'un modèle, celle-ci devient la version recommandée s'il s'agit de la version majeure et secondaire la plus récente. Les nouvelles AWS Proton ressources sont créées à l'aide de la nouvelle version recommandée et AWS Proton invite les administrateurs à utiliser la nouvelle version et à mettre à jour les AWS Proton ressources existantes qui utilisent une version obsolète.

Enregistrez et publiez des modèles

Vous pouvez enregistrer et publier des modèles d'environnement et de service avec AWS Proton, comme décrit dans les sections suivantes.

Vous pouvez créer une nouvelle version d'un modèle à l'aide de la console ou AWS CLI.

Vous pouvez également utiliser la console ou AWS CLI créer un modèle et configurer une synchronisation de modèles pour celui-ci. Cette configuration permet la AWS Proton synchronisation à partir de ensembles de modèles situés dans des référentiels git enregistrés que vous avez définis. Chaque fois qu'un commit est envoyé à votre dépôt et modifie l'un de vos ensembles de modèles, une nouvelle version mineure ou majeure de votre modèle est créée, si cette version n'existe pas déjà. Pour en savoir plus sur les prérequis et les exigences de configuration pour la synchronisation des modèles, consultezConfigurations de synchronisation de modèles.

Enregistrer et publier des modèles d'environnement

Vous pouvez enregistrer et publier les types de modèles d'environnement suivants.

- Enregistrez et publiez un modèle d'environnement standard AWS Proton utilisé pour déployer et gérer l'infrastructure de l'environnement.
- Enregistrez et publiez un modèle d'environnement géré par le client AWS Proton qui permet de se connecter à l'infrastructure provisionnée existante que vous gérez. AWS Proton ne gère pas votre infrastructure provisionnée existante.

> **Important**
>
> En tant qu'administrateur, assurez-vous que votre infrastructure provisionnée et gérée ainsi que tous les paramètres de sortie sont compatibles avec les modèles d'environnement gérés par le client associés. AWS Proton Je ne peux pas comptabiliser les modifications en votre nom, car elles ne sont pas visibles par AWS Proton. Les incohérences se traduisent par des échecs.

Vous pouvez utiliser la console ou le AWS CLI pour enregistrer et publier un modèle d'environnement.

AWS Management Console

Utilisez la console pour enregistrer et publier un nouveau modèle d'environnement.

1. Dans la AWS Proton console, choisissez Modèles d'environnement.
2. Choisissez Créer un modèle d'environnement.
3. Sur la page Créer un modèle d'environnement, dans la section Options du modèle, choisissez l'une des deux options de modèle disponibles.
   - Créez un modèle pour le provisionnement de nouveaux environnements.
   - Créez un modèle pour utiliser l'infrastructure provisionnée que vous gérez.
4. Si vous avez choisi Créer un modèle pour le provisionnement de nouveaux environnements, dans la section Source du bundle de modèles, choisissez l'une des trois options de source du bundle de modèles disponibles. Pour en savoir plus sur les exigences et les prérequis relatifs à la synchronisation des modèles, consultez. Configurations de synchronisation de modèles
   - Utilisez l'un de nos exemples de packs de modèles.
   - Utilisez votre propre ensemble de modèles.
   - Synchronisez les modèles depuis Git.
5. Indiquez le chemin d'accès à un ensemble de modèles.
   a. Si vous avez choisi Utiliser l'un de nos exemples de packs de modèles :

      Dans la section Ensemble de modèles d'échantillons, sélectionnez un ensemble d'exemples de modèles.

b. Si vous avez choisi Sync templates depuis Git, dans la section Code source :

   i. Sélectionnez le référentiel pour la configuration de synchronisation de votre modèle.

   ii. Entrez le nom de la branche du référentiel à partir de laquelle effectuer la synchronisation.

   iii. (Facultatif) Entrez le nom d'un répertoire pour limiter la recherche de votre ensemble de modèles.

c. Sinon, dans la section Emplacement du bundle S3, indiquez le chemin d'accès à votre bundle de modèles.

6. Dans la section Détails du modèle.

   a. Saisissez un Template name (Nom de modèle).

   b. (Facultatif) Entrez un nom d'affichage du modèle.

   c. (Facultatif) Entrez une description du modèle pour le modèle d'environnement.

7. (Facultatif) Cochez la case Personnaliser les paramètres de chiffrement (avancés) dans la section Paramètres de chiffrement pour fournir votre propre clé de chiffrement.

8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.

9. Choisissez Créer un modèle d'environnement.

   Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de balises gérées par le client AWS et une liste de balises gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des AWS Proton ressources. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

10. Le statut d'un nouveau modèle d'environnement commence à l'état Brouillon. Vous et les autres personnes `proton:CreateEnvironment` autorisées pouvez le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.

11. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans l'alerte d'information et ignorer l'étape suivante.

12. Dans la section Versions du modèle, choisissez Publier.

13. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.

14. Dans le volet de navigation, sélectionnez Modèles d'environnement pour afficher la liste de vos modèles d'environnement et leurs détails.

Utilisez la console pour enregistrer les nouvelles versions majeures et mineures d'un modèle d'environnement.

Pour de plus amples informations, veuillez consulter Modèles versionnés.

1. Dans la AWS Proton console, sélectionnez Modèles d'environnement.
2. Dans la liste des modèles d'environnement, choisissez le nom du modèle d'environnement pour lequel vous souhaitez créer une version majeure ou secondaire.
3. Dans la vue détaillée du modèle d'environnement, choisissez Créer une nouvelle version dans la section Versions du modèle.
4. Sur la page Créer une nouvelle version de modèle d'environnement, dans la section Source du bundle de modèles, choisissez l'une des deux options de source du bundle de modèles disponibles.
   - Utilisez l'un de nos exemples de packs de modèles.
   - Utilisez votre propre ensemble de modèles.
5. Fournissez un chemin d'accès au bundle de modèles sélectionné.
   - Si vous avez choisi Utiliser l'un de nos ensembles d'exemples de modèles, dans la section Ensemble d'exemples de modèles, sélectionnez un ensemble d'exemples de modèles.
   - Si vous avez choisi Utiliser votre propre ensemble de modèles, dans la section Emplacement du bundle S3, choisissez le chemin d'accès à votre bundle de modèles.
6. Dans la section Détails du modèle.
   a. (Facultatif) Entrez un nom d'affichage du modèle.
   b. (Facultatif) Entrez une description du modèle pour le modèle de service.
7. Dans la section Détails du modèle, choisissez l'une des options suivantes.
   - Pour créer une version secondaire, laissez la case Cocher pour créer une nouvelle version majeure vide.
   - Pour créer une version majeure, cochez la case Cocher pour créer une nouvelle version majeure.

8. Suivez les étapes de la console pour créer la nouvelle version mineure ou majeure, puis choisissez Créer une nouvelle version.

AWS CLI

Utilisez la CLI pour enregistrer et publier un nouveau modèle d'environnement, comme indiqué dans les étapes suivantes.

1. Créez un modèle d'environnement standard OU géré par le client en spécifiant la région, le nom, le nom d'affichage (facultatif) et la description (facultatif).

   a. Créez un modèle d'environnement standard.

      Exécutez la commande suivante :

      ```
      $ aws proton create-environment-template \
          --name "simple-env" \
          --display-name "Fargate" \
          --description "VPC with public access"
      ```

      Réponse :

      ```
      {
          "environmentTemplate": {
              "arn": "arn:aws:proton:region-id:123456789012:environment-template/
      simple-env",
              "createdAt": "2020-11-11T23:02:45.336000+00:00",
              "description": "VPC with public access",
              "displayName": "VPC",
              "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
              "name": "simple-env"
          }
      }
      ```

   b. Créez un modèle d'environnement géré par le client en ajoutant le `provisioning` paramètre avec une valeur`CUSTOMER_MANAGED`.

      Exécutez la commande suivante :

      ```
      $ aws proton create-environment-template \
          --name "simple-env" \
      ```

```
    --display-name "Fargate" \
    --description "VPC with public access" \
    --provisioning "CUSTOMER_MANAGED"
```

Réponse :

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env",
        "provisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Création d'une version mineure 0 de la version majeure 1 du modèle d'environnement

   Cette étape et les étapes suivantes sont les mêmes pour les modèles d'environnement standard et gérés par le client.

   Incluez le nom du modèle, la version principale, ainsi que le nom du compartiment S3 et la clé du compartiment contenant votre ensemble de modèles d'environnement.

   Exécutez la commande suivante :

```
$ aws proton create-environment-template-version \
    --template-name "simple-env" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}"
```

   Réponse :

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
```

```
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "simple-env"
    }
}
```

3. Utilisez la commande get pour vérifier l'état des inscriptions.

   Exécutez la commande suivante :

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

   Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

4. Publication de la version mineure 0 de la version principale 1 du modèle d'environnement en fournissant le nom du modèle et les versions principale et mineure. Cette version est la `Recommended` version.

   Exécutez la commande suivante :

```
$ aws proton update-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Réponse :

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Après avoir créé un nouveau modèle à l'aide du AWS CLI, vous pouvez consulter une liste AWS de balises gérées par le client. AWS Proton génère automatiquement des tags AWS gérés pour

vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

Exécutez la commande suivante :

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment-
template/simple-env"
```

Enregistrez et publiez des modèles de services

Lorsque vous créez une version de modèle de service, vous spécifiez une liste de modèles d'environnement compatibles. Ainsi, lorsque les développeurs sélectionnent un modèle de service, ils peuvent choisir l'environnement dans lequel ils souhaitent déployer leur service.

Avant de créer un service à partir d'un modèle de service ou avant de publier un modèle de service, vérifiez que les environnements sont déployés à partir des modèles d'environnement compatibles répertoriés.

Vous ne pouvez pas mettre à jour un service vers la nouvelle version majeure s'il est déployé dans un environnement créé à partir d'un modèle d'environnement compatible supprimé.

Pour ajouter ou supprimer des modèles d'environnement compatibles pour une version de modèle de service, vous devez créer une nouvelle version majeure de celui-ci.

Vous pouvez utiliser la console ou le AWS CLI pour enregistrer et publier un modèle de service.

AWS Management Console

Utilisez la console pour enregistrer et publier un nouveau modèle de service.

1. Dans la AWS Proton console, choisissez Modèles de services.
2. Choisissez Créer un modèle de service.
3. Sur la page Créer un modèle de service, dans la section Source du bundle de modèles, choisissez l'une des options de modèle disponibles.
   - Utilisez votre propre ensemble de modèles.
   - Synchronisez les modèles depuis Git.

4. Indiquez le chemin d'accès à un ensemble de modèles.

   a. Si vous avez choisi Sync templates depuis Git, dans la section Référentiel du code source :

      i. Sélectionnez le référentiel pour la configuration de synchronisation de votre modèle.

      ii. Entrez le nom de la branche du référentiel à partir de laquelle effectuer la synchronisation.

      iii. (Facultatif) Entrez le nom d'un répertoire pour limiter la recherche de votre ensemble de modèles.

   b. Sinon, dans la section Emplacement du bundle S3, indiquez le chemin d'accès à votre bundle de modèles.

5. Dans la section Détails du modèle.

   a. Saisissez un Template name (Nom de modèle).

   b. (Facultatif) Entrez un nom d'affichage du modèle.

   c. (Facultatif) Entrez une description du modèle pour le modèle de service.

6. Dans la section Modèles d'environnement compatibles, choisissez parmi la liste des modèles d'environnement compatibles.

7. (Facultatif) Dans la section Paramètres de chiffrement, choisissez Personnaliser les paramètres de chiffrement (avancés) pour fournir votre propre clé de chiffrement.

8. (Facultatif) Dans la section Pipeline :

   Si vous n'incluez pas de définition de pipeline de service dans votre modèle de service, décochez la case Pipeline - facultatif au bas de la page. Vous ne pouvez pas le modifier une fois le modèle de service créé. Pour de plus amples informations, veuillez consulter Packs de modèles.

9. (Facultatif) Dans la section Sources de composants prises en charge, pour Sources de composants, choisissez Directement défini pour permettre l'attachement de composants directement définis à vos instances de service.

10. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.

11. Choisissez Créer un modèle de service.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle de service. Ces informations incluent une liste de balises gérées par le client AWS et une liste de balises gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des AWS Proton ressources. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

12. Le statut d'un nouveau modèle de service commence à l'état Brouillon. Vous et les autres personnes `proton:CreateService` autorisées pouvez le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.
13. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans l'alerte d'information et ignorer l'étape suivante.
14. Dans la section Versions du modèle, choisissez Publier.
15. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.
16. Dans le volet de navigation, sélectionnez Modèles de services pour afficher la liste de vos modèles de service et leurs détails.

Utilisez la console pour enregistrer les nouvelles versions majeures et mineures d'un modèle de service.

Pour de plus amples informations, veuillez consulter Modèles versionnés.

1. Dans la AWS Proton console, choisissez Service Templates.
2. Dans la liste des modèles de service, choisissez le nom du modèle de service pour lequel vous souhaitez créer une version majeure ou secondaire.
3. Dans la vue détaillée du modèle de service, choisissez Créer une nouvelle version dans la section Versions du modèle.
4. Sur la page Créer une nouvelle version de modèle de service, dans la section Source du bundle, sélectionnez Utiliser votre propre bundle de modèles.
5. Dans la section Emplacement du bundle S3, choisissez le chemin d'accès à votre bundle de modèles.
6. Dans la section Détails du modèle.

    a. (Facultatif) Entrez un nom d'affichage du modèle.

b. (Facultatif) Entrez une description du modèle pour le modèle de service.

7. Dans la section Détails du modèle, choisissez l'une des options suivantes.

   - Pour créer une version secondaire, laissez la case Cocher pour créer une nouvelle version majeure vide.
   - Pour créer une version majeure, cochez la case Cocher pour créer une nouvelle version majeure.

8. Suivez les étapes de la console pour créer la nouvelle version mineure ou majeure, puis choisissez Créer une nouvelle version.

AWS CLI

Pour créer un modèle de service qui déploie un service sans pipeline de services, ajoutez le paramètre et la valeur `--pipeline-provisioning "CUSTOMER_MANAGED"` à la `create-service-template` commande. Configurez vos ensembles de modèles comme décrit dans les sections Packs de modèles Création etExigences relatives au schéma pour les ensembles de modèles de services.

> **Note**
>
> Vous ne pouvez pas le modifier une `pipelineProvisioning` fois le modèle de service créé.

1. Utilisez la CLI pour enregistrer et publier un nouveau modèle de service, avec ou sans pipeline de services, comme indiqué dans les étapes suivantes.

   a. Créez un modèle de service avec un pipeline de services à l'aide de la CLI.

      Spécifiez le nom, le nom d'affichage (facultatif) et la description (facultatif).

      Exécutez la commande suivante :

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service"
```

Réponse :

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service"
    }
}
```

b. Créez un modèle de service sans pipeline de services.

Addition `--pipeline-provisioning`.

Exécutez la commande suivante :

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service" \
    --pipeline-provisioning "CUSTOMER_MANAGED"
```

Réponse :

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Créez une version mineure 0 de la version principale 1 du modèle de service.

   Incluez le nom du modèle, les modèles d'environnement compatibles, la version principale, ainsi que le nom du compartiment S3 et la clé du compartiment contenant votre ensemble de modèles de services.

   Exécutez la commande suivante :

```
$ aws proton create-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}" \
    --compatible-environment-templates '[{"templateName":"simple-
env","majorVersion":"1"}]'
```

   Réponse :

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

3. Utilisez la commande get pour vérifier l'état des inscriptions.

   Exécutez la commande suivante :

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Réponse :


```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
```

```
}
```

4. Publiez le modèle de service en utilisant la commande de mise à jour pour changer le statut en`"PUBLISHED"`.

   Exécutez la commande suivante :

```
$ aws proton update-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Réponse :

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello pipeline
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
```

```
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

5. Vérifiez que la version 1.0 AWS Proton a été publiée en utilisant la commande get pour récupérer les données détaillées du modèle de service.

   Exécutez la commande suivante :

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   Réponse :

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:03:04.767000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
   type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
```

```
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

Afficher les données du modèle

Vous pouvez afficher des listes de modèles avec des détails et afficher des modèles individuels avec des données détaillées à l'aide de la AWS Proton console et AWS CLI.

Les données du modèle d'environnement géré par le client incluent le `provisioned` paramètre et la valeur`CUSTOMER_MANAGED`.

Si un modèle de service n'inclut pas de pipeline de services, les données du modèle de service incluent le `pipelineProvisioning` paramètre avec la valeur`CUSTOMER_MANAGED`.

Pour de plus amples informations, veuillez consulter Enregistrez et publiez des modèles.

Vous pouvez utiliser la console ou le AWS CLI pour répertorier et afficher les données du modèle.

AWS Management Console

Utilisez la console pour répertorier et afficher les modèles.

1. Pour afficher la liste des modèles, choisissez des modèles (environnement ou service).
2. Pour afficher les données détaillées, choisissez le nom d'un modèle.

   Affichez les données détaillées du modèle, une liste des versions principales et secondaires du modèle, une liste des AWS Proton ressources déployées à l'aide des versions du modèle et des balises du modèle.

La version majeure et la version mineure recommandées sont étiquetées comme recommandées.

AWS CLI

Utilisez les modèles AWS CLI de liste et d'affichage.

Exécutez la commande suivante :

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Réponse :

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-10T18:35:08.293000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-10T18:35:11.162000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Exécutez la commande suivante :

```
$ aws proton list-environment-templates
```

Réponse :

```
{
    "templates": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-3",
            "createdAt": "2020-11-10T18:35:05.763000+00:00",
            "description": "VPC with Public Access",
            "displayName": "VPC",
            "lastModifiedAt": "2020-11-10T18:35:05.763000+00:00",
            "name": "simple-env-3",
            "recommendedVersion": "1.0"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-1",
            "createdAt": "2020-11-10T00:14:06.881000+00:00",
            "description": "Some SSM Parameters",
            "displayName": "simple-env-1",
            "lastModifiedAt": "2020-11-10T00:14:06.881000+00:00",
            "name": "simple-env-1",
            "recommendedVersion": "1.0"
        }
    ]
}
```

Afficher une version mineure d'un modèle de service.

Exécutez la commande suivante :

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Réponse :

```
{
    "serviceTemplateMinorVersion": {
```

```
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
  my_sample_pipeline_required_input:\n          type: string\n          description:
 \"Another sample input\"\n\n    MyServiceInstanceInputType:\n      type: object
\n      description: \"Service instance input properties\"\n      required:\n
       - my_sample_service_instance_required_input\n      properties:\n
 my_sample_service_instance_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
     my_sample_service_instance_required_input:\n          type: string\n
 description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```


Affichez un modèle de service sans pipeline de services, comme indiqué dans l'exemple de commande et de réponse suivant.

Exécutez la commande suivante :

```
$ aws proton get-service-template \
    --name "simple-svc-template-cli"
```

Réponse :

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/simple-svc-
template-cli",
        "createdAt": "2021-02-18T15:38:57.949000+00:00",
        "displayName": "simple-svc-template-cli",
        "lastModifiedAt": "2021-02-18T15:38:57.949000+00:00",
        "status": "DRAFT",
        "name": "simple-svc-template-cli",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

Mettre à jour un modèle

Vous pouvez mettre à jour un modèle comme décrit dans la liste suivante.

- Modifiez le `description` ou `display name` d'un modèle lorsque vous utilisez la console ou AWS CLI. Vous ne pouvez pas modifier un modèle. `name`
- Mettez à jour le statut d'une version mineure du modèle lorsque vous utilisez la console ou AWS CLI. Vous ne pouvez modifier le statut que de `DRAFT` à`PUBLISHED`.
- Modifiez le nom d'affichage et la description d'une version secondaire ou majeure d'un modèle lorsque vous utilisez le AWS CLI.

AWS Management Console

Modifiez la description et le nom d'affichage d'un modèle à l'aide de la console, comme décrit dans les étapes suivantes.

Dans la liste des modèles.

1. Dans la AWS Proton console, choisissez Modèles (d'environnement ou de service).
2. Dans la liste des modèles, cliquez sur le bouton radio situé à gauche du modèle dont vous souhaitez mettre à jour la description ou le nom d'affichage.
3. Choisissez Actions, puis Modifier.

4. Sur la page Modifier le modèle (environnement ou service), dans la section Détails du modèle, entrez vos modifications dans le formulaire et choisissez Enregistrer les modifications.

Modifiez le statut d'une version secondaire d'un modèle à l'aide de la console pour publier un modèle, comme décrit ci-dessous. Vous ne pouvez modifier le statut que de `DRAFT` à`PUBLISHED`.

Dans la page détaillée du modèle (environnement ou service).

1. Dans la AWS Proton console, choisissez des modèles (environnement ou service).
2. Dans la liste des modèles, choisissez le nom du modèle dont vous souhaitez mettre à jour le statut d'une version mineure de Brouillon à Publié.
3. Sur la page détaillée du modèle (environnement ou service), dans la section Versions du modèle, sélectionnez le bouton radio situé à gauche de la version secondaire que vous souhaitez publier.
4. Choisissez Publier dans la section Versions du modèle. Le statut passe de Brouillon à Publié.

AWS CLI

L'exemple de commande et de réponse suivant montre comment modifier la description d'un modèle d'environnement.

Exécutez la commande suivante.

```
$ aws proton update-environment-template \
    --name "simple-env" \
    --description "A single VPC with public access"
```

Réponse :

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-28T22:02:10.651000+00:00",
        "description": "A single VPC with public access",
        "displayName": "simple-env",
        "lastModifiedAt": "2020-11-29T16:11:18.956000+00:00",
```

```
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```


Vous pouvez également utiliser le AWS CLI pour mettre à jour les modèles de service. Enregistrez et publiez des modèles de servicesReportez-vous à l'étape 5 pour un exemple de mise à jour du statut d'une version mineure d'un modèle de service.

Supprimer des modèles

Les modèles peuvent être supprimés à l'aide de la console et AWS CLI.

Vous pouvez supprimer une version mineure d'un modèle d'environnement si aucun environnement n'est déployé sur cette version.

Vous pouvez supprimer une version mineure d'un modèle de service si aucune instance de service ou aucun pipeline n'est déployé sur cette version. Votre pipeline peut être déployé dans une version de modèle différente de celle de votre instance de service. Par exemple, si votre instance de service est mise à jour vers la version 1.1 à partir de la version 1.0 et que votre pipeline est toujours déployé vers la version 1.0, vous ne pouvez pas supprimer le modèle de service 1.0.

AWS Management Console

Vous pouvez utiliser la console pour supprimer l'intégralité du modèle ou des versions mineures et majeures individuelles d'un modèle.

Utilisez la console pour supprimer les modèles comme suit.

> **Note**
>
> Lorsque vous utilisez la console pour supprimer des modèles.
>
> - Lorsque vous supprimez le modèle dans son intégralité, vous supprimez également les versions principales et secondaires du modèle.

Dans la liste des modèles (d'environnement ou de service).

1. Dans la AWS Proton console, choisissez Modèles (d'environnement ou de service).
2. Dans la liste des modèles, sélectionnez le bouton radio situé à gauche du modèle que vous souhaitez supprimer.

   Vous ne pouvez supprimer un modèle complet que si aucune AWS Proton ressource n'est déployée sur ses versions.
3. Choisissez Actions, puis Supprimer pour supprimer le modèle dans son intégralité.
4. Un modal vous invite à confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

Dans la page détaillée du modèle (environnement ou service).

1. Dans la AWS Proton console, choisissez Modèles (d'environnement ou de service).
2. Dans la liste des modèles, choisissez le nom du modèle que vous souhaitez supprimer entièrement ou supprimez des versions majeures ou secondaires individuelles de celui-ci.
3. Pour supprimer le modèle dans son intégralité.

   Vous ne pouvez supprimer un modèle complet que si aucune AWS Proton ressource n'est déployée sur ses versions.

   a. Choisissez Supprimer dans le coin supérieur droit de la page.

   b. Un modal vous invite à confirmer l'action de suppression.

   c. Suivez les instructions et choisissez Oui, supprimer.

4. Pour supprimer les versions principales ou secondaires d'un modèle.

   Vous ne pouvez supprimer une version mineure d'un modèle que si aucune AWS Proton ressource n'est déployée sur cette version.

   a. Dans la section Versions du modèle, sélectionnez le bouton radio situé à gauche de la version que vous souhaitez supprimer.
   b. Choisissez Supprimer dans la section Versions du modèle.
   c. Un modal vous invite à confirmer l'action de suppression.
   d. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

AWS CLI les opérations de suppression d'un modèle n'incluent pas la suppression d'autres versions d'un modèle. Lorsque vous utilisez le AWS CLI, supprimez les modèles répondant aux conditions suivantes.

- Supprimez un modèle complet s'il n'existe aucune version secondaire ou majeure du modèle.
- Supprimez une version majeure lorsque vous supprimez la dernière version mineure restante.
- Supprimez une version secondaire d'un modèle si aucune AWS Proton ressource n'est déployée sur cette version.
- Supprimez la version secondaire recommandée d'un modèle s'il n'existe aucune autre version mineure du modèle et si aucune AWS Proton ressource n'est déployée sur cette version.

Les exemples de commandes et de réponses suivants montrent comment utiliser le AWS CLI pour supprimer des modèles.

Exécutez la commande suivante :

```
$ aws proton delete-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Réponse :

```
{
```

```
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Exécutez la commande suivante :

```
$ aws proton delete-environment-template \
    --name "simple-env"
```

Réponse :

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with Public Access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-12T00:23:22.339000+00:00",
        "name": "simple-env",
        "recommendedVersion": "1.0"
    }
}
```

Exécutez la commande suivante :

```
$ aws proton delete-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Réponse :

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [{"majorVersion": "1", "templateName":
 "simple-env"}],
        "createdAt": "2020-11-28T22:07:05.798000+00:00",
        "lastModifiedAt": "2020-11-28T22:19:05.368000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

Configurations de synchronisation de modèles

Apprenez à configurer un modèle pour permettre la AWS Proton synchronisation à partir des ensembles de modèles situés dans les référentiels git enregistrés que vous définissez. Lorsqu'un commit est envoyé à votre dépôt, il AWS Proton vérifie si des modifications ont été apportées aux ensembles de modèles de votre dépôt. S'il détecte un changement de bundle de modèles, une nouvelle version mineure ou majeure de son modèle est créée, si cette version n'existe pas déjà. AWS Proton prend actuellement en charge GitHub, GitHub Enterprise et BitBucket.

Transférer un commit vers un ensemble de modèles synchronisés

Lorsque vous envoyez un commit à une branche qui est suivie par l'un de vos modèles, vous AWS Proton clonez votre dépôt et déterminez les modèles à synchroniser. Il analyse les fichiers de votre répertoire pour trouver les répertoires correspondant à la convention de`{template-name}/{major-version}/`.

Après avoir AWS Proton déterminé quels modèles et versions principales sont associés à votre dépôt et à votre branche, il commence à essayer de synchroniser tous ces modèles en parallèle.

Lors de chaque synchronisation avec un modèle particulier, vérifiez d' AWS Proton abord si le contenu du répertoire des modèles a changé depuis la dernière synchronisation réussie. Si le contenu n'a pas changé, AWS Proton ignore l'enregistrement d'un bundle dupliqué. Cela garantit

qu'une nouvelle version mineure du modèle est créée si le contenu du bundle de modèles change. Si le contenu du bundle de modèles a changé, le bundle est enregistré auprès de AWS Proton.

Une fois le bundle de modèles enregistré, AWS Proton surveille l'état de l'enregistrement jusqu'à ce que l'enregistrement soit terminé.

Une seule synchronisation peut avoir lieu entre une version mineure et une version majeure du modèle à la fois. Toutes les validations qui auraient pu être poussées alors qu'une synchronisation était en cours sont regroupées par lots. Les validations par lots sont synchronisées une fois la tentative de synchronisation précédente terminée.

Modèles de service de synchronisation

AWS Proton peut synchroniser les modèles d'environnement et de service à partir de votre dépôt git. Pour synchroniser vos modèles de services, vous ajoutez un fichier supplémentaire nommé `.template-registration.yaml` dans le répertoire de chaque version majeure de votre ensemble de modèles. Ce fichier contient des informations supplémentaires AWS Proton nécessaires lorsqu'il crée une version de modèle de service pour vous après un commit : environnements compatibles et sources de composants prises en charge.

Le chemin complet du fichier est*`service-template-name`*`/`*`major-version`*`/.template-registration.yaml`. Pour de plus amples informations, veuillez consulter the section called “Modèles de service de synchronisation”.

Considérations relatives à la configuration de la synchronisation

Passez en revue les considérations suivantes concernant l'utilisation des configurations de synchronisation de modèles.

- Les référentiels ne doivent pas dépasser 250 Mo.
- Pour configurer la synchronisation des modèles, associez d'abord le référentiel à AWS Proton. Pour de plus amples informations, veuillez consulter the section called “Création d'un lien de référentiel”.
- Lorsqu'une nouvelle version de modèle est créée à partir d'un modèle synchronisé, elle est dans l'`DRAFT`état actuel.
- Une nouvelle version mineure d'un modèle est créée si l'une des conditions suivantes est vraie :
  - Le contenu du bundle de modèles est différent de celui de la dernière version mineure du modèle synchronisé.

  - La dernière version mineure du modèle précédemment synchronisée a été supprimée.
- La synchronisation ne peut pas être interrompue.
- Les nouvelles versions mineures ou majeures sont automatiquement synchronisées.
- Les nouveaux modèles de haut niveau ne peuvent pas être créés par des configurations de synchronisation de modèles.
- Vous ne pouvez pas effectuer de synchronisation avec un modèle à partir de plusieurs référentiels avec une configuration de synchronisation de modèles.
- Vous ne pouvez pas utiliser de balises à la place de branches.
- Lorsque vous créez un modèle de service, vous spécifiez des modèles d'environnement compatibles.
- Vous pouvez créer un modèle d'environnement et l'ajouter en tant qu'environnement compatible pour votre modèle de service dans le même commit.
- Les synchronisations avec une seule version majeure du modèle sont exécutées une par une. Au cours d'une synchronisation, si de nouveaux commits sont détectés, ils sont regroupés par lots et appliqués à la fin de la synchronisation active. Les synchronisations avec les différentes versions majeures du modèle se font en parallèle.
- Si vous modifiez la branche à partir de laquelle vos modèles sont synchronisés, toutes les synchronisations en cours depuis l'ancienne branche sont d'abord terminées. La synchronisation commence ensuite à partir de la nouvelle branche.
- Si vous modifiez le référentiel à partir duquel vos modèles sont synchronisés, toute synchronisation en cours depuis l'ancien référentiel risque d'échouer ou de s'exécuter jusqu'à la fin. Cela dépend de l'étape de synchronisation dans laquelle ils se trouvent.

Pour plus d'informations, consultez le manuel The AWS Proton Service API Reference.

Rubriques

- Création d'une configuration de synchronisation de modèles
- Afficher les détails de configuration de synchronisation des modèles
- Modifier la configuration de synchronisation d'un modèle
- Supprimer un modèle de configuration de synchronisation

Création d'une configuration de synchronisation de modèles

Découvrez comment créer un modèle de configuration de synchronisation avec AWS Proton.

Créez un modèle de configuration prérequis pour la synchronisation :

- Vous avez lié un dépôt à AWS Proton.
- Un ensemble de modèles se trouve dans votre référentiel.

Le lien vers le référentiel se compose des éléments suivants :

- Une CodeConnections connexion qui donne AWS Proton l'autorisation d'accéder à votre dépôt et de vous abonner à ses notifications.
- Un rôle lié à un service. Lorsque vous liez votre référentiel, le rôle lié au service est créé pour vous.

Avant de créer votre première configuration de synchronisation de modèles, transférez un ensemble de modèles vers votre référentiel, comme indiqué dans le schéma de répertoire suivant.

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
/templates/my-env-template/v1/schema/
```

Une fois que vous avez créé votre première configuration de synchronisation de modèles, de nouvelles versions de modèles sont automatiquement créées lorsque vous envoyez un commit qui ajoute un ensemble de modèles mis à jour dans une nouvelle version (par exemple, sous `/my-env-template/v2/`).

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

Vous pouvez inclure de nouvelles versions de bundle de modèles pour un ou plusieurs modèles configurés pour la synchronisation dans un seul commit. AWS Proton crée une nouvelle version de modèle pour chaque nouvelle version de bundle de modèles incluse dans le commit.

Après avoir créé la configuration de synchronisation des modèles, vous pouvez toujours créer manuellement de nouvelles versions du modèle dans la console ou en AWS CLI téléchargeant des ensembles de modèles depuis un compartiment S3. La synchronisation des modèles ne fonctionne que dans un seul sens : de votre dépôt vers AWS Proton. Les versions de modèles créées manuellement ne sont pas synchronisées.

Après avoir configuré une configuration de synchronisation des modèles, AWS Proton écoute les modifications apportées à votre référentiel. Chaque fois qu'une modification est poussée, elle recherche tout répertoire portant le même nom que votre modèle. Il recherche ensuite dans ce répertoire tous les répertoires qui ressemblent à des versions majeures. AWS Proton enregistre le bundle de modèles dans la version principale du modèle correspondant. Les nouvelles versions sont toujours en bon `DRAFT` état. Vous pouvez publier les nouvelles versions avec la console ou AWS CLI.

Supposons, par exemple, que vous ayez un modèle appelé `my-env-template` configuré pour être synchronisé à partir d'`my-repo/templates`une branche `main` avec la mise en page suivante.

```
/code
/code/service.go
README.md
/templates/
/templates/my-env-template/
/templates/my-env-template/v1/
/templates/my-env-template/v1/infrastructure/
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

AWS Proton synchronise le contenu de `/templates/my-env-template/v1/` to `my-env-template:1` et le contenu de `/templates/my-env-template/v2/` to`my-env-template:2`. S'ils n'existent pas déjà, il crée ces versions majeures.

AWS Proton a trouvé le premier répertoire correspondant au nom du modèle. Vous pouvez limiter les AWS Proton recherches dans les annuaires en spécifiant une configuration de synchronisation de modèles `subdirectoryPath` lorsque vous créez ou modifiez un modèle. Par exemple, vous pouvez spécifier `/production-templates/` pour`subdirectoryPath`.

Vous pouvez créer un modèle de configuration de synchronisation à l'aide de la console ou de la CLI.

AWS Management Console

Créez un modèle et une configuration de synchronisation de modèles à l'aide de la console.

1. Dans la AWS Proton console, choisissez Modèles d'environnement.
2. Choisissez Créer un modèle d'environnement.
3. Sur la page Créer un modèle d'environnement, dans la section Options du modèle, choisissez Créer un modèle pour le provisionnement de nouveaux environnements.
4. Dans la section Source du bundle de modèles, choisissez Synchroniser les modèles depuis Git.
5. Dans la section Référentiel du code source :
   a. Pour Repository, sélectionnez le référentiel lié qui contient votre bundle de modèles.
   b. Pour Branch, sélectionnez une branche du référentiel à partir de laquelle effectuer la synchronisation.
   c. (Facultatif) Dans le répertoire des ensembles de modèles, entrez le nom d'un répertoire pour affiner la recherche de votre ensemble de modèles.
6. Dans la section Détails du modèle.
   a. Saisissez un Template name (Nom de modèle).
   b. (Facultatif) Entrez un nom d'affichage du modèle.
   c. (Facultatif) Entrez une description du modèle pour le modèle d'environnement.
7. (Facultatif) Cochez la case Personnaliser les paramètres de chiffrement (avancés) dans la section Paramètres de chiffrement pour fournir votre propre clé de chiffrement.
8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez Créer un modèle d'environnement.

   Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de balises AWS gérées et gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des AWS Proton ressources. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

10. Sur la page détaillée du modèle, choisissez l'onglet Synchronisation pour afficher les données détaillées de configuration de synchronisation du modèle.
11. Cliquez sur l'onglet Versions du modèle pour afficher les versions du modèle avec les détails du statut.
12. Le statut d'un nouveau modèle d'environnement commence à l'état Brouillon. Vous et les autres personnes `proton:CreateEnvironment` autorisées pouvez le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.
13. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans l'alerte d'information et ignorer l'étape suivante.
14. Dans la section Versions du modèle, choisissez Publier.
15. Le statut du modèle passe à Publié. Il s'agit de la version la plus récente et recommandée du modèle.
16. Dans le volet de navigation, sélectionnez Modèles d'environnement pour afficher la liste de vos modèles d'environnement et leurs détails.

La procédure de création d'un modèle de service et d'une configuration de synchronisation de modèles est similaire.

AWS CLI

Créez un modèle et une configuration de synchronisation de modèles à l'aide du AWS CLI.

1. Créez un modèle. Dans cet exemple, un modèle d'environnement est créé.

   Exécutez la commande suivante.

   ```
   $ aws proton create-environment-template \
       --name "env-template"
   ```

   La réponse est la suivante.

   ```
   {
       "environmentTemplate": {
           "arn": "arn:aws:proton:us-east-1:123456789012:environment-template/env-template",
           "createdAt": "2021-11-07T23:32:43.045000+00:00",
           "displayName": "env-template",
   ```

```
        "lastModifiedAt": "2021-11-07T23:32:43.045000+00:00",
        "name": "env-template",
        "status": "DRAFT",
        "templateName": "env-template"
    }
}
```

2. Créez la configuration de synchronisation de votre modèle avec AWS CLI en fournissant les éléments suivants :

   - Le modèle avec lequel vous souhaitez effectuer la synchronisation. Après avoir créé la configuration de synchronisation du modèle, vous pouvez toujours créer de nouvelles versions à partir de celui-ci manuellement dans la console ou à l'aide du AWS CLI.
   - Nom du modèle.
   - Type de modèle.
   - Le référentiel lié à partir duquel vous souhaitez effectuer la synchronisation.
   - Le fournisseur du référentiel lié.
   - Branche dans laquelle se trouve le bundle de modèles.
   - (Facultatif) Le chemin d'accès au répertoire contenant votre ensemble de modèles. Par défaut, AWS Proton recherche le premier répertoire correspondant au nom de votre modèle.

   Exécutez la commande suivante.

```
$ aws proton create-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-name "myrepos/templates" \
    --repository-provider "GITHUB" \
    --branch "main" \
    --subdirectory "env-template/"
```

   La réponse est la suivante.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
```

```
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

3. Pour publier la version de votre modèle, consultezEnregistrez et publiez des modèles.

Modèles de service de synchronisation

Les exemples précédents montrent comment synchroniser les modèles d'environnement. Les modèles de service sont similaires. Pour synchroniser les modèles de services, vous ajoutez un fichier supplémentaire nommé `.template-registration.yaml` dans le répertoire de chaque version majeure de votre ensemble de modèles. Ce fichier contient des informations supplémentaires AWS Proton nécessaires lorsqu'il crée une version de modèle de service pour vous à la suite d'un commit. Lorsque vous créez explicitement une version de modèle de service à l'aide de la AWS Proton console ou de l'API, vous fournissez ces informations en tant qu'entrées, et ce fichier remplace ces entrées pour la synchronisation des modèles.

```
./templates/                                                # subdirectory (optional)
 /templates/my-svc-template/                                # service template name
 /templates/my-svc-template/v1/                             # service template version
 /templates/my-svc-template/v1/.template-registration.yaml  # service template version
 properties
 /templates/my-svc-template/v1/instance_infrastructure/     # template bundle
 /templates/my-svc-template/v1/schema/
```

Le `.template-registration.yaml` fichier contient les informations suivantes :

- Environnements compatibles [obligatoire] — Les environnements basés sur ces modèles d'environnement et les versions majeures sont compatibles avec les services basés sur cette version de modèle de service.
- Sources de composants prises en charge [facultatif] — Les composants utilisant ces sources sont compatibles avec les services basés sur cette version de modèle de service. Si ce n'est pas spécifié, les composants ne peuvent pas être attachés à ces services. Pour plus d'informations sur les composants, consultezÉléments.

La syntaxe YAML du fichier est la suivante :

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
  - DIRECTLY_DEFINED
```

Spécifiez une ou plusieurs combinaisons de modèles d'environnement et de versions principales. `supported_component_sources`La spécification est facultative et la seule valeur prise en charge est`DIRECTLY_DEFINED`.

Example.template-registration.yaml

Dans cet exemple, la version du modèle de service est compatible avec les versions principales 1 et 2 du modèle d'`my-env-template`environnement. Il est également compatible avec les versions principales 1 et 3 du modèle d'`another-env-template`environnement. Le fichier ne le précise `supported_component_sources` pas. Les composants ne peuvent donc pas être attachés à des services basés sur cette version de modèle de service.

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
  - another-env-template:3
```

> **Note**
>
> Précédemment, AWS Proton défini un fichier différent`.compatible-envs`, pour spécifier les environnements compatibles. AWS Proton supporte toujours ce fichier et son format pour des raisons de rétrocompatibilité. Nous ne recommandons plus de l'utiliser, car il n'est pas extensible et ne peut pas prendre en charge de nouvelles fonctionnalités telles que les composants.

Afficher les détails de configuration de synchronisation des modèles

Affichez les données détaillées de configuration de synchronisation des modèles à l'aide de la console ou de la CLI.

AWS Management Console

Utilisez la console pour consulter les détails de configuration de synchronisation des modèles.

1. Dans le volet de navigation, choisissez des modèles (environnement ou service).
2. Pour afficher les données détaillées, choisissez le nom d'un modèle pour lequel vous avez créé une configuration de synchronisation de modèles.
3. Sur la page détaillée du modèle, sélectionnez l'onglet Synchronisation pour afficher les données détaillées de configuration de synchronisation du modèle.

AWS CLI

Utilisez le AWS CLI pour afficher un modèle synchronisé.

Exécutez la commande suivante.

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

La réponse est la suivante.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
        "templateType": "SERVICE"
    }
}
```

Utilisez le AWS CLI pour obtenir l'état de synchronisation des modèles.

Pour`template-version`, entrez la version principale du modèle.

Exécutez la commande suivante.

```
$ aws proton get-template-sync-status \
```

```
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

Modifier la configuration de synchronisation d'un modèle

Vous pouvez modifier tous les paramètres de configuration de synchronisation des modèles, à l'exception de `template-name` et`template-type`.

Apprenez à modifier la configuration d'un modèle de synchronisation à l'aide de la console ou de la CLI.

AWS Management Console

Modifiez une branche de configuration de synchronisation de modèles à l'aide de la console.

Dans la liste des modèles.

1. Dans la AWS Proton console, choisissez Modèles (d'environnement ou de service).
2. Dans la liste des modèles, choisissez le nom du modèle avec la configuration de synchronisation des modèles que vous souhaitez modifier.
3. Sur la page détaillée du modèle, choisissez l'onglet Synchronisation du modèle.
4. Dans la section Détails de synchronisation des modèles, choisissez Modifier.
5. Sur la page Modifier, dans la section Référentiel de code source, pour Branch, sélectionnez une branche, puis choisissez Enregistrer la configuration.

AWS CLI

Les exemples de commande et de réponse suivants montrent comment modifier un modèle de configuration de synchronisation à **`branch`** l'aide de la CLI.

Exécutez la commande suivante.

```
$ aws proton update-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-provider "GITHUB" \
    --repository-name "myrepos/templates" \
    --branch "fargate" \
```

```
    --subdirectory "env-template"
```

La réponse est la suivante.

```
{
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Vous pouvez également utiliser le AWS CLI pour mettre à jour les modèles de services synchronisés.

Supprimer un modèle de configuration de synchronisation

Supprimez un modèle de configuration de synchronisation à l'aide de la console ou de la CLI.

AWS Management Console

Supprimez un modèle de configuration de synchronisation à l'aide de la console.

1. Sur la page des détails du modèle, choisissez l'onglet Synchroniser.
2. Dans la section Détails de synchronisation, choisissez Déconnecter.

AWS CLI

Les exemples de commandes et de réponses suivants montrent comment utiliser le AWS CLI pour supprimer des configurations de modèles synchronisés.

Exécutez la commande suivante.

```
$ aws proton delete-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT"
```

La réponse est la suivante.

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Configurations de synchronisation des services

Grâce à la synchronisation des services, vous pouvez configurer et déployer vos AWS Proton services à l'aide de Git. Vous pouvez utiliser la synchronisation des services pour gérer les déploiements initiaux et les mises à jour de votre AWS Proton service avec une configuration définie dans un référentiel Git. Grâce à Git, vous pouvez utiliser des fonctionnalités telles que le suivi des versions et les pull requests pour configurer, gérer et déployer vos services. Service Sync combine AWS Proton Git pour vous aider à mettre en place une infrastructure standardisée définie et gérée via AWS Proton des modèles. Il gère les définitions de services dans votre dépôt Git et réduit le changement d'outil. Par rapport à l'utilisation de Git uniquement, la standardisation des modèles et du déploiement vous AWS Proton permet de passer moins de temps à gérer votre infrastructure. AWS Proton offre également une transparence et une auditabilité accrues tant pour les développeurs que pour les équipes de plateforme.

AWS Proton fichier OPS

Le `proton-ops` fichier définit où se AWS Proton trouve le fichier de spécifications utilisé pour mettre à jour votre instance de service. Il définit également l'ordre dans lequel les instances de service doivent être mises à jour et le moment où il convient de promouvoir les modifications d'une instance à l'autre.

Le `proton-ops` fichier prend en charge la synchronisation d'une instance de service à l'aide du fichier de spécifications, ou de plusieurs fichiers de spécifications, trouvés dans votre référentiel lié. Vous pouvez le faire en définissant un bloc de synchronisation dans le `proton-ops` fichier, comme dans l'exemple suivant.

Exemple. /configuration/proton-ops.yaml :

```
sync:
```

```
  services:
      frontend-svc:
          alpha:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          beta:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          gamma:
              branch: pre-prod
              spec: ./frontend-svc/pre-prod/frontend-spec.yaml
          prod-one:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-two:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-three:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

Dans l'exemple précédent, `frontend-svc` c'est le nom du service`alpha`, et`beta`,`gamma`,`prod-one`,`prod-two`,, `prod-three` sont les instances.

Le `spec` fichier peut être constitué de toutes les instances ou d'un sous-ensemble des instances définies dans le `proton-ops` fichier. Cependant, au minimum, l'instance doit être définie dans la branche et la spécification à partir de laquelle elle est synchronisée. Si les instances ne sont pas définies dans le `proton-ops` fichier, avec la branche et l'emplacement `spec` du fichier spécifiques, Service Sync ne créera ni ne mettra à jour ces instances.

Les exemples suivants montrent à quoi ressemblent les `spec` fichiers. N'oubliez pas que le `proton-ops` fichier est synchronisé à partir de ces `spec` fichiers.

Exemple **`./frontend-svc/test/frontend-spec.yaml`** :

```
proton: "ServiceSpec"
instances:
- name: "alpha"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
```

```
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "beta"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Exemple **`./frontend-svc/pre-prod/frontend-spec.yaml`** :

```
proton: "ServiceSpec"
instances:
- name: "gamma"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Exemple **`./frontend-svc/prod/frontend-spec-second.yaml`** :

```
proton: "ServiceSpec"
instances:
- name: "prod-one"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-two"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-three"
  environment: "frontend-env"
  spec:
```

```
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Si une instance ne se synchronise pas et que le problème persiste lors de la tentative de synchronisation, l'appel de l'`GetServiceInstanceSyncStatus`API peut aider à résoudre le problème.

> **Note**
>
> Les clients qui utilisent la synchronisation des services sont toujours soumis à AWS Proton des limites.

Bloqueurs

En synchronisant votre service à l'aide de la synchronisation des AWS Proton services, vous pouvez mettre à jour les spécifications de votre service et créer et mettre à jour des instances de service à partir de votre référentiel Git. Cependant, il peut arriver que vous deviez mettre à jour un service ou une instance manuellement via le AWS Management Console ou AWS CLI.

AWS Proton permet d'éviter de remplacer les modifications manuelles que vous apportez via le AWS Management Console ou AWS CLI, telles que la mise à jour d'une instance de service ou la suppression d'une instance de service. Pour ce faire, crée AWS Proton automatiquement un bloqueur de synchronisation des services en désactivant la synchronisation des services lorsqu'il détecte une modification manuelle.

Pour obtenir tous les bloqueurs associés à un service, vous devez effectuer les opérations suivantes dans l'ordre pour chacun des bloqueurs `serviceInstance` associés au service :

- Appelez l'`getServiceSyncBlockerSummary`API avec uniquement le`serviceName`.
- Appelez l'`getServiceSyncBlockerSummary`API avec le `serviceName` et`serviceInstanceName`.

Cela renvoie une liste des bloqueurs les plus récents et le statut qui leur est associé.
Si des bloqueurs sont marqués comme ACTIFS, vous devez les résoudre en appelant l'`UpdateServiceSyncBlocker`API avec `blockerId` et `resolvedReason` pour chacun d'eux.

Si vous mettez à jour ou créez manuellement une instance de service, AWS Proton crée un bloqueur de synchronisation de service sur l'instance de service. AWS Proton continue de synchroniser toutes les autres instances de service, mais désactive la synchronisation de cette instance de service jusqu'à ce que le bloqueur soit résolu. Si vous supprimez une instance de service d'un service, AWS Proton crée un bloqueur de synchronisation de services sur le service. Cela AWS Proton empêche la synchronisation des instances de service tant que le bloqueur n'est pas résolu.

Une fois que vous avez tous les bloqueurs actifs, vous devez les résoudre en appelant l'`UpdateServiceSyncBlocker`API avec le `blockerId` et `resolvedReason` pour chacun des bloqueurs actifs.

À l'aide du AWS Management Console, vous pouvez déterminer si la synchronisation d'un service est désactivée en accédant à l'onglet Synchronisation des services AWS Proton et en le sélectionnant. Si le service ou les instances de service sont bloqués, un bouton Activer apparaît. Pour activer la synchronisation des services, choisissez Activer.

Rubriques

Création d'une configuration de synchronisation des services

Vous pouvez créer une configuration de synchronisation des services à l'aide de la console ou AWS CLI.

AWS Management Console

1. Sur la page Choisir un modèle de service, sélectionnez un modèle et choisissez Configurer.
2. Sur la page Configurer le service, dans la section Détails du service, entrez un nouveau nom de service.
3. (Facultatif) Entrez une description du service.
4. Dans la section Référentiel du code source de l'application, choisissez Choisir un dépôt Git lié pour sélectionner un référentiel auquel vous êtes déjà lié AWS Proton. Si vous n'avez pas

encore de dépôt lié, choisissez Lier un autre dépôt Git et suivez les instructions de la section Créer un lien vers votre dépôt.

5. Pour Repository, choisissez le nom de votre référentiel de code source dans la liste.
6. Pour Branch, choisissez le nom de la branche du référentiel pour votre code source dans la liste.
7. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
8. Choisissez Suivant.
9. Sur la page Configurer les instances de service, dans la section Source de définition du service, sélectionnez Synchroniser votre service depuis Git.
10. Dans la section Fichiers de définition du service, si vous AWS Proton souhaitez créer votre `proton-ops` fichier, sélectionnez Je veux qu'AWS Proton crée les fichiers. Avec cette option, AWS Proton crée le `proton-ops` fichier `spec` et aux emplacements que vous spécifiez. Sélectionnez Je fournis mes propres fichiers pour créer votre propre fichier OPS.
11. Dans la section Référentiel de définitions de services, choisissez Choisir un dépôt Git lié pour sélectionner un référentiel auquel vous êtes déjà lié AWS Proton.
12. Pour Nom du référentiel, choisissez le nom de votre référentiel de code source dans la liste.
13. Pour la branche de **`proton-ops`** fichier, choisissez le nom de votre branche dans la liste où AWS Proton seront placés votre fichier OPS et vos fichiers de spécifications.
14. Dans la section Instances de service, chaque champ est automatiquement rempli en fonction des valeurs du `proton-ops` fichier.
15. Choisissez Next et passez en revue vos entrées.
16. Choisissez Créer.

AWS CLI

Créez une configuration de synchronisation des services à l'aide du AWS CLI

- Exécutez la commande suivante.

```
$ aws proton create-service-sync-config \
    --resource "service-arn" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
```

```
    --proton-ops-file "./configuration/custom-proton-ops.yaml" (optional)
```

La réponse est la suivante.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Afficher les détails de configuration pour une synchronisation de services

Vous pouvez consulter les données de configuration pour une synchronisation de service à l'aide de la console ou AWS CLI.

AWS Management Console

Utiliser la console pour afficher les détails de configuration d'une synchronisation de service

1. Dans le panneau de navigation, choisissez Services.
2. Pour afficher les données détaillées, choisissez le nom d'un service pour lequel vous avez créé une configuration de synchronisation de services.
3. Sur la page détaillée du service, sélectionnez l'onglet Synchronisation du service pour afficher les données détaillées de configuration pour la synchronisation du service.

AWS CLI

Utilisez le AWS CLI pour obtenir un service synchronisé.

Exécutez la commande suivante.

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

La réponse est la suivante.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Utilisez le AWS CLI pour obtenir l'état de synchronisation du service.

Exécutez la commande suivante.

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

Modifier une configuration de synchronisation de services

Vous pouvez modifier une configuration de synchronisation de services à l'aide de la console ou AWS CLI.

AWS Management Console

Modifiez une configuration de synchronisation de services à l'aide de la console.

1. Dans le panneau de navigation, choisissez Services.
2. Pour afficher les données détaillées, choisissez le nom d'un service pour lequel vous avez créé une configuration de synchronisation de services.
3. Sur la page détaillée du service, choisissez l'onglet Synchronisation des services.
4. Dans la section Synchronisation des services, choisissez Modifier.
5. Sur la page Modifier, mettez à jour les informations que vous souhaitez modifier, puis choisissez Enregistrer.

AWS CLI

Les exemples de commande et de réponse suivants montrent comment modifier une configuration de synchronisation de services à l'aide du AWS CLI.

Exécutez la commande suivante.

```
$ aws proton update-service-sync-config \
    --service-name "service name" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --ops-file "./configuration/custom-proton-ops.yaml"
```

La réponse est la suivante.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Supprimer une configuration de synchronisation de services

Vous pouvez supprimer une configuration de synchronisation de services à l'aide de la console ou AWS CLI.

AWS Management Console

Supprimer une configuration de synchronisation de services à l'aide de la console

1. Sur la page des détails du service, choisissez l'onglet Synchronisation des services.
2. Dans la section Détails de synchronisation du service, choisissez Déconnecter pour déconnecter votre référentiel. Une fois votre dépôt déconnecté, nous ne synchronisons plus le service à partir de ce référentiel.

AWS CLI

Les exemples de commandes et de réponses suivants montrent comment utiliser le AWS CLI pour supprimer les configurations synchronisées du service.

Exécutez la commande suivante.

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

La réponse est la suivante.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> La synchronisation des services ne supprime pas les instances de service. Elle supprime uniquement la configuration.

AWS Proton environnements

En AWS Proton effet, un environnement représente l'ensemble des ressources partagées et des politiques dans AWS Proton lesquelles les services sont déployés. Ils peuvent contenir toutes les ressources censées être partagées entre les instances AWS Proton de service. Ces ressources peuvent inclure des clusters VPCs, des équilibreurs de charge partagés ou des passerelles d'API. Un AWS Proton environnement doit être créé avant qu'un service puisse y être déployé.

Cette section décrit comment gérer les environnements à l'aide des opérations de création, d'affichage, de mise à jour et de suppression. Pour plus d'informations, consultez le manuel The AWS Proton Service API Reference.

Rubriques

Rôles IAM

Avec AWS Proton, vous fournissez les rôles et les AWS KMS clés IAM pour les AWS ressources que vous possédez et gérez. Ils sont ensuite appliqués et utilisés par les ressources détenues et gérées par les développeurs. Vous créez un rôle IAM pour contrôler l'accès de votre équipe de développeurs à l' AWS Proton API.

AWS Proton rôle de service

Lorsque vous créez un nouvel environnement, vous fournissez un rôle de service IAM associé. Le rôle contient toutes les autorisations nécessaires pour mettre à jour toute l'infrastructure provisionnée définie à la fois dans les modèles d'environnement et les modèles de service. Pour des exemples

de rôles, voirAWS Proton rôle de service pour le provisionnement à l'aide CloudFormation. Si vous utilisez des connexions à des comptes d'environnement et des comptes d'environnement, vous créez le rôle dans un compte d'environnement sélectionné. Pour plus d'informations, consultez Création d'un environnement sur un compte et mise à disposition sur un autre compte et Connexions aux comptes environnementaux.

La manière dont vous fournissez ce rôle de service, et qui assume ce rôle, dépend de la méthode de provisionnement de votre environnement.

- AWS-provisionnement géré : vous attribuez le rôle à AWS Proton, soit directement lors de la création d'un environnement, soit indirectement par le biais de connexions à des comptes. AWS Proton assume le rôle dans le compte pertinent pour fournir un environnement et une infrastructure de services.
- Approvisionnement autogéré : il est de votre responsabilité de configurer votre automatisation du provisionnement pour qu'elle assume le rôle approprié à l'aide des informations d'identification appropriées lorsqu'une pull request (PR) déclenche une action de provisionnement. Pour un exemple GitHub d'action qui assume un rôle, voir Assumer un rôle dans la documentation « Configurer les AWS informations d'identification » Action for GitHub Actions.

Pour plus d'informations sur les méthodes de provisionnement, consultezthe section called "Méthodes de provisionnement".

Création d'un environnement

Apprenez à créer des AWS Proton environnements.

Vous pouvez créer un AWS Proton environnement de deux manières différentes :

- Créez, gérez et approvisionnez un environnement standard à l'aide d'un modèle d'environnement standard. AWS Proton fournit une infrastructure pour votre environnement.
- Connectez-vous AWS Proton à une infrastructure gérée par le client à l'aide d'un modèle d'environnement géré par le client. Vous provisionnez vos propres ressources partagées en dehors de AWS Proton, puis vous fournissez des sorties de provisionnement qui AWS Proton peuvent être utilisées.

Vous pouvez choisir l'une des différentes approches de provisionnement lorsque vous créez un environnement.

- AWS provisionnement géré : créez, gérez et approvisionnez un environnement à partir d'un seul compte. AWS Proton approvisionne votre environnement.

  Cette méthode ne prend en charge que les modèles de code d' CloudFormation infrastructure (IaC).

- AWS provisionnement géré vers un autre compte : dans un seul compte de gestion, créez et gérez un environnement approvisionné sur un autre compte avec des connexions à des comptes d'environnement. AWS Proton approvisionne votre environnement dans l'autre compte. Pour plus d'informations, consultez Création d'un environnement sur un compte et mise à disposition sur un autre compte et Connexions aux comptes environnementaux.

  Cette méthode ne prend en charge que les CloudFormation modèles IaC.

- Provisionnement autogéré : AWS Proton envoie des pull requests de provisionnement à un référentiel lié doté de votre propre infrastructure de provisionnement.

  Cette méthode ne prend en charge que les modèles Terraform iAc.

- CodeBuild provisionnement : permet AWS Proton AWS CodeBuild d'exécuter les commandes shell que vous fournissez. Vos commandes peuvent lire les entrées qui AWS Proton fournissent et sont responsables du provisionnement ou du déprovisionnement de l'infrastructure et de la génération de valeurs de sortie. Un ensemble de modèles pour cette méthode inclut vos commandes dans un fichier manifeste et tous les programmes, scripts ou autres fichiers dont ces commandes peuvent avoir besoin.

  À titre d'exemple d'utilisation du CodeBuild provisionnement, vous pouvez inclure du code qui utilise le AWS Cloud Development Kit (AWS CDK) pour provisionner AWS des ressources, ainsi qu'un manifeste qui installe le CDK et exécute votre code CDK.

  Pour de plus amples informations, veuillez consulter the section called “CodeBuild offre groupée”.

> **Note**
>
> Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

Grâce au provisionnement AWS géré (à la fois sur le même compte et vers un autre compte), passez AWS Proton des appels directs pour approvisionner vos ressources.

Avec le provisionnement autogéré, AWS Proton émet des pull requests pour fournir des fichiers iAc compilés que votre moteur iAc utilise pour provisionner des ressources.

Pour plus d'informations, consultez the section called "Méthodes de provisionnement", the section called "Packs de modèles" et the section called "Exigences relatives au schéma d'environnement".

Rubriques

- Création et mise en service d'un environnement standard dans le même compte
- Création d'un environnement sur un compte et mise à disposition sur un autre compte
- Création et provisionnement d'un environnement à l'aide d'un provisionnement autogéré

Création et mise en service d'un environnement standard dans le même compte

Utilisez la console ou AWS CLI pour créer et approvisionner un environnement dans un seul compte. Le provisionnement est géré par AWS.

AWS Management Console

Utilisez la console pour créer et approvisionner un environnement dans un seul compte

1. Dans la AWS Proton console, choisissez Environments.
2. Choisissez Create environment.
3. Sur la page Choisir un modèle d'environnement, sélectionnez un modèle et choisissez Configurer.
4. Sur la page Configurer l'environnement, dans la section Provisioning, choisissez AWS Managed Provisioning.
5. Dans la section Compte de déploiement, choisissez Ceci Compte AWS.
6. Sur la page Configurer l'environnement, dans la section Paramètres d'environnement, entrez un nom d'environnement.
7. (Facultatif) Entrez une description de l'environnement.
8. Dans la section Rôles environnementaux, sélectionnez le rôle de AWS Proton service que vous avez créé dans le cadre de celui-ciConfiguration des rôles AWS Proton de service.

9. (Facultatif) Dans la section Rôle du composant, sélectionnez un rôle de service qui permet à des composants directement définis de s'exécuter dans l'environnement et limite les ressources qu'ils peuvent fournir. Pour de plus amples informations, veuillez consulter Éléments.
10. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
11. Choisissez Suivant.
12. Dans la page Configurer les paramètres personnalisés de l'environnement, vous devez entrer des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont définies.
13. Choisissez Next et passez en revue vos entrées.
14. Choisissez Créer.

    Consultez les détails et l'état de l'environnement, ainsi que les balises AWS gérées et les balises gérées par le client pour votre environnement.
15. Dans le panneau de navigation, choisissez Environments (Environnements).

    Une nouvelle page affiche la liste de vos environnements ainsi que leur statut et d'autres détails relatifs à l'environnement.

AWS CLI

Utilisez le AWS CLI pour créer et approvisionner un environnement dans un seul compte.

Pour créer un environnement, vous devez spécifier l'ARN du rôle de AWS Proton service, le chemin d'accès à votre fichier de spécifications, le nom de l'environnement, l'ARN du modèle d'environnement, les versions principale et secondaire et la description (facultatif).

Les exemples suivants montrent un fichier de spécifications YAML formaté qui spécifie les valeurs de deux entrées définies dans le fichier de schéma du modèle d'environnement. Vous pouvez utiliser la `get-environment-template-minor-version` commande pour afficher le schéma du modèle d'environnement.

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

Créez un environnement en exécutant la commande suivante.

```
$ aws proton create-environment \
    --name "MySimpleEnv" \
    --template-name simple-env \
    --template-major-version 1 \
    --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
 \
    --spec "file://env-spec.yaml"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateName": "simple-env"
    }
}
```

Après avoir créé un nouvel environnement, vous pouvez consulter une liste de AWS balises gérées par le client, comme illustré dans l'exemple de commande suivant. AWS Proton génère automatiquement des tags AWS gérés pour vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

Commande :

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv"
```

Création d'un environnement sur un compte et mise à disposition sur un autre compte

Utilisez la console ou AWS CLI créez un environnement standard dans un compte de gestion qui provisionne l'infrastructure environnementale dans un autre compte. Le provisionnement est géré par AWS.

Avant d'utiliser la console ou la CLI, effectuez les étapes suivantes.

1. Identifiez le compte Compte AWS IDs de gestion et d'environnement et copiez-le pour une utilisation ultérieure.
2. Dans le compte d'environnement, créez un rôle AWS Proton de service avec des autorisations minimales pour la création de l'environnement. Pour de plus amples informations, veuillez consulter AWS Proton rôle de service pour le provisionnement à l'aide CloudFormation.

AWS Management Console

Utilisez la console pour créer un environnement dans un compte et le provisionner dans un autre.

1. Dans le compte d'environnement, créez une connexion au compte d'environnement et utilisez-la pour envoyer une demande de connexion au compte de gestion.

   a. Dans AWS Proton la console, choisissez Environment account connections dans le volet de navigation.

   b. Sur la page de connexions aux comptes Environment, choisissez Request to connect.

   > **Note**
   >
   > Vérifiez que l'identifiant de compte indiqué dans l'en-tête de la page de connexion au compte d'environnement correspond à votre identifiant de compte d'environnement préidentifié.

   c. Sur la page Demande de connexion, dans la section Rôle dans l'environnement, sélectionnez Rôle de service existant et le nom du rôle de service que vous avez créé pour l'environnement.

d. Dans la section Se connecter au compte de gestion, entrez l'ID du compte de gestion et un nom d'environnement pour votre AWS Proton environnement. Copiez le nom pour une utilisation ultérieure.

e. Choisissez Demande de connexion dans le coin inférieur droit de la page.

f. Votre demande apparaît comme étant en attente dans le tableau des connexions de l'environnement envoyées à un compte de gestion et un modal indique comment accepter la demande depuis le compte de gestion.

2. Dans le compte de gestion, acceptez une demande de connexion depuis le compte d'environnement.

a. Connectez-vous à votre compte de gestion et choisissez Environment account connections dans la AWS Proton console.

b. Sur la page Connexions au compte d'environnement, dans le tableau des demandes de connexion au compte d'environnement, sélectionnez la connexion au compte d'environnement dont l'ID de compte d'environnement correspond à votre identifiant de compte d'environnement pré-identifié.

> **Note**
>
> Vérifiez que l'identifiant de compte indiqué dans l'en-tête de la page de connexion au compte Environnement correspond à votre identifiant de compte de gestion préidentifié.

c. Choisissez Accepter. Le statut passe de EN ATTENTE à CONNECTÉ.

3. Dans le compte de gestion, créez un environnement.

a. Dans le volet de navigation, sélectionnez Modèles d'environnement.

b. Sur la page Modèles d'environnement, choisissez Créer un modèle d'environnement.

c. Sur la page Choisir un modèle d'environnement, choisissez un modèle d'environnement.

d. Sur la page Configurer l'environnement, dans la section Provisioning, choisissez AWS Managed Provisioning.

e. Dans la section Compte de déploiement, choisissez Un autre AWS compte ;.

f. Dans la section Détails de l'environnement, sélectionnez la connexion à votre compte d'environnement et le nom de l'environnement.

g. Choisissez Suivant.

h. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page Réviser et créer.

i. Vérifiez et choisissez Créer un environnement.

AWS CLI

Utilisez le AWS CLI pour créer un environnement dans un compte et le provisionner dans un autre.

Dans le compte d'environnement, créez une connexion au compte d'environnement et demandez à vous connecter en exécutant la commande suivante.

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role",
        "status": "PENDING"
    }
}
```

Dans le compte de gestion, acceptez la demande de connexion au compte d'environnement en exécutant la commande suivante.

```
$ aws proton accept-environment-account-connection \
```

```
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Consultez la connexion à votre compte d'environnement en exécutant la commande suivante.

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Dans le compte de gestion, créez un environnement en exécutant la commande suivante.

```
$ aws proton create-environment \
    --name "simple-env-connected" \
    --template-name simple-env-template \
    --template-major-version "1" \
    --template-minor-version "1" \
    --spec "file://simple-env-template/specs/original.yaml" \
    --environment-account-connection-id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-
connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
        "templateName": "simple-env-template"
    }
}
```

Création et provisionnement d'un environnement à l'aide d'un provisionnement autogéré

Lorsque vous utilisez le provisionnement autogéré, vous AWS Proton soumettez des pull requests de provisionnement à un référentiel lié doté de votre propre infrastructure de provisionnement. Les pull requests démarrent votre propre flux de travail, qui fait appel à AWS des services, pour fournir une infrastructure.

Considérations relatives au provisionnement autogéré :

- Avant de créer un environnement, configurez un répertoire de ressources de référentiel pour un provisionnement autogéré. Pour de plus amples informations, veuillez consulter AWS Proton infrastructure sous forme de fichiers de code.

- Après avoir créé l'environnement, AWS Proton attend de recevoir des notifications asynchrones concernant l'état du provisionnement de votre infrastructure. Votre code de provisionnement doit utiliser l' AWS Proton `NotifyResourceStateChange`API pour envoyer ces notifications asynchrones à. AWS Proton

Vous pouvez utiliser le provisionnement autogéré dans la console ou avec le. AWS CLI Les exemples suivants montrent comment utiliser le provisionnement autogéré avec Terraform.

AWS Management Console

Utilisez la console pour créer un environnement Terraform à l'aide d'un provisionnement autogéré.

1. Dans la AWS Proton console, choisissez Environments.
2. Choisissez Create environment.
3. Sur la page Choisir un modèle d'environnement, sélectionnez un modèle Terraform et choisissez Configurer.
4. Sur la page Configurer l'environnement, dans la section Provisioning, choisissez Self-managed Provisioning.
5. Dans la section Détails du référentiel de provisionnement :

   a. Si vous n'avez pas encore lié votre référentiel de provisionnement à AWS Proton, choisissez Nouveau référentiel, choisissez l'un des fournisseurs de référentiels, puis, pour la CodeStarconnexion, choisissez l'une de vos connexions.

   > **Note**
   >
   > Si vous n'êtes pas encore connecté au compte du fournisseur de référentiel concerné, choisissez Ajouter une nouvelle CodeStar connexion. Créez ensuite une connexion, puis cliquez sur le bouton d'actualisation situé à côté du menu de CodeStar connexion. Vous devriez maintenant pouvoir choisir votre nouvelle connexion dans le menu.

   Si vous avez déjà lié votre dépôt à AWS Proton, choisissez Dépôt existant.

   b. Pour Nom du référentiel, choisissez un référentiel. Le menu déroulant affiche les référentiels liés pour le référentiel existant ou la liste des référentiels dans le compte du fournisseur pour le nouveau référentiel.

c. Dans Nom de la branche, choisissez l'une des branches du référentiel.

6. Dans la section Paramètres d'environnement, entrez un nom d'environnement.
7. (Facultatif) Entrez une description de l'environnement.
8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez Suivant.
10. Dans la page Configurer les paramètres personnalisés de l'environnement, vous devez entrer des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont définies.
11. Choisissez Next et passez en revue vos entrées.
12. Choisissez Create pour envoyer une pull request.

    - Si vous approuvez la pull request, le déploiement est en cours.
    - Si vous rejetez la pull request, la création de l'environnement est annulée.
    - Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.
13. Consultez les détails et l'état de l'environnement, ainsi que les balises AWS gérées et les balises gérées par le client pour votre environnement.
14. Dans le panneau de navigation, choisissez Environments (Environnements).

    Une nouvelle page affiche la liste de vos environnements ainsi que leur statut et d'autres détails relatifs à l'environnement.

AWS CLI

Lorsque vous créez un environnement à l'aide du provisionnement autogéré, vous ajoutez le `provisioningRepository` paramètre et omettez les `ProtonServiceRoleArn` paramètres et. `environmentAccountConnectionId`

Utilisez le AWS CLI pour créer un environnement Terraform avec un provisionnement autogéré.

1. Créez un environnement et envoyez une pull request au référentiel pour examen et approbation.

   Les exemples suivants montrent un fichier de spécifications YAML formaté qui définit les valeurs de deux entrées en fonction du fichier de schéma du modèle d'environnement.

Vous pouvez utiliser la `get-environment-template-minor-version` commande pour afficher le schéma du modèle d'environnement.

Spécification :

```
proton: EnvironmentSpec
spec:
  ssm_parameter_value: "test"
```

Créez un environnement en exécutant la commande suivante.

```
$ aws proton create-environment \
    --name "pr-environment" \
    --template-name "pr-env-template" \
    --template-major-version "1" \
    --provisioning-repository="branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec.yaml"
```

Réponse : **>**

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T17:06:58.679000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T17:06:58.679000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateName": "pr-env-template"
    }
```

2. Passez en revue la demande.

   - Si vous approuvez la demande, le provisionnement est en cours.

- Si vous rejetez la demande, la création de l'environnement est annulée.
- Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.

3. Fournissez de manière asynchrone le statut de provisionnement à. AWS Proton L'exemple suivant indique que le provisionnement a AWS Proton été effectué avec succès.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Afficher les données d'environnement

Vous pouvez afficher les données détaillées de l'environnement à l'aide de la AWS Proton console ou du AWS CLI.

AWS Management Console

Vous pouvez consulter des listes d'environnements avec des détails et des environnements individuels avec des données détaillées à l'aide de la AWS Proton console.

1. Pour afficher la liste de vos environnements, choisissez Environments dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un environnement.

   Consultez les données détaillées de votre environnement.

AWS CLI

Utilisez les détails de l'environnement AWS CLI get ou list.

Exécutez la commande suivante :

```
$ aws proton get-environment \
    --name "MySimpleEnv"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "lastDeploymentSucceededAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\nspec:\n  my_sample_input: \"the first\"\n
 my_other_sample_input: \"the second\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```


Mise à jour d'un environnement

Si l' AWS Proton environnement est associé à une connexion à un compte d'environnement, ne mettez pas à jour ni n'incluez le `protonServiceRoleArn` paramètre pour mettre à jour ou vous connecter à une connexion à un compte d'environnement.

Vous ne pouvez effectuer une mise à jour vers une nouvelle connexion à un compte d'environnement que si les deux conditions suivantes sont réunies :

- La connexion au compte d'environnement a été créée dans le même compte d'environnement que celui dans lequel la connexion au compte d'environnement actuel a été créée.
- >La connexion au compte d'environnement est associée à l'environnement actuel.

Si l'environnement n'est pas associé à une connexion à un compte d'environnement, ne mettez pas à jour ou n'incluez pas le `environmentAccountConnectionId` paramètre.

Vous pouvez mettre à jour le `protonServiceRoleArn` paramètre `environmentAccountConnectionId` ou la valeur. Vous ne pouvez pas mettre à jour les deux.

Si votre environnement utilise le provisionnement autogéré, ne mettez pas à jour le `provisioning-repository` paramètre et omettez les `environmentAccountConnectionId` paramètres et. `protonServiceRoleArn`

Il existe quatre modes de mise à jour d'un environnement, comme décrit dans la liste suivante. Lorsque vous utilisez le AWS CLI, le `deployment-type` champ définit le mode. Lorsque vous utilisez la console, ces modes sont associés aux actions Modifier, Mettre à jour, Mettre à jour mineure et Mettre à jour majeure qui se trouvent dans la liste déroulante Actions.

`NONE`

Dans ce mode, aucun déploiement n'a lieu. Seuls les paramètres de métadonnées demandés sont mis à jour.

`CURRENT_VERSION`

Dans ce mode, l'environnement est déployé et mis à jour avec les nouvelles spécifications que vous fournissez. Seuls les paramètres demandés sont mis à jour. N'incluez pas de paramètres de version mineurs ou majeurs lorsque vous l'utilisez`deployment-type`.

`MINOR_VERSION`

Dans ce mode, l'environnement est déployé et mis à jour avec la (dernière) version mineure publiée et recommandée de la version majeure actuellement utilisée par défaut. Vous pouvez également spécifier une version mineure différente de la version principale actuellement utilisée.

`MAJOR_VERSION`

Dans ce mode, l'environnement est déployé et mis à jour avec la (dernière) version majeure et mineure publiée et recommandée du modèle actuel par défaut. Vous pouvez également spécifier une version majeure différente, supérieure à la version principale utilisée, et une version secondaire (facultatif).

Rubriques

Mettre à jour un AWS environnement de provisionnement géré

Le provisionnement standard n'est pris en charge que par les environnements qui fournissent avec CloudFormation.

Utilisez la console ou AWS CLI pour mettre à jour votre environnement.

AWS Management Console

Mettez à jour un environnement à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Choisissez l'une des 2 étapes suivantes.
   a. Dans la liste des environnements.
      i. Dans la AWS Proton console, choisissez Environments.
      ii. Dans la liste des environnements, cliquez sur le bouton radio situé à gauche de l'environnement que vous souhaitez mettre à jour.
   b. Sur la page détaillée de l'environnement de console.
      i. Dans la AWS Proton console, choisissez Environments.
      ii. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez mettre à jour.
2. Choisissez l'une des 4 étapes suivantes pour mettre à jour votre environnement.
   a. Pour effectuer une modification qui ne nécessite pas de déploiement dans un environnement.
      i. Par exemple, pour modifier une description.

         Choisissez Modifier.
      ii. Remplissez le formulaire et choisissez Next.
      iii. Vérifiez votre modification et choisissez Mettre à jour.
   b. Pour mettre à jour les entrées de métadonnées uniquement.
      i. Choisissez Actions, puis Mettre à jour.
      ii. Remplissez le formulaire et choisissez Modifier.
      iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.
      iv. Passez en revue vos mises à jour et choisissez Mettre à jour.
   c. Pour mettre à jour une nouvelle version mineure de son modèle d'environnement.
      i. Choisissez Actions, puis Mettre à jour une version mineure.

ii. Remplissez le formulaire et choisissez Next.

iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.

iv. Passez en revue vos mises à jour et choisissez Mettre à jour.

d. Pour mettre à jour une nouvelle version majeure de son modèle d'environnement.

i. Choisissez Actions, puis Mettre à jour en majeur.

ii. Remplissez le formulaire et choisissez Next.

iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.

iv. Passez en revue vos mises à jour et choisissez Mettre à jour.

AWS CLI

Utilisez le AWS Proton AWS CLI pour mettre à jour un environnement vers une nouvelle version mineure.

Exécutez la commande suivante pour mettre à jour votre environnement :

```
$ aws proton update-environment \
        --name "MySimpleEnv" \
        --deployment-type "MINOR_VERSION" \
        --template-major-version "1" \
        --template-minor-version "1" \
        --proton-service-role-arn arn:aws:iam::123456789012:role/service-
role/ProtonServiceRole \
        --spec "file:///spec.yaml"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:29:55.472000+00:00",
        "name": "MySimpleEnv",
```

```
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

Exécutez la commande suivante pour obtenir et confirmer le statut :

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "MySimpleEnv",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Mettre à jour un environnement de provisionnement autogéré

Le provisionnement autogéré n'est pris en charge que par les environnements qui fournissent avec Terraform.

Utilisez la console ou AWS CLI pour mettre à jour votre environnement.

AWS Management Console

Mettez à jour un environnement à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Choisissez l'une des 2 étapes suivantes.
   a. Dans la liste des environnements.
      i. Dans la AWS Proton console, choisissez Environments.
      ii. Dans la liste des environnements, cliquez sur le bouton radio situé à gauche du modèle d'environnement que vous souhaitez mettre à jour.
   b. Sur la page détaillée de l'environnement de console.
      i. Dans la AWS Proton console, choisissez Environments.
      ii. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez mettre à jour.
2. Choisissez l'une des 4 étapes suivantes pour mettre à jour votre environnement.
   a. Pour effectuer une modification qui ne nécessite pas de déploiement dans un environnement.
      i. Par exemple, pour modifier une description.

         Choisissez Modifier.
      ii. Remplissez le formulaire et choisissez Next.
      iii. Vérifiez votre modification et choisissez Mettre à jour.
   b. Pour mettre à jour les entrées de métadonnées uniquement.
      i. Choisissez Actions, puis Mettre à jour.
      ii. Remplissez le formulaire et choisissez Modifier.
      iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.
      iv. Passez en revue vos mises à jour et choisissez Mettre à jour.
   c. Pour mettre à jour une nouvelle version mineure de son modèle d'environnement.
      i. Choisissez Actions, puis Mettre à jour une version mineure.
      ii. Remplissez le formulaire et choisissez Next.

iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.

iv. Passez en revue vos mises à jour et choisissez Mettre à jour.

d. Pour mettre à jour une nouvelle version majeure de son modèle d'environnement.

i. Choisissez Actions, puis Mettre à jour en majeur.

ii. Remplissez le formulaire et choisissez Next.

iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.

iv. Passez en revue vos mises à jour et choisissez Mettre à jour.

AWS CLI

Utilisez le AWS CLI pour mettre à jour un environnement Terraform vers une nouvelle version mineure avec un provisionnement autogéré.

1. Exécutez la commande suivante pour mettre à jour votre environnement :

```
$ aws proton update-environment \
    --name "pr-environment" \
    --deployment-type "MINOR_VERSION" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --provisioning-repository "branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec-mod.yaml"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:09:15.745000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
```

```
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "pr-env-template"
    }
}
```

2. Exécutez la commande suivante pour obtenir et confirmer le statut :

```
$ aws proton get-environment \
    --name "pr-environment"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:25:41.998000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "spec": "proton: EnvironmentSpec\nspec:\n   ssm_parameter_value: \"test
\"\n ssm_another_parameter_value: \"update\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "pr-env-template"
    }
}
```

3. Passez en revue la pull request envoyée par AWS Proton.

   - Si vous approuvez la demande, le provisionnement est en cours.
   - Si vous rejetez la demande, la création de l'environnement est annulée.
   - Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.

4. Indiquez le statut de provisionnement à. AWS Proton

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Annuler un déploiement d'environnement en cours

Vous pouvez essayer d'annuler le déploiement d'une mise à jour d'environnement si elle `deploymentStatus` existe`IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez le déploiement d'une mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

Avec AWS-managed provisioning, AWS Proton effectue les opérations suivantes :

- Définit l'état de déploiement sur`CANCELLING`.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement à quel moment`IN_PROGRESS`.
- Définit l'état de déploiement sur`CANCELLED`.
- Rétablit l'état de la ressource tel qu'il était avant le début du déploiement.

Avec le provisionnement autogéré, AWS Proton effectue les opérations suivantes :

- Tente de fermer la pull request pour empêcher la fusion des modifications apportées à votre dépôt.
- Définit l'état de déploiement `CANCELLED` comme si la pull request a été correctement fermée.

Pour obtenir des instructions sur la façon d'annuler le déploiement d'un environnement, reportez-vous CancelEnvironmentDeploymentà la référence des AWS Proton API.

Vous pouvez utiliser la console ou la CLI pour annuler les environnements en cours.

AWS Management Console

Utilisez la console pour annuler le déploiement d'une mise à jour d'environnement, comme indiqué dans les étapes suivantes.

1. Dans la AWS Proton console, choisissez Environments dans le volet de navigation.
2. Dans la liste des environnements, choisissez le nom de l'environnement contenant la mise à jour de déploiement que vous souhaitez annuler.
3. Si l'état du déploiement de votre mise à jour est En cours, sur la page détaillée de l'environnement, choisissez Actions, puis Annuler le déploiement.
4. Une fenêtre modale vous invite à confirmer que vous souhaitez annuler. Choisissez Annuler le déploiement.
5. L'état du déploiement de votre mise à jour est défini sur Annulation, puis sur Annulé pour terminer l'annulation.

AWS CLI

Utilisez le AWS Proton AWS CLI pour annuler le déploiement d'une mise à jour de l'environnement IN_PROGRESS vers une nouvelle version mineure 2.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant que le déploiement de la mise à jour ne réussisse.

Exécutez la commande suivante pour annuler la mise à jour :

```
$ aws proton cancel-environment-deployment \
        --environment-name "MySimpleEnv"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
```

```
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Exécutez la commande suivante pour obtenir et confirmer le statut : »

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Supprimer un environnement

Vous pouvez supprimer un AWS Proton environnement à l'aide de la AWS Proton console ou du AWS CLI.

> **Note**
>
> Vous ne pouvez pas supprimer un environnement associé à un composant. Pour supprimer un tel environnement, vous devez d'abord supprimer tous les composants qui s'exécutent dans l'environnement. Pour plus d'informations sur les composants, consultezÉléments.

AWS Management Console

Supprimez un environnement à l'aide de la console comme décrit dans les deux options suivantes.

Dans la liste des environnements.

1. Dans la AWS Proton console, choisissez Environments.
2. Dans la liste des environnements, sélectionnez le bouton radio situé à gauche de l'environnement que vous souhaitez supprimer.
3. Choisissez Actions, puis Supprimer.
4. Un modal vous invite à confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

Sur la page détaillée de l'environnement.

1. Dans la AWS Proton console, choisissez Environments.
2. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez supprimer.
3. Sur la page détaillée de l'environnement, choisissez Actions, puis Supprimer.
4. Une fenêtre modale vous invite à confirmer que vous souhaitez supprimer.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Utilisez le AWS CLI pour supprimer un environnement.

Ne supprimez pas un environnement si des services ou des instances de service y sont déployés.

Exécutez la commande suivante :

```
$ aws proton delete-environment \
    --name "MySimpleEnv"
```

Réponse :

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "DELETE_IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Connexions aux comptes environnementaux

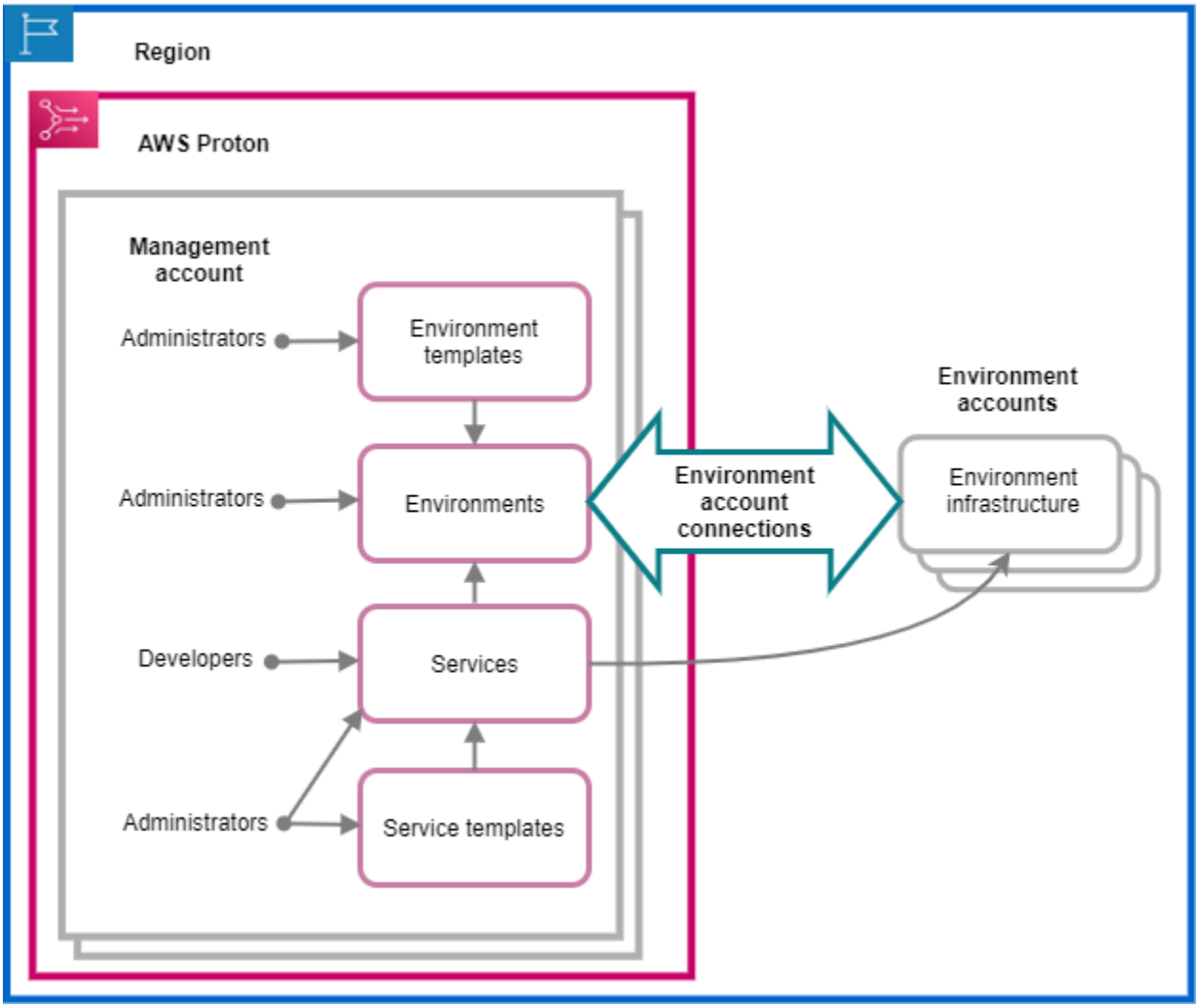
Présentation

Découvrez comment créer et gérer un AWS Proton environnement dans un compte et provisionner ses ressources d'infrastructure dans un autre compte. Cela peut contribuer à améliorer la visibilité et l'efficacité à grande échelle. Les connexions aux comptes d'environnement ne prennent en charge que le provisionnement standard avec CloudFormation l'infrastructure sous forme de code.

> **Note**
>
> Les informations contenues dans cette rubrique concernent les environnements configurés avec un provisionnement AWS géré. Lorsque les environnements sont configurés avec un

provisionnement autogéré, AWS Proton cela ne fournit pas directement votre infrastructure. Au lieu de cela, il envoie des pull requests (PRs) à votre référentiel pour le provisionnement. Il est de votre responsabilité de vous assurer que votre code d'automatisation assume l'identité et le rôle appropriés.
Pour plus d'informations sur les méthodes de provisionnement, consultezthe section called "Méthodes de provisionnement".

Terminologie



Avec les connexions aux comptes d' AWS Proton environnement, vous pouvez créer un AWS Proton environnement à partir d'un compte et provisionner son infrastructure sur un autre compte.

Compte de gestion

Compte unique dans lequel vous créez, en tant qu'administrateur, un AWS Proton environnement qui provisionne les ressources d'infrastructure dans un autre compte d'environnement.

Compte environnemental

Compte dans lequel l'infrastructure de l'environnement est provisionnée, lorsque vous créez un AWS Proton environnement dans un autre compte.

Connexion au compte environnemental

Connexion bidirectionnelle sécurisée entre un compte de gestion et un compte d'environnement. Il maintient les autorisations et les autorisations, comme décrit plus en détail dans les sections suivantes.

Lorsque vous créez une connexion à un compte d'environnement dans un compte d'environnement d'une région spécifique, seuls les comptes de gestion de la même région peuvent voir et utiliser la connexion au compte d'environnement. Cela signifie que l' AWS Proton environnement créé dans le compte de gestion et l'infrastructure environnementale provisionnée dans le compte d'environnement doivent se trouver dans la même région.

Considérations relatives à la connexion aux comptes environnementaux

- Vous avez besoin d'une connexion à un compte d'environnement pour chaque environnement que vous souhaitez provisionner dans un compte d'environnement.
- Pour plus d'informations sur les quotas de connexion aux comptes d'environnement, consultezAWS Proton quotas.

Identification

Dans le compte d'environnement, utilisez la console ou les balises gérées par le client AWS CLI pour afficher et gérer la connexion au compte d'environnement. AWS les balises gérées ne sont pas générées pour les connexions aux comptes d'environnement. Pour de plus amples informations, veuillez consulter Identification.

Créez un environnement sur un compte et provisionnez son infrastructure sur un autre compte

Pour créer et approvisionner un environnement à partir d'un seul compte de gestion, configurez un compte d'environnement pour un environnement que vous envisagez de créer.

Commencez dans le compte d'environnement et créez une connexion.

Dans le compte d'environnement, créez un rôle de AWS Proton service limité aux seules autorisations nécessaires au provisionnement des ressources de l'infrastructure de votre environnement. Pour de plus amples informations, veuillez consulter AWS Proton rôle de service pour le provisionnement à l'aide CloudFormation.

Créez ensuite une demande de connexion à un compte d'environnement et envoyez-la à votre compte de gestion. Lorsque la demande est acceptée, AWS Proton vous pouvez utiliser le rôle IAM associé qui permet le provisionnement des ressources environnementales dans le compte d'environnement associé.

Dans le compte de gestion, acceptez ou refusez la connexion au compte d'environnement.

Dans le compte de gestion, acceptez ou rejetez la demande de connexion au compte d'environnement. Vous ne pouvez pas supprimer une connexion à un compte d'environnement depuis votre compte de gestion.

Si vous acceptez la demande, vous AWS Proton pouvez utiliser le rôle IAM associé qui permet le provisionnement des ressources dans le compte d'environnement associé.

Les ressources de l'infrastructure environnementale sont provisionnées dans le compte d'environnement associé. Vous pouvez uniquement accéder AWS Proton APIs à votre environnement et à ses ressources d'infrastructure et les gérer à partir de votre compte de gestion. Pour plus d'informations, consultez Création d'un environnement sur un compte et mise à disposition sur un autre compte et Mise à jour d'un environnement.

Une fois que vous avez rejeté une demande, vous ne pouvez pas accepter ou utiliser la connexion au compte d'environnement rejetée.

> **Note**
>
> Vous ne pouvez pas refuser la connexion d'un compte d'environnement connecté à un environnement. Pour refuser la connexion au compte d'environnement, vous devez d'abord supprimer l'environnement associé.

Dans le compte d'environnement, accédez aux ressources d'infrastructure provisionnées.

Dans le compte d'environnement, vous pouvez consulter et accéder aux ressources d'infrastructure provisionnées. Par exemple, vous pouvez utiliser des actions d' CloudFormation API pour surveiller et nettoyer les piles si nécessaire. Vous ne pouvez pas utiliser les actions AWS Proton d'API pour accéder ou gérer l' AWS Proton environnement qui a été utilisé pour approvisionner les ressources de l'infrastructure.

Dans le compte d'environnement, vous pouvez supprimer les connexions au compte d'environnement que vous avez créées dans le compte d'environnement. Vous ne pouvez ni les accepter ni les rejeter. Si vous supprimez une connexion à un compte d'environnement utilisée par un AWS Proton environnement, vous ne AWS Proton serez pas en mesure de gérer les ressources de l'infrastructure de l'environnement tant qu'une nouvelle connexion à l'environnement n'aura pas été acceptée pour le compte d'environnement et l'environnement nommé. Vous êtes responsable du nettoyage des ressources provisionnées qui restent sans connexion à l'environnement.

Utiliser la console ou la CLI pour gérer les connexions aux comptes d'environnement

Vous pouvez utiliser la console ou la CLI pour créer et gérer les connexions aux comptes d'environnement.

AWS Management Console

Utilisez la console pour créer une connexion à un compte d'environnement et envoyer une demande au compte de gestion, comme indiqué dans les étapes suivantes.

1. Choisissez le nom de l'environnement que vous souhaitez créer dans votre compte de gestion ou choisissez le nom d'un environnement existant qui nécessite une connexion à un compte d'environnement.
2. Dans un compte d'environnement, dans la AWS Proton console, choisissez Connexions au compte d'environnement dans le volet de navigation.

3. Sur la page de connexions aux comptes Environment, choisissez Request to connect.

> **Note**
>
> Vérifiez l'ID de compte indiqué dans l'en-tête de la page de connexion au compte Environment. Assurez-vous qu'il correspond à l'ID de compte du compte d'environnement que vous souhaitez approvisionner en environnement nommé.

4. Sur la page Demande de connexion :

   a. Dans la section Se connecter au compte de gestion, entrez l'ID du compte de gestion et le nom de l'environnement que vous avez saisis à l'étape 1.

   b. Dans la section Rôle environnemental, choisissez Nouveau rôle de service et crée AWS Proton automatiquement un nouveau rôle pour vous. Vous pouvez également sélectionner Rôle de service existant et le nom du rôle de service que vous avez créé précédemment.

   > **Note**
   >
   > Le rôle créé AWS Proton automatiquement pour vous dispose d'autorisations étendues. Nous vous recommandons de limiter le rôle aux autorisations requises pour provisionner les ressources d'infrastructure de votre environnement. Pour de plus amples informations, veuillez consulter AWS Proton rôle de service pour le provisionnement à l'aide CloudFormation.

   c. (Facultatif) Dans la section Balises, choisissez Ajouter une nouvelle balise pour créer une balise gérée par le client pour la connexion à votre compte d'environnement.

   d. Choisissez Request to connect.

5. Votre demande apparaît comme étant en attente dans la table des connexions de l'environnement envoyées à un compte de gestion et un modal vous indique comment accepter la demande depuis le compte de gestion.

Acceptez ou rejetez une demande de connexion à un compte d'environnement.

1. Dans un compte de gestion, dans la AWS Proton console, choisissez Environment account connections dans le volet de navigation.

2. Sur la page Connexions au compte d'environnement, dans le tableau des demandes de connexion au compte d'environnement, choisissez la demande de connexion à l'environnement à accepter ou à rejeter.

> **Note**
>
> Vérifiez l'ID de compte indiqué dans l'en-tête de la page de connexion au compte Environment. Assurez-vous qu'il correspond à l'identifiant du compte de gestion associé à la connexion au compte d'environnement à rejeter. Une fois que vous avez rejeté cette connexion au compte d'environnement, vous ne pouvez pas accepter ou utiliser la connexion au compte d'environnement rejetée.

3. Choisissez Refuser ou Accepter.

   - Si vous avez sélectionné Rejeter, le statut passe de En attente à Rejeté.
   - Si vous avez sélectionné Accepter, le statut passe de En attente à Connecté.

Supprimez une connexion à un compte d'environnement.

1. Dans un compte d'environnement, dans la AWS Proton console, choisissez Connexions au compte d'environnement dans le volet de navigation.

> **Note**
>
> Vérifiez l'ID de compte indiqué dans l'en-tête de la page de connexion au compte Environment. Assurez-vous qu'il correspond à l'identifiant du compte de gestion associé à la connexion au compte d'environnement à rejeter. Une fois que vous avez supprimé cette connexion au compte d'environnement, vous ne AWS Proton pouvez pas gérer les ressources de l'infrastructure environnementale dans le compte d'environnement. Il ne peut le gérer qu'une fois qu'une nouvelle connexion au compte d'environnement pour le compte d'environnement et l'environnement nommé a été acceptée par le compte de gestion.

2. Sur la page Connexions au compte d'environnement, dans la section Demandes envoyées pour se connecter au compte de gestion, choisissez Supprimer.
3. Une fenêtre modale vous invite à confirmer que vous souhaitez supprimer. Sélectionnez Delete (Supprimer).

AWS CLI

Choisissez le nom de l'environnement que vous souhaitez créer dans votre compte de gestion ou choisissez le nom d'un environnement existant qui nécessite une connexion à un compte d'environnement.

Créez une connexion à un compte d'environnement dans un compte d'environnement.

Exécutez la commande suivante :

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role",
        "status": "PENDING"
    }
}
```

Acceptez ou rejetez une connexion à un compte d'environnement dans un compte de gestion, comme indiqué dans la commande et la réponse suivantes.

> **Note**
>
> Si vous rejetez cette connexion au compte d'environnement, vous ne pourrez ni accepter ni utiliser la connexion au compte d'environnement rejetée.

Si vous spécifiez Rejeter, le statut passe de En attente à Rejeté.

Si vous spécifiez Accepter, le statut passe de En attente à Connecté.

Exécutez la commande suivante pour accepter la connexion au compte d'environnement :

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Exécutez la commande suivante pour rejeter la connexion au compte d'environnement :

```
$ aws proton reject-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-reject",
```

```
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role"
    }
}
```

Afficher les connexions d'un compte d'environnement. Vous pouvez obtenir ou répertorier les connexions aux comptes d'environnement.

Exécutez la commande get suivante :

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Supprimez une connexion à un compte d'environnement dans un compte d'environnement.

> **Note**
>
> Si vous supprimez cette connexion au compte d'environnement, vous AWS Proton ne pourrez pas gérer les ressources de l'infrastructure environnementale dans le

compte d'environnement tant qu'une nouvelle connexion à l'environnement n'aura pas été acceptée pour le compte d'environnement et l'environnement nommé. Vous êtes responsable du nettoyage des ressources provisionnées qui restent sans connexion à l'environnement.

Exécutez la commande suivante :

```
$ aws proton delete-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Environnements gérés par le client

Avec les environnements gérés par le client, vous pouvez utiliser l'infrastructure existante, telle qu'un VPC, que vous avez déjà déployée en tant qu'environnement. AWS Proton Lorsque vous utilisez des environnements gérés par le client, vous pouvez provisionner vos propres ressources partagées en dehors de. AWS Proton Cependant, vous pouvez toujours autoriser la consommation AWS Proton des sorties de provisionnement pertinentes en tant qu'entrées pour les AWS Proton services lors de leur déploiement. Si les sorties peuvent changer, AWS Proton est en mesure d'accepter les mises à jour. AWS Proton n'est cependant pas en mesure de modifier directement l'environnement, car le provisionnement est géré en dehors de AWS Proton.

Une fois l'environnement créé, il vous incombe de fournir les mêmes résultats que ceux AWS Proton qui auraient été créés si l'environnement AWS Proton avait été créé, tels que les noms de clusters Amazon ECS ou Amazon VPC IDs.

Grâce à cette fonctionnalité, vous pouvez déployer et mettre à jour des ressources de AWS Proton service à partir d'un modèle de AWS Proton service vers cet environnement. Toutefois, l'environnement lui-même n'est pas modifié par les mises à jour des modèles dans AWS Proton. Vous êtes responsable de l'exécution des mises à jour de l'environnement et de la mise à jour de ces sorties dans AWS Proton.

Vous pouvez avoir plusieurs environnements dans un seul compte qui sont à la fois des environnements AWS Proton gérés et des environnements gérés par le client. Vous pouvez également associer un deuxième compte et utiliser un AWS Proton modèle dans le compte principal pour exécuter les déploiements et les mises à jour des environnements et des services de ce deuxième compte lié.

Comment utiliser les environnements gérés par le client

La première chose que les administrateurs doivent faire est d'enregistrer un modèle d'environnement importé et géré par le client. Ne fournissez pas de manifestes ou de fichiers d'infrastructure dans le bundle de modèles. Fournissez uniquement le schéma.

Le schéma ci-dessous présente une liste de sorties utilisant le format d'API ouvert et reproduit les sorties à partir d'un CloudFormation modèle.

> **Important**
>
> Seules les entrées sous forme de chaîne sont autorisées pour les sorties.

L'exemple suivant est un extrait des sections de sortie d'un CloudFormation modèle pour un modèle Fargate correspondant.

```
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
```

```
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

Le schéma de l'environnement AWS Proton importé correspondant est similaire au suivant. Ne fournissez pas de valeurs par défaut dans le schéma.

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentOutput"
  types:
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:
          type: string
          description: "The ARN of the ECS role"
        VpcId:
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Au moment de l'enregistrement du modèle, vous indiquez que ce modèle est importé et vous indiquez l'emplacement du compartiment Amazon S3 pour le bundle. AWS Proton vérifie que le schéma contient uniquement `environment_input_type` et aucun paramètre de CloudFormation modèle avant de mettre le modèle en brouillon.

Vous devez fournir les informations suivantes pour créer un environnement importé.

- Rôle IAM à utiliser lors des déploiements.
- Une spécification avec les valeurs pour les sorties requises.

Vous pouvez fournir les deux via la console ou à l' AWS CLI aide d'un processus similaire au déploiement d'un environnement normal.

CodeBuild création d'un rôle de provisionnement

Les outils d'infrastructure en tant que code (iAAC) tels que CloudFormation Terraform nécessitent des autorisations pour les nombreux types de ressources. AWS Par exemple, si un modèle iAAC déclare un compartiment Amazon S3, il a besoin d'autorisations pour créer, lire, mettre à jour et supprimer des compartiments Amazon S3. Il est considéré comme une bonne pratique de sécurité de limiter les rôles aux autorisations minimales requises. Compte tenu de l'étendue des AWS ressources, il est difficile de créer des politiques de moindre privilège pour les modèles iAAC, en particulier lorsque les ressources gérées par ces modèles peuvent changer ultérieurement. Par exemple, dans vos dernières modifications apportées à un modèle géré par AWS Proton, vous ajoutez une ressource de base de données RDS.

La configuration des autorisations appropriées permet de faciliter les déploiements de votre iAc. AWS Proton CodeBuild Le provisionnement exécute des commandes CLI arbitraires fournies par le client dans CodeBuild un projet situé dans le compte du client. Généralement, ces commandes créent et suppriment une infrastructure à l'aide d'un outil d'infrastructure en tant que code (iAAC) tel que AWS CDK. Lorsqu'une AWS ressource est déployée dont le modèle utilise le CodeBuild provisionnement, elle AWS démarre une génération dans un CodeBuild projet géré par. AWS Un rôle est transmis à CodeBuild, qui CodeBuild suppose d'exécuter des commandes. Ce rôle, appelé rôle de CodeBuild provisionnement, est fourni par le client et contient les autorisations requises pour approvisionner l'infrastructure. Il est censé être assumé uniquement par lui CodeBuild et ne AWS Proton peut même pas l'assumer.

Création du rôle

Le rôle CodeBuild Provisioning peut être créé dans la console IAM ou dans le. AWS CLI Pour le créer dans AWS CLI :

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AWSProtonCodeBuildProvisioningBasicAccess
```

Cela attache également le`AWSProtonCodeBuildProvisioningBasicAccess`, qui contient les autorisations minimales requises par le CodeBuild service pour exécuter une version.

Si vous préférez utiliser la console, vérifiez les points suivants lors de la création du rôle :

1. Pour une entité de confiance, sélectionnez AWS service, puis sélectionnez CodeBuild.

2. À l'étape Ajouter des autorisations, sélectionnez `AWSProtonCodeBuildProvisioningBasicAccess` et toutes les autres politiques que vous souhaitez associer.

Accès administrateur

Si vous associez la `AdministratorAccess` politique au rôle de CodeBuild provisionnement, cela garantira qu'aucun modèle iAAC n'échouera en raison d'un manque d'autorisations. Cela signifie également que toute personne capable de créer un modèle d'environnement ou un modèle de service peut effectuer des actions au niveau de l'administrateur, même si cet utilisateur n'est pas un administrateur. AWS Proton ne recommande pas de l'utiliser `AdministatorAccess` avec le rôle de CodeBuild provisioning. Si vous décidez de l'utiliser `AdministratorAccess` avec le rôle de CodeBuild provisionnement, faites-le dans un environnement sandbox.

Vous pouvez créer un rôle `AdministratorAccess` dans la console IAM ou en exécutant cette commande :

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

Création d'un rôle à portée minimale

Si vous souhaitez créer un rôle avec des autorisations minimales, il existe plusieurs approches :

- Déployez avec les autorisations d'administrateur, puis limitez le rôle. Nous vous recommandons d'utiliser IAM Access Analyzer.
- Utilisez des politiques gérées pour donner accès aux services que vous prévoyez d'utiliser.

AWS CDK

Si vous utilisez AWS CDK avec AWS Proton et que vous avez exécuté `cdk bootstrap` sur chaque compte/région d'environnement, il existe déjà un rôle pour. `cdk deploy` Dans ce cas, associez la politique suivante au rôle de CodeBuild provisionnement :

```
{
    "Action": "sts:AssumeRole",
```

```
    "Resource": [
        "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
        "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
    ],
    "Effect": "Allow"
}
```

VPC personnalisé

Si vous décidez de l'exécuter CodeBuild dans un VPC personnalisé, vous aurez besoin des autorisations suivantes dans votre CodeBuild rôle :

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:network-interface/*",
        "arn:aws:ec2:region:account-id:subnet/*",
        "arn:aws:ec2:region:account-id:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:*/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
```

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:AuthorizedService": "codebuild.amazonaws.com"
        }
    }
}
```

Vous pouvez également utiliser la politique AmazonEC2FullAccess gérée, même si elle inclut des autorisations dont vous n'aurez peut-être pas besoin. Pour associer la politique gérée à l'aide de la CLI :

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

AWS Proton services

Un AWS Proton service est une instanciation d'un modèle de service, qui inclut généralement plusieurs instances de service et un pipeline. Une instance AWS Proton de service est une instanciation d'un modèle de service dans un environnement spécifique. Un modèle de service est une définition complète de l'infrastructure et du pipeline de services optionnel pour un AWS Proton service.

Après avoir déployé vos instances de service, vous pouvez les mettre à jour en appuyant sur le code source pour lancer le CI/CD pipeline ou en mettant à jour le service avec les nouvelles versions de son modèle de service. AWS Proton vous invite lorsque de nouvelles versions de son modèle de service sont disponibles afin que vous puissiez mettre à jour vos services vers une nouvelle version. Lorsque votre service est mis à jour, AWS Proton redéploie le service et les instances de service.

Ce chapitre explique comment gérer les services à l'aide des opérations de création, d'affichage, de mise à jour et de suppression. Pour plus d'informations, consultez le manuel The AWS Proton Service API Reference.

Rubriques

Créer un service

Pour déployer une application avec AWS Proton, en tant que développeur, vous devez créer un service et fournir les entrées suivantes.

1. Le nom d'un modèle de AWS Proton service publié par l'équipe de la plateforme.
2. Nom du service.

3. Le nombre d'instances de service que vous souhaitez déployer.
4. Sélection d'environnements que vous souhaitez utiliser.
5. Une connexion à votre référentiel de code si vous utilisez un modèle de service incluant un pipeline de services (facultatif).

Qu'y a-t-il dans un service ?

Lorsque vous créez un AWS Proton service, vous pouvez choisir entre deux types de modèles de service différents :

- Modèle de service qui inclut un pipeline de services (par défaut).
- Modèle de service qui n'inclut pas de pipeline de services.

Vous devez créer au moins une instance de service lorsque vous créez votre service.

Une instance de service et un pipeline optionnel sont associés à un service. Vous ne pouvez créer ou supprimer un pipeline que dans le contexte des actions de création et de suppression de services. Pour savoir comment ajouter et supprimer des instances d'un service, consultezModifier un service.

> **Note**
>
> Votre environnement est configuré pour un provisionnement autogéré ou autogéré. AWS AWS Proton fournit des services dans un environnement en utilisant la même méthode de provisionnement que celle utilisée par l'environnement. Le développeur qui crée ou met à jour des instances de service ne voit pas la différence et son expérience est la même dans les deux cas.
> Pour plus d'informations sur les méthodes de provisionnement, consultezthe section called "Méthodes de provisionnement".

Modèles de services

Des versions majeures et mineures des modèles de service sont disponibles. Lorsque vous utilisez la console, vous sélectionnez la dernière version `Recommended` majeure et mineure du modèle de service. Lorsque vous utilisez le AWS CLI et que vous spécifiez uniquement la version principale du modèle de service, vous spécifiez implicitement sa dernière version `Recommended` mineure.

Ce qui suit décrit la différence entre les versions principales et secondaires des modèles ainsi que leur utilisation.

- Les nouvelles versions d'un modèle sont `Recommended` publiées dès qu'elles sont approuvées par un membre de l'équipe de la plateforme. Cela signifie que de nouveaux services sont créés à l'aide de cette version et que vous êtes invité à mettre à jour les services existants vers la nouvelle version.
- Grâce à AWS Proton cela, l'équipe de la plateforme peut automatiquement mettre à jour les instances de service vers une nouvelle version mineure d'un modèle de service. Les versions mineures doivent être rétrocompatibles.
- Étant donné que les versions majeures nécessitent que vous fournissiez de nouvelles entrées dans le cadre du processus de mise à jour, vous devez mettre à jour votre service vers une version majeure de son modèle de service. Les versions majeures ne sont pas rétrocompatibles.

Créer un service

Les procédures suivantes indiquent comment utiliser la AWS Proton console ou AWS CLI créer un service avec ou sans pipeline de services.

AWS Management Console

Créez un service comme indiqué dans les étapes de console suivantes.

1. Dans la AWS Proton console, choisissez Services.
2. Choisissez Créer un service.
3. Sur la page Choisir un modèle de service, sélectionnez un modèle et choisissez Configurer.

   Si vous ne souhaitez pas utiliser un pipeline activé, choisissez un modèle marqué avec Exclut le pipeline pour votre service.
4. Sur la page Configurer le service, dans la section Paramètres du service, entrez un nom de service.
5. (Facultatif) Entrez une description du service.
6. Dans la section Paramètres du référentiel de services :

   a. Pour CodeStar Connexion, choisissez votre connexion dans la liste.

   b. Pour Repository ID, choisissez le nom de votre référentiel de code source dans la liste.

c. Pour Nom de la branche, choisissez le nom de la branche de votre référentiel de code source dans la liste.

7. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
8. Choisissez Suivant.
9. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, dans la section Nouvelle instance. Vous devez saisir des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont définies.
10. Dans la section Entrées du pipeline, vous devez saisir des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont définies.
11. Choisissez Next et passez en revue vos entrées.
12. Choisissez Créer.

    Consultez les détails et l'état du service, ainsi que les balises AWS gérées et les balises gérées par le client pour votre service.
13. Dans le panneau de navigation, choisissez Services.

    Une nouvelle page affiche la liste de vos services ainsi que leur statut et d'autres détails.

AWS CLI

Lorsque vous utilisez le AWS CLI, vous spécifiez les entrées de service dans un `spec` fichier au format YAML.`aws-proton/service.yaml`, situé dans le répertoire de votre code source.

Vous pouvez utiliser la `get-service-template-minor-version` commande CLI pour afficher le schéma requis et les paramètres facultatifs pour lesquels vous fournissez des valeurs dans votre fichier de spécifications.

Si vous souhaitez utiliser un modèle de service qui en possèdepipelineProvisioning: "CUSTOMER_MANAGED", n'incluez pas la `pipeline:` section dans votre spécification et n'incluez *`-repository-connection-arn`* pas de `-branch-name` paramètres dans votre `create-service` commande. `-repository-id`

Créez un service avec un pipeline de services, comme indiqué dans les étapes de la CLI suivantes.

1. Configurez le rôle de service pour le pipeline comme indiqué dans l'exemple de commande CLI suivant.

   Commande :

   ```
   $ aws proton update-account-settings \
           --pipeline-service-role-arn "arn:aws:iam::123456789012:role/AWS
    ProtonServiceRole"
   ```

2. La liste suivante présente un exemple de spécification, basé sur le schéma du modèle de service, qui inclut le pipeline de services et les entrées d'instance.

   Spécification :

   ```
   proton: ServiceSpec

   pipeline:
     my_sample_pipeline_required_input: "hello"
     my_sample_pipeline_optional_input: "bye"

   instances:
     - name: "acme-network-dev"
       environment: "ENV_NAME"
       spec:
         my_sample_service_instance_required_input: "hi"
         my_sample_service_instance_optional_input: "ho"
   ```

   Créez un service avec un pipeline, comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

   Commande :

   ```
   $ aws proton create-service \
           --name "MySimpleService" \
           --branch-name "mainline" \
           --template-major-version "1" \
           --template-name "fargate-service" \
           --repository-connection-arn "arn:aws:codestar-connections:region-
   id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
           --repository-id "myorg/myapp" \
           --spec "file://spec.yaml"
   ```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Créez un service sans pipeline de services, comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Voici un exemple de spécification qui n'inclut pas les entrées du pipeline de services.

Spécification :

```
proton: ServiceSpec

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

Pour créer un service sans pipeline de services provisionné, vous devez fournir le chemin d'accès à un **`spec.yaml`** et vous n'incluez pas les paramètres du référentiel, comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Commande :

```
$ aws proton create-service \
        --name "MySimpleServiceNoPipeline" \
        --template-major-version "1" \
```

```
        --template-name "fargate-service" \
        --spec "file://spec-no-pipeline.yaml"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/
MySimpleServiceNoPipeline",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleServiceNoPipeline",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service-no-pipeline"
    }
}
```

Afficher les données de service

Vous pouvez afficher et répertorier les données détaillées des services à l'aide de la AWS Proton console ou du AWS CLI.

AWS Management Console

Répertoriez et affichez les détails du service à l'aide de la AWS Proton console, comme indiqué dans les étapes suivantes.

1. Pour afficher la liste de vos services, sélectionnez Services dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un service.

   Consultez les données détaillées de votre service.

AWS CLI

Affichez les détails d'un service avec un pipeline de services, comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Commande :

```
$ aws proton get-service \
```

```
    --name "simple-svc"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "repositoryId": "myorg/myapp",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

Affichez les détails d'un service sans pipeline de services, comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Commande :

```
$ aws proton get-service \
    --name "simple-svc-no-pipeline"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc-without-
pipeline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc-without-pipeline",
        "spec": "proton: ServiceSpec\ninstances:\n- name: instance-svc-simple\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_required_input:
 hi\n    my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple-no-pipeline"
    }
}
```

Modifier un service

Vous pouvez apporter les modifications suivantes à un AWS Proton service.

- Modifiez la description du service.
- Modifiez un service en ajoutant et en supprimant des instances de service.

Modifier la description du service

Vous pouvez utiliser la console ou le AWS CLI pour modifier la description d'un service.

AWS Management Console

Modifiez un service à l'aide de la console comme décrit dans les étapes suivantes.

Dans la liste des services.

1. Dans la AWS Proton console, choisissez Services.

2. Dans la liste des services, cliquez sur le bouton radio situé à gauche du service que vous souhaitez mettre à jour.
3. Choisissez Modifier.
4. Sur la page Configurer le service, remplissez le formulaire et choisissez Next.
5. Sur la page Configurer les paramètres personnalisés, choisissez Next.
6. Passez en revue vos modifications et choisissez Enregistrer les modifications.

Sur la page détaillée du service.

1. Dans la AWS Proton console, choisissez Services.
2. Dans la liste des services, choisissez le nom du service que vous souhaitez modifier.
3. Sur la page détaillée du service, choisissez Modifier.
4. Sur la page Configurer le service, remplissez le formulaire et choisissez Next.
5. Sur la page Configurer les paramètres personnalisés, remplissez le formulaire et choisissez Suivant.
6. Passez en revue vos modifications et choisissez Enregistrer les modifications.

AWS CLI

Modifiez une description comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Commande :

```
$ aws proton update-service \
    --name "MySimpleService" \
    --description "Edit by updating description"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2021-03-12T22:44:21.975000+00:00",
```

```
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "ACTIVE",
        "templateName": "fargate-service"
    }
}
```

Modifier un service pour ajouter ou supprimer des instances de service

Pour un AWS Proton service, vous pouvez ajouter ou supprimer des instances de service en soumettant une spécification modifiée. Les conditions suivantes doivent être remplies pour que la demande soit acceptée :

- Votre service et votre pipeline ne sont pas encore en cours de modification ou de suppression lorsque vous soumettez la demande de modification.
- Votre spécification modifiée n'inclut pas les modifications qui modifient le pipeline de services ni les modifications apportées aux instances de service existantes qui ne doivent pas être supprimées.
- Votre spécification modifiée ne supprime aucune instance de service existante à laquelle un composant est attaché. Pour supprimer une telle instance de service, vous devez d'abord mettre à jour le composant afin de le détacher de son instance de service. Pour plus d'informations sur les composants, consultezÉléments.

Les instances dont la suppression a échoué sont des instances de service en l'`DELETE_FAILED`état. Lorsque vous demandez une modification de service, AWS Proton tente de supprimer pour vous les instances dont la suppression a échoué, dans le cadre du processus de modification. Si l'une de vos instances de service n'a pas pu être supprimée, certaines ressources sont peut-être toujours associées aux instances, même si elles ne sont pas visibles depuis la console ou AWS CLI. Vérifiez les ressources de votre infrastructure d'instance dont la suppression a échoué et nettoyez-les afin de pouvoir les supprimer pour AWS Proton vous.

Pour connaître le quota d'instances de service pour un service, consultezAWS Proton quotas. Vous devez également gérer au moins une instance de service pour votre service après sa création. Au cours du processus de mise à jour, AWS Proton compte les instances de service existantes et les instances à ajouter ou à supprimer. Les instances dont la suppression a échoué sont incluses dans ce décompte et vous devez en tenir compte lorsque vous modifiez votre. `spec`

Utiliser la console ou AWS CLI pour ajouter ou supprimer des instances de service

AWS Management Console

Modifiez votre service pour ajouter ou supprimer des instances de service à l'aide de la console.

Dans la AWS Proton console

1. Dans le panneau de navigation, choisissez Services.
2. Sélectionnez le service que vous souhaitez modifier.
3. Choisissez Modifier.
4. (Facultatif) Sur la page Configurer le service, modifiez le nom ou la description du service, puis choisissez Suivant.
5. Sur la page Configurer les paramètres personnalisés, choisissez Supprimer pour supprimer une instance de service et choisissez Ajouter une nouvelle instance pour ajouter une instance de service et remplir le formulaire.
6. Choisissez Suivant.
7. Passez en revue votre mise à jour et choisissez Enregistrer les modifications.
8. Un modal vous demande de vérifier la suppression des instances de service. Suivez les instructions et choisissez Oui, supprimer.
9. Sur la page détaillée du service, consultez les détails de l'état de votre service.

AWS CLI

Ajoutez et supprimez des instances de service avec une modification**`spec`**, comme indiqué dans les AWS CLI exemples de commandes et de réponses suivants.

Lorsque vous utilisez la CLI, vous `spec` devez exclure les instances de service à supprimer et inclure à la fois les instances de service à ajouter et les instances de service existantes que vous n'avez pas marquées pour suppression.

La liste suivante montre l'exemple `spec` avant la modification et une liste des instances de service déployées par la spécification. Cette spécification a été utilisée dans l'exemple précédent pour modifier la description d'un service.

Spécification :

```
proton: ServiceSpec
```

```
pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "def"
      my_sample_service_instance_required_input: "456"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

L'exemple de `list-service-instances` commande et de réponse de la CLI ci-dessous montre les instances actives avant l'ajout ou la suppression d'une instance de service.

Commande :

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Réponse :

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "example-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
```

```
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.160000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.160000+00:00",
            "name": "my-instance",
            "serviceName": "example-svc",
            "serviceTemplateArn": "arn:aws:proton:region-id:123456789012:service-
template/fargate-service",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

La liste suivante montre l'exemple modifié `spec` utilisé pour supprimer et ajouter une instance. L'instance existante nommée `my-instance` est supprimée et une nouvelle instance nommée `yet-another-instance` est ajoutée.

Spécification :

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Vous pouvez l'utiliser `"${Proton::CURRENT_VAL}"` pour indiquer les valeurs de paramètres à conserver par rapport à l'originalspec, si les valeurs existent dans lespec. `get-service`À

utiliser pour afficher l'original `spec` d'un service, comme décrit dans[Afficher les données de service](#).

La liste suivante indique comment vous pouvez vous assurer que les modifications apportées `"${Proton::CURRENT_VAL}"` aux valeurs des paramètres `spec` ne sont pas incluses pour que les instances de services existantes soient conservées.

Spécification :

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

La liste suivante indique la commande et la réponse de la CLI pour modifier le service.

Commande :

```
$ aws proton update-service
    --name "MySimpleService" \
    --description "Edit by adding and deleting a service instance" \
    --spec "file://spec.yaml"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by adding and deleting a service instance",
```

```
        "lastModifiedAt": "2021-03-12T22:55:48.169000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "UPDATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

La `list-service-instances` commande et la réponse suivantes confirment que l'instance existante nommée `my-instance` est supprimée et qu'une nouvelle instance nommée `yet-another-instance` est ajoutée.

Commande :

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Réponse :

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/yet-another-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:56:01.565000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:56:01.565000+00:00",
            "name": "yet-another-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
```

```
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Que se passe-t-il lorsque vous ajoutez ou supprimez des instances de service

Après avoir soumis une modification de service pour supprimer et ajouter des instances de service, AWS Proton prend les mesures suivantes.

- Définit le service sur`UPDATE_IN_PROGRESS`.
- Si le service possède un pipeline, définit son statut sur `IN_PROGRESS` et bloque les actions du pipeline.
- Définit toutes les instances de service à supprimer sur`DELETE_IN_PROGRESS`.
- Bloque les actions de service.
- Bloque les actions sur les instances de service marquées pour suppression.
- Crée de nouvelles instances de service.
- Supprime les instances que vous avez répertoriées pour suppression.
- Tente de supprimer les instances dont la suppression a échoué.
- Une fois les ajouts et les suppressions terminés, réapprovisionne le pipeline de services (le cas échéant), définit votre service sur `ACTIVE` et active les actions de service et de pipeline.

AWS Proton tente de remédier aux modes de défaillance comme suit.

- Si une ou plusieurs instances de service n'ont pas pu être créées, AWS Proton essaie de déprovisionner toutes les instances de service nouvellement créées et rétablit `spec` l'état précédent. Il ne supprime aucune instance de service et ne modifie en aucune façon le pipeline.
- Si une ou plusieurs instances de service n'ont pas pu être supprimées, AWS Proton réapprovisionne le pipeline sans les instances supprimées. Le `spec` est mis à jour pour inclure les instances ajoutées et pour exclure les instances marquées pour suppression.

- Si le pipeline échoue au provisionnement, aucune annulation n'est tentée et le service et le pipeline reflètent un état d'échec de mise à jour.

Marquage et modifications de service

Lorsque vous ajoutez des instances de service dans le cadre de la modification de votre service, les balises AWS gérées se propagent vers les nouvelles instances et les ressources allouées et sont automatiquement créées pour celles-ci. Si vous créez de nouvelles balises, celles-ci ne sont appliquées qu'aux nouvelles instances. Les balises de service gérées par les clients existants se propagent également aux nouvelles instances. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

Supprimer un service

Vous pouvez supprimer un AWS Proton service, avec ses instances et son pipeline, à l'aide de la AWS Proton console ou du AWS CLI.

Vous ne pouvez pas supprimer un service qui possède une instance de service associée à un composant. Pour supprimer un tel service, vous devez d'abord mettre à jour tous les composants attachés afin de les détacher de leurs instances de service. Pour plus d'informations sur les composants, consultezÉléments.

AWS Management Console

Supprimez un service à l'aide de la console comme décrit dans les étapes suivantes.

Sur la page détaillée du service.

1. Dans la AWS Proton console, choisissez Services.
2. Dans la liste des services, choisissez le nom du service que vous souhaitez supprimer.
3. Sur la page détaillée du service, choisissez Actions, puis Supprimer.
4. Un modal vous invite à confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Supprimez un service comme indiqué dans l'exemple de commande et de réponse de la CLI ci-dessous.

Commande :

```
$ aws proton delete-service \
    --name "simple-svc"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2020-11-29T00:30:39.248000+00:00",
        "name": "simple-svc",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "DELETE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Afficher les données des instances de service

Apprenez à afficher les données détaillées des instances de AWS Proton service. Vous pouvez utiliser la console ou le AWS CLI.

Une instance de service appartient à un service. Vous ne pouvez créer ou supprimer une instance que dans le cadre d'actions de modification, de création et de suppression de services. Pour savoir comment ajouter et supprimer des instances d'un service, consultezModifier un service.

AWS Management Console

Répertoriez et affichez les détails de l'instance de service à l'aide de la AWS Proton console, comme indiqué dans les étapes suivantes.

1. Pour afficher la liste de vos instances de service, choisissez Instances de services dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'une instance de service.

Consultez les données détaillées de votre instance de service.

AWS CLI

Répertoriez et affichez les détails de l'instance de service, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Commande :

```
$ aws proton list-service-instances
```

Réponse :

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
service-instance/instance-one",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentArn": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "name": "instance-one",
            "serviceName": "simple-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Commande :

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Réponse :

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
        "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: hello world\n
 my_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_optional_input:
 Ola\n    my_sample_service_instance_required_input: Ciao\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```


Mettre à jour une instance de service

Apprenez à mettre à jour une instance AWS Proton de service et à annuler la mise à jour.

Une instance de service appartient à un service. Vous ne pouvez créer ou supprimer une instance que dans le cadre d'actions de modification, de création et de suppression de services. Pour savoir comment ajouter et supprimer des instances d'un service, consultezModifier un service.

Il existe quatre modes de mise à jour d'une instance de service, comme décrit dans la liste suivante. Lorsque vous utilisez le AWS CLI, le `deployment-type` champ définit le mode. Lorsque vous utilisez la console, ces modes correspondent aux actions Modifier et Mettre à jour vers la dernière version mineure et Mettre à jour vers la dernière version majeure qui figurent dans la liste déroulante Actions de la page détaillée de l'instance de service.

`NONE`

Dans ce mode, aucun déploiement n'a lieu. Seuls les paramètres de métadonnées demandés sont mis à jour.

`CURRENT_VERSION`

Dans ce mode, l'instance de service est déployée et mise à jour avec les nouvelles spécifications que vous fournissez. Seuls les paramètres demandés sont mis à jour. N'incluez pas de paramètres de version mineurs ou majeurs lorsque vous l'utilisez`deployment-type`.

`MINOR_VERSION`

Dans ce mode, l'instance de service est déployée et mise à jour avec la (dernière) version mineure publiée et recommandée de la version majeure actuellement utilisée par défaut. Vous pouvez également spécifier une version mineure différente de la version principale actuellement utilisée.

`MAJOR_VERSION`

Dans ce mode, l'instance de service est déployée et mise à jour avec la (dernière) version majeure et mineure publiée et recommandée du modèle actuel par défaut. Vous pouvez également spécifier une version majeure différente, supérieure à la version principale utilisée, et une version secondaire (facultatif).

Vous pouvez essayer d'annuler le déploiement d'une mise à jour d'instance de service si tel `deploymentStatus` est le cas`IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez le déploiement d'une mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

- Définit l'état de déploiement sur`CANCELLING`.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement à quel moment`IN_PROGRESS`.
- Définit l'état de déploiement sur`CANCELLED`.
- Rétablit l'état de la ressource tel qu'il était avant le début du déploiement.

Pour plus d'informations sur l'annulation du déploiement d'une instance de service, consultez CancelServiceInstanceDeploymentla référence des AWS Proton API.

Utilisez la console ou AWS CLI pour effectuer des mises à jour ou annuler des déploiements de mises à jour.

AWS Management Console

Mettez à jour une instance de service à l'aide de la console en suivant ces étapes.

1. Dans la AWS Proton console, choisissez Service instances dans le volet de navigation.
2. Dans la liste des instances de service, choisissez le nom de l'instance de service que vous souhaitez mettre à jour.
3. Choisissez Actions, puis choisissez l'une des options de mise à jour, Modifier pour mettre à jour les spécifications ou les actions, puis Mettre à jour vers la dernière version mineure ou Mettre à jour vers la dernière version majeure.
4. Remplissez chaque formulaire et choisissez Suivant jusqu'à ce que vous atteigniez la page d'évaluation.
5. Passez en revue vos modifications et choisissez Mettre à jour.

AWS CLI

Mettez à jour une instance de service vers une nouvelle version mineure, comme indiqué dans les exemples de commandes et de réponses de la CLI.

Lorsque vous mettez à jour votre instance de service avec une modification`spec`, vous pouvez l'utiliser `"${Proton::CURRENT_VAL}"` pour indiquer les valeurs de paramètres à conserver par rapport à l'original`spec`, si les valeurs existent dans le`spec`. `get-service`À utiliser pour afficher l'original d'`spec`une instance de service, comme décrit dansAfficher les données de service.

L'exemple suivant montre comment vous pouvez utiliser `"${Proton::CURRENT_VAL}"` dans un`spec`.

Spécification :

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
```

```
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Commande : pour mettre à jour

```
$ aws proton update-service-instance \
    --name "instance-one" \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Réponse :

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentName": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Réponse :


```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```


AWS Management Console

Annulez le déploiement d'une instance de service à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Dans la AWS Proton console, choisissez Service instances dans le volet de navigation.
2. Dans la liste des instances de service, choisissez le nom de l'instance de service dont vous souhaitez annuler la mise à jour de déploiement.
3. Si le statut de votre déploiement de mise à jour est En cours, sur la page détaillée de l'instance de service, choisissez Actions, puis Annuler le déploiement.
4. Un modal vous demande de confirmer l'annulation. Choisissez Annuler le déploiement.
5. L'état du déploiement de votre mise à jour est défini sur Annulation, puis sur Annulé pour terminer l'annulation.

AWS CLI

Annulez la mise à jour du déploiement d'une instance de service IN_PROGRESS vers la nouvelle version mineure 2, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant que le déploiement de la mise à jour ne réussisse.

Commande : pour annuler

```
$ aws proton cancel-service-instance-deployment \
    --service-instance-name "instance-one" \
    --service-name "simple-svc"
```

Réponse :

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: abc\n  my_sample_pipeline_required_input:
 '123'\ninstances:\n- name: my-instance\n  environment: MySimpleEnv
\n  spec:\n    my_sample_service_instance_optional_input: def\n
 my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
 environment: MySimpleEnv\n  spec:\n    my_sample_service_instance_required_input:
 '789'\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Réponse :

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Mettre à jour un pipeline de services

Apprenez à mettre à jour un pipeline AWS Proton de services et à annuler la mise à jour.

Un pipeline de services appartient à un service. Vous ne pouvez créer ou supprimer un pipeline que dans le contexte des actions de création et de suppression de services.

Il existe quatre modes de mise à jour d'un pipeline de services, comme décrit dans la liste suivante. Lorsque vous utilisez le AWS CLI, le `deployment-type` champ définit le mode. Lorsque vous

utilisez la console, ces modes correspondent au pipeline d'édition et au mode Mettre à jour vers la version recommandée.

`NONE`

Dans ce mode, aucun déploiement n'a lieu. Seuls les paramètres de métadonnées demandés sont mis à jour.

`CURRENT_VERSION`

Dans ce mode, le pipeline de services est déployé et mis à jour avec les nouvelles spécifications que vous fournissez. Seuls les paramètres demandés sont mis à jour. N'incluez pas de paramètres de version mineurs ou majeurs lorsque vous l'utilisez`deployment-type`.

`MINOR_VERSION`

Dans ce mode, le pipeline de services est déployé et mis à jour avec la (dernière) version mineure publiée et recommandée de la version majeure actuellement utilisée par défaut. Vous pouvez également spécifier une version mineure différente de la version principale actuellement utilisée.

`MAJOR_VERSION`

Dans ce mode, le pipeline de services est déployé et mis à jour avec la (dernière) version majeure et mineure publiée et recommandée du modèle actuel par défaut. Vous pouvez également spécifier une version majeure différente, supérieure à la version principale utilisée, et une version secondaire (facultatif).

Vous pouvez essayer d'annuler le déploiement d'une mise à jour du pipeline de services si `deploymentStatus` c'est le cas`IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez le déploiement d'une mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

- Définit l'état de déploiement sur`CANCELLING`.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement à quel moment`IN_PROGRESS`.
- Définit l'état de déploiement sur`CANCELLED`.
- Rétablit l'état de la ressource tel qu'il était avant le début du déploiement.

Pour plus d'informations sur l'annulation d'un déploiement de pipeline de services, consultez CancelServicePipelineDeploymentla référence des AWS Proton API.

Utilisez la console ou AWS CLI pour effectuer des mises à jour ou annuler des déploiements de mises à jour.

AWS Management Console

Mettez à jour un pipeline de services à l'aide de la console, comme décrit dans les étapes suivantes.

1. Dans la AWS Proton console, choisissez Services.
2. Dans la liste des services, choisissez le nom du service pour lequel vous souhaitez mettre à jour le pipeline.
3. La page détaillée du service comporte deux onglets : Aperçu et Pipeline. Choisissez Pipeline.
4. Si vous souhaitez mettre à jour les spécifications, choisissez Modifier le pipeline, remplissez chaque formulaire, puis cliquez sur Suivant jusqu'à ce que vous remplissiez le formulaire final, puis choisissez Mettre à jour le pipeline.

   Si vous souhaitez effectuer une mise à jour vers une nouvelle version et qu'une icône d'information indique qu'une nouvelle version est disponible dans le modèle de pipeline, choisissez le nom de la nouvelle version du modèle.

   a. Choisissez Mettre à jour vers la version recommandée.
   b. Remplissez chaque formulaire et choisissez Suivant jusqu'à ce que vous remplissiez le formulaire final et que vous choisissiez Mettre à jour.

AWS CLI

Mettez à jour un pipeline de services vers une nouvelle version mineure, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Lorsque vous mettez à jour votre pipeline de service avec une modificationspec, vous pouvez l'utiliser `"${Proton::CURRENT_VAL}"` pour indiquer les valeurs de paramètres à conserver par rapport à l'originalspec, si les valeurs existent dans lespec. `get-service`À utiliser pour afficher l'original d'specun pipeline de services, comme décrit dansAfficher les données de service.

L'exemple suivant montre comment vous pouvez utiliser `"${Proton::CURRENT_VAL}"` dans unspec.

Spécification :

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Commande : pour mettre à jour

```
$ aws proton update-service-pipeline \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Réponse :

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"my-instance\"\n    environment: \"MySimpleEnv
\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
```

```
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"MySimpleEnv\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service \
    --name "simple-svc"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

AWS Management Console

Annulez le déploiement d'un pipeline de services à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Dans la AWS Proton console, choisissez Services dans le volet de navigation.
2. Dans la liste des services, choisissez le nom du service qui possède le pipeline contenant la mise à jour de déploiement que vous souhaitez annuler.
3. Sur la page détaillée du service, choisissez l'onglet Pipeline.
4. Si le statut de votre déploiement de mise à jour est En cours, sur la page détaillée du pipeline de services, choisissez Annuler le déploiement.
5. Un modal vous demande de confirmer l'annulation. Choisissez Annuler le déploiement.
6. L'état du déploiement de votre mise à jour est défini sur Annulation, puis sur Annulé pour terminer l'annulation.

AWS CLI

Annulez une mise à jour du déploiement du pipeline de service IN_PROGRESS vers la version mineure 2, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant que le déploiement de la mise à jour ne réussisse.

Commande : pour annuler

```
$ aws proton cancel-service-pipeline-deployment \
```

```
    --service-name "simple-svc"
```

Réponse :

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service \
    --name "simple-svc"
```

Réponse :

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "CANCELLED",
            "deploymentStatusMessage": "User initiated cancellation.",
            "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
```

```
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

AWS Proton composants

Les composants sont un type de AWS Proton ressource. Ils ajoutent de la flexibilité aux modèles de services. Les composants fournissent aux équipes chargées de la plateforme un mécanisme permettant d'étendre les modèles d'infrastructure de base et de définir des garanties qui permettent aux développeurs de gérer certains aspects de leur infrastructure d'applications.

Dans AWS Proton les administrateurs, définissez l'infrastructure standard utilisée par les équipes de développement et les applications. Cependant, les équipes de développement peuvent avoir besoin d'inclure des ressources supplémentaires pour leurs cas d'utilisation spécifiques, comme les files d'attente Amazon Simple Queue Service (Amazon SQS) ou les tables Amazon DynamoDB. Ces ressources spécifiques aux applications peuvent changer fréquemment, en particulier au début du développement des applications. Le maintien de ces modifications fréquentes dans les modèles créés par les administrateurs peut s'avérer difficile à gérer et à adapter : les administrateurs devraient gérer un plus grand nombre de modèles sans véritable valeur ajoutée pour les administrateurs. L'alternative, qui consiste à laisser les développeurs d'applications créer des modèles pour leurs applications, n'est pas idéale non plus, car elle empêche les administrateurs de normaliser les principaux composants de l'architecture, tels que les tâches. AWS Fargate C'est là que les composants entrent en jeu.

Avec un composant, un développeur peut ajouter des ressources supplémentaires à son application, au-delà de ce que les administrateurs ont défini dans les modèles d'environnement et de service. Le développeur attache ensuite le composant à une instance de service. AWS Proton provisionne les ressources d'infrastructure définies par le composant de la même manière qu'il provisionne les ressources pour les environnements et les instances de service.

Un composant peut lire les entrées d'une instance de service et fournir des sorties à l'instance de service, pour une expérience totalement intégrée. Par exemple, si le composant ajoute un compartiment Amazon Simple Storage Service (Amazon S3) destiné à être utilisé par une instance de service, le modèle de composant peut prendre en compte les noms de l'environnement et des instances de service pour nommer le compartiment. Lors du AWS Proton rendu du modèle de service pour provisionner une instance de service, l'instance de service peut faire référence au compartiment et l'utiliser.

Les composants AWS Proton actuellement pris en charge sont des composants directement définis. Vous transmettez le fichier Infrastructure as Code (IaC) qui définit l'infrastructure du composant directement à l' AWS Proton API ou à la console. Cela est différent d'un environnement ou d'un service, dans lequel vous définissez iAc dans un ensemble de modèles et vous enregistrez le

bundle en tant que ressource de modèle, puis vous utilisez une ressource de modèle pour créer l'environnement ou le service.

> **Note**
>
> Les composants directement définis permettent aux développeurs de définir une infrastructure supplémentaire et de la provisionner. AWS Proton provisionne tous les composants directement définis exécutés dans le même environnement en utilisant le même rôle Gestion des identités et des accès AWS (IAM).

Un administrateur peut contrôler ce que les développeurs peuvent faire avec les composants de deux manières :

- Sources de composants prises en charge : un administrateur peut autoriser l'attachement de composants à des instances de service en fonction d'une propriété des versions des modèles de AWS Proton service. Par défaut, les développeurs ne peuvent pas associer de composants aux instances de service.

  Pour plus d'informations sur cette propriété, consultez le supportedComponentSourcesparamètre de l'action d'CreateServiceTemplateVersionAPI dans la référence AWS Proton d'API.

  > **Note**
  >
  > Lorsque vous utilisez la synchronisation de modèles, AWS Proton crée implicitement des versions de modèles de service lorsque vous validez les modifications apportées à un ensemble de modèles de services dans un référentiel. Dans ce cas, au lieu de spécifier les sources de composants prises en charge lors de la création de la version du modèle de service, vous spécifiez cette propriété dans un fichier associé à chaque version majeure du modèle de service. Pour de plus amples informations, veuillez consulter the section called "Modèles de service de synchronisation".

- Rôles des composants : un administrateur peut attribuer un rôle de composant à un environnement. AWS Proton assume ce rôle lorsqu'il provisionne une infrastructure définie par un composant directement défini dans l'environnement. Par conséquent, le rôle du composant limite l'infrastructure que les développeurs peuvent ajouter à l'aide de composants directement définis dans l'environnement. En l'absence du rôle de composant, les développeurs ne peuvent pas créer de composants directement définis dans l'environnement.

Pour plus d'informations sur l'attribution d'un rôle de composant, consultez le componentRoleArnparamètre de l'action d'CreateEnvironmentAPI dans la référence d'AWS Proton API.

> **Note**
>
> Les rôles des composants ne sont pas utilisés dans Provisionnement autogéré les environnements.

Rubriques

Comment les composants se comparent-ils aux autres AWS Proton ressources ?

À bien des égards, les composants sont similaires aux autres AWS Proton ressources. Leur infrastructure est définie dans un fichier modèle IaC, créé au format CloudFormation YAML ou Terraform HCL. AWS Proton peut provisionner une infrastructure de composants à l'aide d'un provisionnement AWS géré ou d'un provisionnement autogéré.

Les composants sont toutefois différents des autres AWS Proton ressources à plusieurs égards :

- État détaché : les composants sont conçus pour être attachés à des instances de service et pour étendre leur infrastructure, mais ils peuvent également être dans un état détaché, dans lequel ils ne sont attachés à aucune instance de service. Pour plus d'informations sur les états des composants, consultezthe section called “États des composants”.

- Pas de schéma : les composants ne sont pas associés à un schéma contrairement aux ensembles de modèles. Les entrées des composants sont définies par un service. Un composant peut consommer des entrées lorsqu'il est attaché à une instance de service.
- Aucun composant géré par le client : provisionne AWS Proton toujours l'infrastructure des composants pour vous. Il n'existe pas de version « apportez vos propres ressources » des composants. Pour plus d'informations sur les environnements gérés par le client, consultez. the section called “Créer”
- Aucune ressource de modèle : les composants directement définis n'ont pas de ressource de modèle associée similaire aux modèles d'environnement et de service. Vous fournissez un fichier modèle IaC directement au composant. De même, vous fournissez directement un manifeste qui définit le langage du modèle et le moteur de rendu pour le provisionnement de l'infrastructure du composant. Vous créez le fichier modèle et le manifeste de la même manière que vous créez un ensemble de modèles. Cependant, avec des composants directement définis, il n'est pas nécessaire de stocker les fichiers IaC sous forme de bundles dans des emplacements particuliers, et vous ne devez pas créer de ressource modèle dans AWS Proton des fichiers IaC externes.
- Pas de provisionnement CodeBuild basé : vous ne pouvez pas approvisionner des composants directement définis à l'aide de votre propre script de provisionnement personnalisé, connu sous le nom de provisionnement CodeBuildbasé. Pour de plus amples informations, veuillez consulter the section called “CodeBuild approvisionnement”.

Composants de la AWS Proton console

Utilisez la AWS Proton console pour créer, mettre à jour, afficher et utiliser AWS Proton des composants.

Les pages de console suivantes concernent les composants. Nous incluons des liens directs vers les pages de console de haut niveau.

- Composants : consultez la liste des composants de votre AWS compte. Vous pouvez créer de nouveaux composants et mettre à jour ou supprimer des composants existants. Choisissez le nom d'un composant dans la liste pour afficher sa page de détails.

  Des listes similaires existent également sur les pages Détails de l'environnement et Détails des instances de service. Ces listes affichent uniquement les composants associés à la ressource en cours de consultation. Lorsque vous créez un composant à partir de l'une de ces listes, AWS Proton présélectionne l'environnement associé sur la page Créer un composant.

- Détails du composant : pour afficher la page des détails du composant, choisissez un nom de composant dans la liste des composants.

  Sur la page de détails, consultez les détails et le statut du composant, puis mettez-le à jour ou supprimez-le. Affichez et gérez les listes de sorties (par exemple, les ressources allouées ARNs), les CloudFormation piles provisionnées et les balises attribuées.
- Créer un composant — Créez un composant. Entrez le nom et la description du composant, choisissez les ressources associées, spécifiez le fichier iAc source du composant et attribuez des balises.
- Mettre à jour un composant : pour mettre à jour un composant, sélectionnez-le dans la liste des composants, puis dans le menu Actions, choisissez Mettre à jour le composant. Sur les pages de détails du composant, vous pouvez également sélectionner Mettre à jour.

  Vous pouvez mettre à jour la plupart des détails du composant. Vous ne pouvez pas mettre à jour le nom du composant. Et vous pouvez choisir de redéployer ou non le composant après une mise à jour réussie.
- Configurer l'environnement : lorsque vous créez ou mettez à jour un environnement, vous pouvez spécifier un rôle de composant. Ce rôle contrôle la capacité à exécuter des composants directement définis dans l'environnement et fournit des autorisations pour leur mise en service.
- Créer une nouvelle version de modèle de service : lorsque vous créez une version de modèle de service, vous pouvez spécifier les sources de composants prises en charge pour la version du modèle. Cela contrôle la possibilité d'associer des composants à des instances de service de services en fonction de cette version de modèle.

Composants de l' AWS Proton API et AWS CLI

Utilisez l' AWS Proton API ou le AWS CLI pour créer, mettre à jour, afficher et utiliser AWS Proton des composants.

Les actions d'API suivantes gèrent directement les ressources des AWS Proton composants.

- CreateComponent— Crée un AWS Proton composant.
- DeleteComponent— Supprime un AWS Proton composant.
- GetComponent— Obtenez des données détaillées pour un composant.
- ListComponentOutputs— Obtenez une liste des sorties de l'infrastructure sous forme de code (IaC) des composants.

- ListComponentProvisionedResources— Répertoriez les ressources allouées pour un composant avec des informations détaillées.
- ListComponents— Répertoriez les composants avec des données récapitulatives. Vous pouvez filtrer la liste des résultats par environnement, par service ou par instance de service unique.

Les actions d'API suivantes d'autres AWS Proton ressources comportent certaines fonctionnalités liées aux composants.

- CreateEnvironment, UpdateEnvironment— `componentRoleArn` À utiliser pour spécifier le nom de ressource Amazon (ARN) du rôle de service IAM AWS Proton utilisé lors du provisionnement de composants directement définis dans cet environnement. Il détermine l'étendue de l'infrastructure qu'un composant directement défini peut fournir.
- CreateServiceTemplateVersion— `supportedComponentSources` À utiliser pour spécifier les sources de composants prises en charge. Les composants dont les sources sont prises en charge peuvent être attachés à des instances de service sur la base de cette version de modèle de service.

Questions fréquemment posées sur les composants

Quel est le cycle de vie d'un composant ?

Les composants peuvent être attachés ou détachés. Ils sont conçus pour être attachés à une instance de service et améliorer son infrastructure la plupart du temps. Les composants détachés sont dans un état de transition qui vous permet de supprimer un composant ou de l'associer à une autre instance de service de manière contrôlée et sûre. Pour de plus amples informations, veuillez consulter the section called “États des composants”.

Pourquoi ne puis-je pas supprimer les composants associés ?

Solution : pour supprimer un composant attaché, mettez-le à jour pour le détacher de l'instance de service, validez la stabilité de l'instance de service, puis supprimez le composant.

Pourquoi est-ce nécessaire ? Les composants attachés fournissent l'infrastructure supplémentaire dont votre application a besoin pour exécuter ses fonctions d'exécution. L'instance de service utilise peut-être les sorties des composants pour détecter et utiliser les ressources de cette infrastructure. La suppression du composant, et donc de ses ressources d'infrastructure, peut perturber l'instance de service attachée.

Par mesure de sécurité supplémentaire, AWS Proton vous devez mettre à jour le composant et le détacher de son instance de service avant de pouvoir le supprimer. Vous pouvez ensuite valider votre instance de service pour vous assurer qu'elle continue à se déployer et à fonctionner correctement. Si vous détectez un problème, vous pouvez rapidement rattacher le composant à l'instance de service, puis essayer de le résoudre. Lorsque vous êtes certain que votre instance de service est exempte de toute dépendance à l'égard du composant, vous pouvez le supprimer en toute sécurité.

Pourquoi ne puis-je pas modifier directement l'instance de service attachée à un composant ?

Solution : pour modifier l'attachement, mettez à jour le composant pour le détacher de l'instance de service, validez la stabilité du composant et de l'instance de service, puis attachez le composant à la nouvelle instance de service.

Pourquoi est-ce nécessaire ? Un composant est conçu pour être attaché à une instance de service. Votre composant peut utiliser des entrées d'instance de service pour la dénomination et la configuration des ressources d'infrastructure. La modification de l'instance de service attachée peut perturber le composant (en plus d'une éventuelle interruption de l'instance de service, comme décrit dans la FAQ précédente, Pourquoi ne puis-je pas supprimer mes composants attachés ?). Par exemple, cela peut entraîner le changement de nom, voire le remplacement, des ressources définies dans le modèle IaC du composant.

Par mesure de sécurité supplémentaire, AWS Proton vous devez mettre à jour le composant et le détacher de son instance de service avant de pouvoir l'associer à une autre instance de service. Vous pouvez ensuite valider la stabilité du composant et de l'instance de service avant d'attacher le composant à la nouvelle instance de service.

États des composants

AWS Proton les composants peuvent être dans deux états fondamentalement différents :

- Attaché — Le composant est attaché à une instance de service. Il définit l'infrastructure qui prend en charge les fonctionnalités d'exécution de l'instance de service. Le composant étend l'infrastructure définie dans les modèles d'environnement et de service avec une infrastructure définie par le développeur.

  Un composant typique reste attaché pendant la majeure partie de la partie utile de son cycle de vie.
- Détaché : le composant est associé à un AWS Proton environnement et n'est attaché à aucune instance de service de l'environnement.

Il s'agit d'un état de transition destiné à prolonger la durée de vie d'un composant au-delà d'une seule instance de service.

Le tableau suivant fournit une comparaison de haut niveau des différents états des composants.

	Attaché	Detached
Objectif principal de l'État	Pour étendre l'infrastructure d'une instance de service.	Pour maintenir l'infrastructure du composant entre les pièces jointes des instances de service.
Associé à	Une instance de service et un environnement	Un environnement
Principales propriétés spécifiques	• Nom du service • Nom de l'instance de service • Spec	• Nom de l'environnement
Peut être supprimé	✗ Non	✓ Oui
Peut être mis à jour vers une autre instance de service	✗ Non	✓ Oui
Peut lire les entrées	✓ Oui	✗ Non

L'objectif principal d'un composant est d'être rattaché à une instance de service et d'étendre son infrastructure avec des ressources supplémentaires. Un composant attaché peut lire les entrées de l'instance de service conformément à la spécification. Vous ne pouvez pas supprimer directement le composant ou l'associer à une autre instance de service. Vous ne pouvez pas non plus supprimer son instance de service, ni le service et l'environnement associés. Pour effectuer l'une de ces opérations, mettez d'abord le composant à jour pour le détacher de son instance de service.

Pour maintenir l'infrastructure du composant au-delà de la durée de vie d'une instance de service unique, vous devez mettre à jour le composant et le détacher de son instance de service en supprimant les noms du service et de l'instance de service. Cet état détaché est un état de transition. Le composant ne possède aucune entrée. Son infrastructure reste provisionnée et vous pouvez la mettre à jour. Vous pouvez supprimer les ressources auxquelles le composant était associé lors de son attachement (instance de service, service). Vous pouvez supprimer le composant ou le mettre à jour pour qu'il soit à nouveau rattaché à une instance de service.

Infrastructure de composants sous forme de fichiers de code

Les fichiers d'infrastructure de composants sous forme de code (IaC) sont similaires à ceux des autres AWS Proton ressources. Découvrez ici certains détails spécifiques aux composants. Pour obtenir des informations complètes sur la création de fichiers iAc pour AWS Proton, consultezCréation de modèles et d'ensembles.

Utilisation de paramètres avec des composants

L'espace de noms de AWS Proton paramètres inclut certains paramètres auxquels un fichier IaC de service peut faire référence pour obtenir le nom et les sorties d'un composant associé. L'espace de noms inclut également des paramètres auxquels un fichier iAC de composant peut faire référence pour obtenir des entrées, des sorties et des valeurs de ressources à partir de l'environnement, du service et de l'instance de service auxquels le composant est associé.

Un composant ne possède pas ses propres entrées : il obtient ses entrées de l'instance de service à laquelle il est attaché. Un composant peut également lire les sorties de l'environnement.

Pour plus d'informations sur l'utilisation des paramètres dans les fichiers iAc des composants et des services associés, consultezthe section called “Paramètres du composant CloudFormation IaC”. Pour obtenir des informations générales sur AWS Proton les paramètres et une référence complète de l'espace de noms des paramètres, consultezthe section called “Parameters”.

Création de fichiers iAc robustes

En tant qu'administrateur, lorsque vous créez une version de modèle de service, vous pouvez décider si vous souhaitez autoriser les instances de service créées à partir de la version du modèle à être associées à des composants. Consultez le supportedComponentSourcesparamètre de l'action d'CreateServiceTemplateVersionAPI dans la référence AWS Proton d'API. Toutefois, pour toute future instance de service, la personne qui crée l'instance, décide d'y attacher ou non un composant et (dans le cas de composants définis directement) crée le composant iAC est généralement une

personne différente : un développeur utilisant votre modèle de service. Par conséquent, vous ne pouvez pas garantir qu'un composant sera attaché à une instance de service. Vous ne pouvez pas non plus garantir l'existence de noms de sortie de composants spécifiques ni la validité et la sécurité des valeurs de ces sorties.

AWS Proton et la syntaxe Jinja vous aident à contourner ces problèmes et à créer des modèles de service robustes qui s'affichent sans faille de la manière suivante :

- AWS Proton filtres de paramètres : lorsque vous faites référence aux propriétés de sortie des composants, vous pouvez utiliser des filtres de paramètres, c'est-à-dire des modificateurs qui valident, filtrent et formatent les valeurs des paramètres. Pour plus d'informations et d'exemples, consultez the section called "CloudFormation filtres de paramètres".
- Propriété unique par défaut : lorsque vous faites référence à une seule ressource ou propriété de sortie d'un composant, vous pouvez garantir que le rendu de votre modèle de service n'échouera pas en utilisant le `default` filtre, avec ou sans valeur par défaut. Si le composant, ou un paramètre de sortie spécifique auquel vous faites référence, n'existe pas, la valeur par défaut (ou une chaîne vide, si vous n'avez pas spécifié de valeur par défaut) est rendue à la place, et le rendu réussit. Pour de plus amples informations, veuillez consulter the section called "Fournir des valeurs par défaut".

  Exemples :

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

  > **Note**
  >
  > Ne confondez pas la `.default` partie de l'espace de noms, qui désigne les composants directement définis, avec le `default` filtre, qui fournit une valeur par défaut lorsque la propriété référencée n'existe pas.

- Référence d'objet entière : lorsque vous faites référence à l'ensemble du composant ou à la collection des sorties d'un composant, vous AWS Proton renvoyez un objet vide et vous `{}` garantissez ainsi que le rendu de votre modèle de service n'échouera pas. Vous n'êtes pas obligé d'utiliser de filtre. Assurez-vous de faire la référence dans un contexte qui peut prendre un objet vide ou d'utiliser une `{{ if .. }}` condition pour tester la présence d'un objet vide.

  Exemples :

- `{{ service_instance.components.default }}`
- `{{ service_instance.components.default.outputs }}`

CloudFormation Exemple de composant

Voici un exemple complet d'un composant AWS Proton directement défini et de la façon dont vous pouvez l'utiliser dans un AWS Proton service. Le composant fournit un bucket Amazon Simple Storage Service (Amazon S3) et la politique d'accès associée. L'instance de service peut faire référence à ce compartiment et l'utiliser. Le nom du compartiment est basé sur les noms de l'environnement, du service, de l'instance de service et du composant, ce qui signifie que le compartiment est couplé à une instance spécifique du modèle de composant étendant une instance de service spécifique. Les développeurs peuvent créer plusieurs composants sur la base de ce modèle de composant, afin de fournir des compartiments Amazon S3 pour différentes instances de service et différents besoins fonctionnels.

L'exemple couvre la création des différents fichiers d' CloudFormation infrastructure sous forme de code (IaC) requis et la création d'un rôle requis Gestion des identités et des accès AWS (IAM). L'exemple regroupe les étapes en fonction des rôles des personnes propriétaires.

Étapes de l'administrateur

Pour permettre aux développeurs d'utiliser des composants avec un service

1. Créez un rôle Gestion des identités et des accès AWS (IAM) qui délimite les ressources que les composants directement définis exécutés dans votre environnement peuvent fournir. AWS Proton assume ce rôle ultérieurement pour approvisionner des composants directement définis dans l'environnement.

   Pour cet exemple, utilisez la politique suivante :

   Example rôle de composant directement défini

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

2. Indiquez le rôle que vous avez créé à l'étape précédente lorsque vous créez ou mettez à jour l'environnement. Dans la AWS Proton console, spécifiez un rôle de composant sur la page

Configurer l'environnement. Si vous utilisez l' AWS Proton API ou AWS CLI, spécifiez l'une CreateEnvironmentou l'`componentRoleArn`autre des actions de UpdateEnvironmentl'API.

3. Créez un modèle de service qui fait référence à un composant directement défini attaché à l'instance de service.

   L'exemple montre comment écrire un modèle de service robuste qui ne se brise pas si aucun composant n'est attaché à l'instance de service.

   Example fichier CloudFormation IaC de service utilisant un composant

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

4. Créez une nouvelle version mineure du modèle de service qui déclare que les composants directement définis sont pris en charge.

   - Ensemble de modèles dans Amazon S3 — Dans la AWS Proton console, lorsque vous créez une version de modèle de service, pour Sources de composants prises en charge, sélectionnez Directement défini. Si vous utilisez l' AWS Proton API ou AWS CLI, spécifiez-le `DIRECTLY_DEFINED` dans le `supportedComponentSources` paramètre des actions de l'UpdateServiceTemplateVersionAPI CreateServiceTemplateVersionou.
   - Synchronisation des modèles — Apportez une modification à votre référentiel de packs de modèles de services, que vous spécifiez `supported_component_sources:` en `DIRECTLY_DEFINED` tant qu'élément du `.template-registration.yaml` fichier dans le répertoire des versions principales. Pour plus d'informations sur ce fichier, consultez the section called "Modèles de service de synchronisation".

5. Publiez la nouvelle version mineure du modèle de service. Pour de plus amples informations, veuillez consulter the section called "Publication".
6. Assurez-vous d'autoriser les `proton:CreateComponent` développeurs qui utilisent ce modèle de service à jouer le rôle IAM.

Étapes pour les développeurs

Pour utiliser un composant directement défini avec une instance de service

1. Créez un service qui utilise la version du modèle de service créée par l'administrateur avec le support des composants. Vous pouvez également mettre à jour l'une de vos instances de service existantes pour utiliser la dernière version du modèle.
2. Rédigez un fichier modèle de composant iAC qui approvisionne un compartiment Amazon S3 et une politique d'accès associée et expose ces ressources sous forme de sorties.

   Example fichier CloudFormation iAC du composant

   ```
   # cloudformation.yaml

   # A component that defines an S3 bucket and a policy for accessing the bucket.
   Resources:
     S3Bucket:
       Type: AWS::S3::Bucket
       Properties:
   ```

```
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

3. Si vous utilisez l' AWS Proton API AWS CLI, rédigez un fichier manifeste pour le composant.

   Example manifeste de composant défini directement

   ```
   infrastructure:
     templates:
       - file: "cloudformation.yaml"
         rendering_engine: jinja
         template_language: cloudformation
   ```

4. Créez un composant directement défini. AWS Proton assume le rôle de composant défini par l'administrateur pour provisionner le composant.

   Dans la AWS Proton console, sur la page Composants, choisissez Créer un composant. Pour les paramètres du composant, entrez un nom de composant et une description du composant facultative. Pour Attacher un composant, choisissez Attacher le composant à une instance de service. Sélectionnez votre environnement, votre service et votre instance de service. Pour Source du composant, choisissez CloudFormation, puis choisissez le fichier iAC du composant.

> **Note**
>
> Il n'est pas nécessaire de fournir un manifeste : la console en crée un pour vous.

Si vous utilisez l' AWS Proton API ou AWS CLI utilisez l'action CreateComponentAPI. Définissez un composant `name` et optionnel`description`. Set `environmentName``serviceName`, et`serviceInstanceName`. Définissez `templateSource` et `manifest` vers les chemins des fichiers que vous avez créés.

> **Note**
>
> La spécification d'un nom d'environnement est facultative lorsque vous spécifiez des noms de service et d'instance de service. La combinaison de ces deux éléments est unique dans votre AWS compte et AWS Proton peut déterminer l'environnement à partir de l'instance de service.

5. Mettez à jour votre instance de service pour la redéployer. AWS Proton utilise les sorties de votre composant dans le modèle d'instance de service rendu, pour permettre à votre application d'utiliser le compartiment Amazon S3 approvisionné par le composant.

Utilisation des référentiels git avec AWS Proton

AWS Proton utilise les référentiels git à diverses fins. La liste suivante classe les types de référentiels associés aux AWS Proton ressources. Pour les AWS Proton fonctionnalités qui se connectent à plusieurs reprises à votre dépôt pour y envoyer du contenu ou en extraire du contenu, vous devez enregistrer un lien vers le référentiel AWS Proton dans votre AWS compte. Un lien vers un référentiel est un ensemble de propriétés qui AWS Proton peut être utilisé lorsqu'il se connecte à un référentiel. AWS Proton prend actuellement en charge GitHub, GitHub Enterprise et BitBucket.

Référentiels pour développeurs

- Référentiel de code : référentiel utilisé par les développeurs pour stocker le code de l'application. Utilisé pour le déploiement du code. AWS Proton n'interagit pas directement avec ce dépôt. Lorsqu'un développeur fournit un service qui inclut un pipeline, il fournit le nom du référentiel et la branche à partir de laquelle lire le code de son application. AWS Proton transmet ces informations au pipeline qu'il approvisionne.

  Pour de plus amples informations, veuillez consulter the section called “Créer”.

Référentiels pour administrateurs

- Référentiel de modèles : référentiel dans lequel les administrateurs stockent les ensembles de AWS Proton modèles. Utilisé pour la synchronisation des modèles. Lorsqu'un administrateur crée un modèle dans AWS Proton, il peut pointer vers un référentiel de modèles AWS Proton et synchroniser le nouveau modèle avec celui-ci. Lorsque l'administrateur met à jour le bundle de modèles dans le référentiel, il crée AWS Proton automatiquement une nouvelle version du modèle. Liez un référentiel de modèles à celui-ci AWS Proton avant de pouvoir l'utiliser pour la synchronisation.

  Pour de plus amples informations, veuillez consulter the section called “Configurations de synchronisation de modèles”.

  > **Note**
  >
  > Un référentiel de modèles n'est pas nécessaire si vous continuez à télécharger vos modèles sur Amazon Simple Storage Service (Amazon S3) et que vous appelez AWS Proton la APIs direction des modèles pour créer de nouveaux modèles ou des versions de modèles.

Référentiels de provisionnement autogérés

Référentiel d'infrastructure : référentiel hébergeant des modèles d'infrastructure rendus. Utilisé pour le provisionnement autogéré de l'infrastructure de ressources. Lorsqu'un administrateur crée un environnement pour un provisionnement autogéré, il fournit un référentiel. AWS Proton soumet des pull requests (PRs) à ce référentiel afin de créer l'infrastructure pour l'environnement et pour toute instance de service déployée dans l'environnement. Liez un référentiel d'infrastructure à celui-ci AWS Proton avant de pouvoir l'utiliser pour le provisionnement autogéré de l'infrastructure.

Référentiel de pipelines : référentiel utilisé pour créer des pipelines. Utilisé pour le provisionnement autogéré des pipelines. L'utilisation d'un référentiel supplémentaire pour approvisionner les pipelines permet AWS Proton de stocker les configurations de pipeline indépendamment de tout environnement ou service individuel. Il vous suffit de fournir un référentiel de pipeline unique pour tous vos services de provisionnement autogérés. Liez un référentiel de pipelines à AWS Proton avant de pouvoir l'utiliser pour le provisionnement de pipelines autogéré.

Pour de plus amples informations, veuillez consulter the section called “AWS-provisionnement géré”.

Rubriques

Créez un lien vers votre référentiel

Vous pouvez créer un lien vers votre référentiel à l'aide de la console ou de la CLI. Lorsque vous créez un lien vers un référentiel, vous AWS Proton créez un rôle lié à un service.

AWS Management Console

Créez un lien vers votre référentiel comme indiqué dans les étapes de console suivantes.

1. Dans la AWS Proton console, choisissez Repositories.
2. Choisissez Créer un référentiel.
3. Sur la page Associer un nouveau référentiel, dans la section Détails du référentiel :

a. Choisissez votre fournisseur de dépôt.

b. Choisissez l'une de vos connexions existantes. Si vous n'en avez pas, choisissez Ajouter une nouvelle CodeStar connexion pour créer une connexion, puis revenez à la AWS Proton console, actualisez la liste des connexions et choisissez votre nouvelle connexion.

c. Choisissez parmi vos référentiels de code source connectés.

4. [facultatif] Dans la section Balises, choisissez Ajouter une nouvelle balise une ou plusieurs fois, puis entrez les paires clé et valeur.
5. Choisissez Créer un référentiel.
6. Consultez les données détaillées de votre référentiel lié.

AWS CLI

Créez et enregistrez un lien vers votre référentiel.

Exécutez la commande suivante :

```
$ aws proton create-repository \
    --name myrepos/environments \
    --connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
    --provider "GITHUB" \
    --encryption-key "arn:aws:kms:region-id:123456789012:key/bPxRfiCYEXAMPLEKEY" \
    --tags key=mytag1,value=value1 key=mytag2,value=value2
```

Les deux derniers paramètres, --encryption-key et--tags, sont facultatifs.

Réponse :

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
        "connectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/2ad03b28-a7c4-EXAMPLE11111",
        "encryptionKey": "arn:aws:kms:region-id:123456789012:key/
bPxRfiCYEXAMPLEKEY",
        "name": "myrepos/environments",
```

```
        "provider": "GITHUB"
    }
}
```

Après avoir créé un lien vers un référentiel, vous pouvez afficher une liste de AWS balises gérées par le client, comme illustré dans l'exemple de commande suivant. AWS Proton génère automatiquement des tags AWS gérés pour vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour de plus amples informations, veuillez consulter AWS Proton ressources et balisage.

Commande :

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments"
```

Afficher les données du référentiel lié

Vous pouvez répertorier et afficher les détails des référentiels liés à l'aide de la console ou du AWS CLI. Pour les liens de dépôt utilisés pour synchroniser les référentiels git AWS Proton, vous pouvez récupérer la définition et le statut de synchronisation du référentiel à l'aide du AWS CLI.

AWS Management Console

Répertoriez et affichez les détails du référentiel lié à l'aide de la AWS Proton console.

1. Pour répertorier vos référentiels liés, choisissez Repositories dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un référentiel.

AWS CLI

Répertoriez vos référentiels liés.

Exécutez la commande suivante :

```
$ aws proton list-repositories
```

Réponse :

```
{
    "repositories": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
            "name": "myrepos/templates",
            "provider": "GITHUB"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
            "name": "myrepos/environments",
            "provider": "GITHUB"
        }
    ]
}
```

Afficher les détails d'un dépôt lié.

Exécutez la commande suivante :

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

Réponse :

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Répertoriez vos référentiels synchronisés.

L'exemple suivant répertorie les référentiels que vous avez configurés pour la synchronisation des modèles.

Exécutez la commande suivante :

```
$ aws proton list-repository-sync-definitions \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Afficher l'état de synchronisation du référentiel.

L'exemple suivant permet de récupérer l'état de synchronisation d'un modèle de référentiel de synchronisation.

Exécutez la commande suivante :

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Réponse :

```
{
    "latestSync": {
        "events": [
            {
                "event": "Clone started",
                "time": "2021-11-21T00:26:35.883000+00:00",
                "type": "CLONE_STARTED"
            },
            {
                "event": "Updated configuration",
                "time": "2021-11-21T00:26:41.894000+00:00",
                "type": "CONFIG_UPDATED"
            },
            {
                "event": "Starting syncs for commit 62c03ff86eEXAMPLE1111111",
                "externalId": "62c03ff86eEXAMPLE1111111",
                "time": "2021-11-21T00:26:44.861000+00:00",
                "type": "STARTING_SYNC"
            }
        ],
        "startedAt": "2021-11-21T00:26:29.728000+00:00",
```

```
        "status": "SUCCEEDED"
    }
}
```

Suppression d'un lien de référentiel

Vous pouvez supprimer un lien vers un dépôt à l'aide de la console ou du AWS CLI.

> **Note**
>
> La suppression d'un lien vers un dépôt entraîne uniquement la suppression du lien AWS Proton enregistré qui se trouve dans votre AWS compte. Il ne supprime aucune information de votre référentiel.

AWS Management Console

Supprimez un lien vers un dépôt à l'aide de la console.

Sur la page détaillée du référentiel.

1. Dans la AWS Proton console, choisissez Repositories.
2. Dans la liste des référentiels, cliquez sur le bouton radio situé à gauche du référentiel que vous souhaitez supprimer.
3. Sélectionnez Delete (Supprimer).
4. Un modal vous invite à confirmer l'action Supprimer.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Supprimez un lien vers un dépôt.

Exécutez la commande suivante :

```
$ aws proton delete-repository \
    --name myrepos/templates \
    --provider"GITHUB"
```

Réponse :

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Surveillance AWS Proton

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS Proton et des performances de vos AWS solutions. La section suivante décrit les outils de surveillance que vous pouvez utiliser AWS Proton.

Automatisez AWS Proton avec EventBridge

Vous pouvez suivre AWS Proton les événements sur Amazon EventBridge. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, applications software-as-a-service (SaaS) et Services AWS. Vous pouvez configurer des événements pour répondre aux modifications de l'état des AWS ressources. EventBridge achemine ensuite ces données vers des services cibles tels qu' AWS Lambda Amazon Simple Notification Service. Ces événements sont les mêmes que ceux qui apparaissent dans Amazon CloudWatch Events. CloudWatch Les événements fournissent un flux d'événements système en temps quasi réel qui décrivent les modifications apportées aux AWS ressources. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

EventBridge À utiliser pour être informé des changements d'état dans les flux de travail de AWS Proton provisionnement.

Types d’événements

Les événements sont composés de règles qui incluent un modèle d'événements et des cibles. Vous configurez une règle en choisissant le modèle d'événement et les objets cibles :

Modèle d’événement

Chaque règle est exprimée sous la forme d'un modèle d'événements avec la source et le type d'événements à surveiller ainsi que les cibles des événements. Pour surveiller les événements, vous créez une règle avec le service que vous surveillez comme source d'événements. Par exemple, vous pouvez créer une règle avec un modèle d'événement utilisé AWS Proton comme source d'événement pour déclencher une règle en cas de modification de l'état d'un déploiement.

Targets

La règle reçoit un service sélectionné comme cible de l'événement. Vous pouvez configurer un service cible pour envoyer des notifications, capturer des informations d'état, prendre des mesures correctives, initier des événements ou prendre d'autres mesures.

Les objets d'événement contiennent des champs standard tels que l'ID, le compte Région AWS, le type de détail, la source, la version, la ressource, l'heure (facultatif). Le champ de détail est un objet imbriqué contenant des champs personnalisés pour l'événement.

AWS Proton les événements sont émis dans la mesure du possible. La fourniture de tous les efforts signifie que le service tente d'envoyer tous les événements à EventBridge, mais dans de rares cas, un événement peut ne pas être livré.

Pour chaque AWS Proton ressource pouvant émettre des événements, le tableau suivant répertorie la valeur du type de détail, les champs de détail et (le cas échéant) une référence à une liste de valeurs pour les champs de `previousStatus` détail `status` et de détail. Lorsqu'une ressource est supprimée, la valeur du champ de `status` détail est`DELETED`.

Ressource	Valeur du type de détail	Champs de détail

`EnvironmentTemplate`	AWS Proton Modification du statut du modèle d'environnement	`name`
		`status`
		`previousStatus`

`EnvironmentTemplateVersion`	AWS Proton Modification de l'état de la version du modèle d'environnement	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		valeurs de statut

`ServiceTemplate`	AWS Proton Modification du statut du modèle de service	`name` `status`

Ressource	Valeur du type de détail	Champs de détail
		`previousStatus`

`ServiceTemplateVersion`	AWS Proton Modification du statut de la version du modèle de service	`name`
		
		`majorVersion`
		
		`minorVersion`
		
		`status`
		
		`previousStatus`
		
		valeurs de statut

`Environment`	AWS Proton Modification de l'état de l'environnement	`name`
		
		`status`
		
		`previousStatus`

`Service`	AWS Proton Modification de l'état du service	`name`
		
		`status`
		
		`previousStatus`
		
		valeurs de statut

Ressource	Valeur du type de détail	Champs de détail

`ServiceInstance`	AWS Proton Modification du statut de l'instance de service	`name`
		`serviceName`
		`status`
		`previousStatus`

`ServicePipeline`	AWS Proton Modification de l'état du pipeline de services	`serviceName`
		`status`
		`previousStatus`

`EnvironmentAccountConnection`	AWS Proton Modification de l'état de connexion au compte d'environnement	`id`
		`status`
		`previousStatus`
		valeurs de statut

`Component`	AWS Proton Modification de l'état du composant	`name`
		`status`
		`previousStatus`

AWS Proton exemples d'événements

Les exemples suivants montrent AWS Proton comment envoyer des événements à EventBridge.

Modèle de service

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name"],
    "detail": {
        "name": "sample-service-template-name",
        "status": "PUBLISHED",
        "previousStatus": "DRAFT"
    }
}
```

Version du modèle de service

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Version Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name:1.0"],
    "detail": {
        "name": "sample-service-template-name",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_FAILED",
        "previousStatus": "REGISTRATION_IN_PROGRESS"
    }
}
```

Environnement

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Environment Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:environment/sample-
environment"],
    "detail": {
        "name": "sample-environment",
        "status": "DELETE_FAILED",
        "previousStatus": "DELETE_IN_PROGRESS"
```

```
    }
}
```

EventBridgeTutorial: Envoyer des alertes Amazon Simple Notification Service en cas de modification AWS Proton de l'état du service

Dans ce didacticiel, vous allez utiliser une règle d'événement AWS Proton préconfigurée qui capture les modifications de statut de votre AWS Proton service. EventBridgeenvoie les modifications de statut à une rubrique Amazon SNS. Vous vous abonnez au sujet et Amazon SNS vous envoie des e-mails de modification du statut de votre AWS Proton service.

Conditions préalables

Vous avez un AWS Proton service existant avec un `Active` statut. Dans le cadre de ce didacticiel, vous pouvez ajouter des instances de service à ce service, puis les supprimer.

Si vous devez créer un AWS Proton service, consultezPrise en main. Pour plus d'informations, consultez AWS Proton quotas et the section called "Modifier".

Étape 1 : Créer une rubrique Amazon SNS et s'y abonner

Créez une rubrique Amazon SNS qui servira de cible d'événement pour la règle d'événement que vous créez à l'étape 2.

Créer une rubrique Amazon SNS

1. Connectez-vous et ouvrez la console Amazon SNS.
2. Dans le volet de navigation, choisissez Rubriques, puis Créer une rubrique.
3. Dans la page Créer un sujet :
    a. Choisissez le type Standard.
    b. Dans le champ Nom, entrez **`tutorial-service-status-change`** et choisissez Créer un sujet.
4. Sur la page tutorial-service-status-changedétaillée, choisissez Créer un abonnement.
5. Sur la page Créer un abonnement :

a. Pour Protocole, choisissez E-mail.

b. Dans le champ Endpoint (Point de terminaison), saisissez l'adresse e-mail à laquelle vous avez actuellement accès et choisissez Create subscription (Créer un abonnement).

6. Vérifiez votre compte de messagerie et attendez de recevoir un e-mail de confirmation de l'abonnement. Lorsque vous le recevez, ouvrez-le et choisissez Confirmer l'abonnement.

Étape 2 : Enregistrer une règle d'événement

Enregistrez une règle d'événement qui capture les modifications de statut de votre AWS Proton service. Pour de plus amples informations, veuillez consulter Conditions préalables.

Créez une règle d'événement.

1. Ouvrez la EventBridge console Amazon.
2. Dans le volet de navigation, choisissez Events, Rules.
3. Sur la page Règles, dans la section Règles, choisissez Créer une règle.
4. Sur la page Créer une règle :

   a. Dans la section Nom et description, pour Nom, entrez**`tutorial-rule`**.

   b. Dans la section Définir le modèle, choisissez le modèle d'événement.

      i. Dans Modèle de correspondance d'événement, choisissez Prédéfini par un service.

      ii. Pour Service Provider (Fournisseur de service), sélectionnez AWS.

      iii. Pour Nom du service, choisissez AWS Proton.

      iv. Pour Type d'événement, choisissez Changement AWS Proton de statut du service.

      Le modèle d'événement apparaît dans un éditeur de texte.

      v. Ouvrez la AWS Proton console.

      vi. Dans le panneau de navigation, choisissez Services.

      vii. Dans la page Services, choisissez le nom de votre AWS Proton service.

      viii. Sur la page des détails du service, copiez le nom de ressource Amazon (ARN) du service.

      ix. Revenez à la EventBridge console et à la règle de votre didacticiel, puis choisissez Modifier dans l'éditeur de texte.

x. Dans l'éditeur de texte, pour`"resources":`, entrez l'ARN du service que vous avez copié à l'étape viii.

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

xi. Enregistrez le modèle d'événement.

c. Dans la section Sélectionner les cibles :

i. Pour Target (Cible), choisissez SNS topic (Rubrique SNS).

ii. Dans le champ Sujet, sélectionnez tutorial-service-status-change.

d. Choisissez Créer.

Étape 3 : Testez la règle de votre événement

Vérifiez que votre règle d'événement fonctionne en ajoutant une instance à votre AWS Proton service.

1. Passez à la AWS Proton console.
2. Dans le panneau de navigation, choisissez Services.
3. Dans la page Services, choisissez le nom de votre service.
4. Sur la page Détails du service, choisissez Modifier.
5. Dans la page Configurer le service, choisissez Next.
6. Dans la page Configurer les paramètres personnalisés, dans la section Instances de service, choisissez Ajouter une nouvelle instance.
7. Complétez le formulaire pour votre nouvelle instance :

a. Entrez un nom pour votre nouvelle instance.

b. Sélectionnez les mêmes environnements compatibles que ceux que vous avez choisis pour vos instances existantes.

c. Entrez des valeurs pour les entrées requises.

d. Choisissez Suivant.

8. Passez en revue vos entrées et choisissez Mettre à jour.
9. Une fois l'état du service activé`Active`, consultez vos e-mails pour vérifier que vous avez reçu AWS des notifications contenant des mises à jour du statut.

```
{
    "version": "0",
    "id": "af76c382-2b3c-7a0a-cf01-936dff228276",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:40:16Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "ACTIVE",
        "status": "UPDATE_IN_PROGRESS",
        "name": "your-service"
    }
}
```

```
{
    "version": "0",
    "id": "87131e29-ad95-bda2-cd30-0ce825dfb0cd",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:42:27Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "UPDATE_IN_PROGRESS",
        "status": "ACTIVE",
        "name": "your-service"
    }
}
```

Étape 4 : nettoyer

Supprimez votre rubrique Amazon SNS et votre abonnement, puis supprimez votre EventBridge règle.

Supprimez votre rubrique Amazon SNS et votre abonnement.

1. Accédez à la console Amazon SNS.
2. Dans le volet de navigation, choisissez Abonnements.
3. Sur la page Abonnements, choisissez l'abonnement que vous avez souscrit à la rubrique nommée, `tutorial-service-status-change` puis choisissez Supprimer.
4. Dans le panneau de navigation, sélectionnez Sujets.
5. Sur la page Sujets, choisissez le sujet nommé, `tutorial-service-status-change` puis cliquez sur Supprimer.
6. Un modal vous invite à vérifier la suppression. Suivez les instructions et choisissez Supprimer.

Supprimez votre EventBridge règle.

1. Accédez à la EventBridge console Amazon.
2. Dans le volet de navigation, choisissez Events, Rules.
3. Sur la page Règles, choisissez la règle nommée, `tutorial-rule` puis sélectionnez Supprimer.
4. Un modal vous invite à vérifier la suppression. Sélectionnez Delete (Supprimer).

Supprimez l'instance de service ajoutée.

1. Accédez à la console AWS Proton.
2. Dans le panneau de navigation, choisissez Services.
3. Sur la page Services, choisissez le nom de votre service.
4. Sur la page des détails du service, choisissez Modifier, puis Suivant.
5. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, choisissez Supprimer pour l'instance de service que vous avez créée dans le cadre de ce didacticiel, puis cliquez sur Suivant.
6. Passez en revue vos entrées et choisissez Mettre à jour.
7. Un modal vous invite à vérifier la suppression. Suivez les instructions et choisissez Oui, supprimer.

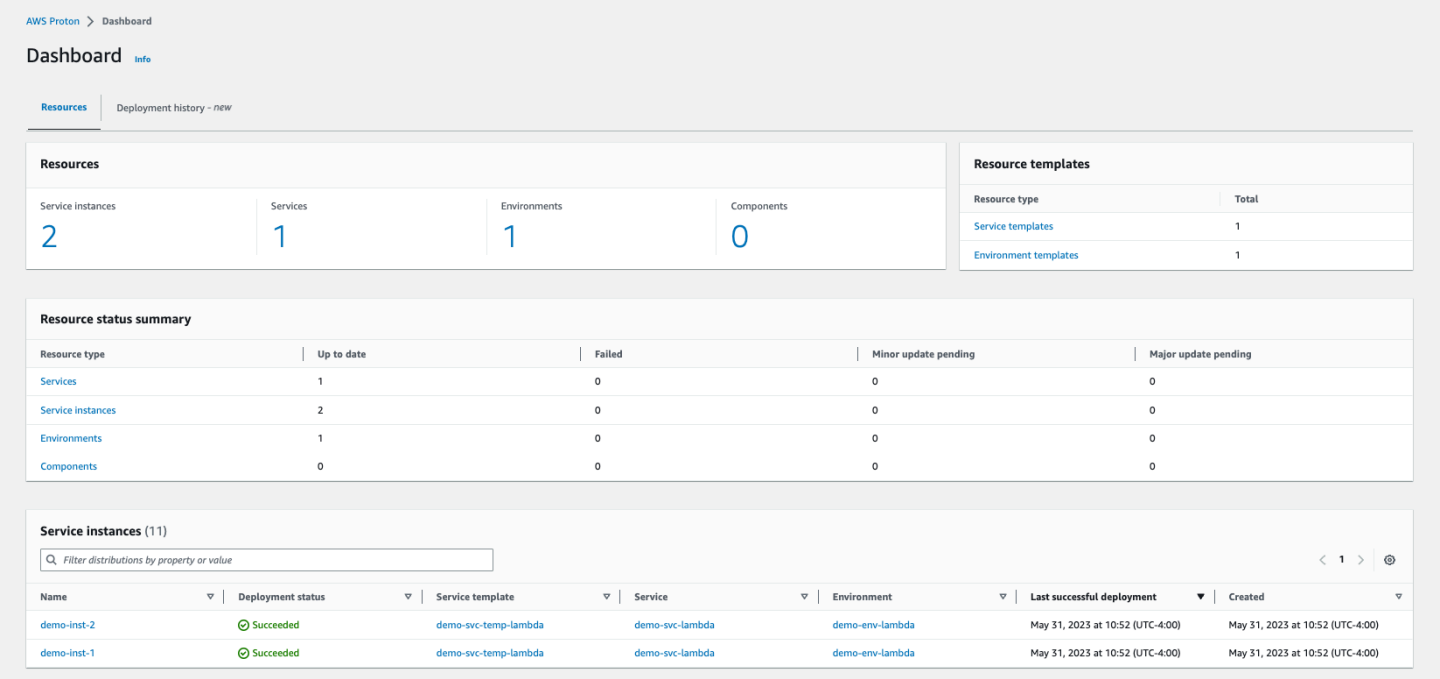
Maintenez l'infrastructure à jour grâce au AWS Proton tableau de bord

Le AWS Proton tableau de bord fournit un résumé des AWS Proton ressources de votre AWS compte, en mettant particulièrement l'accent sur l'obsolescence, c'est-à-dire sur la manière dont les ressources déployées sont mises à jour. Une ressource déployée est à jour lorsqu'elle utilise la version recommandée du modèle qui lui est associé. Une ressource out-of-date déployée peut nécessiter une mise à jour majeure ou mineure de la version du modèle.

Afficher le tableau de bord dans la AWS Proton console

Pour afficher le AWS Proton tableau de bord, ouvrez la AWS Proton console, puis, dans le volet de navigation, sélectionnez Tableau de bord.

Ressources

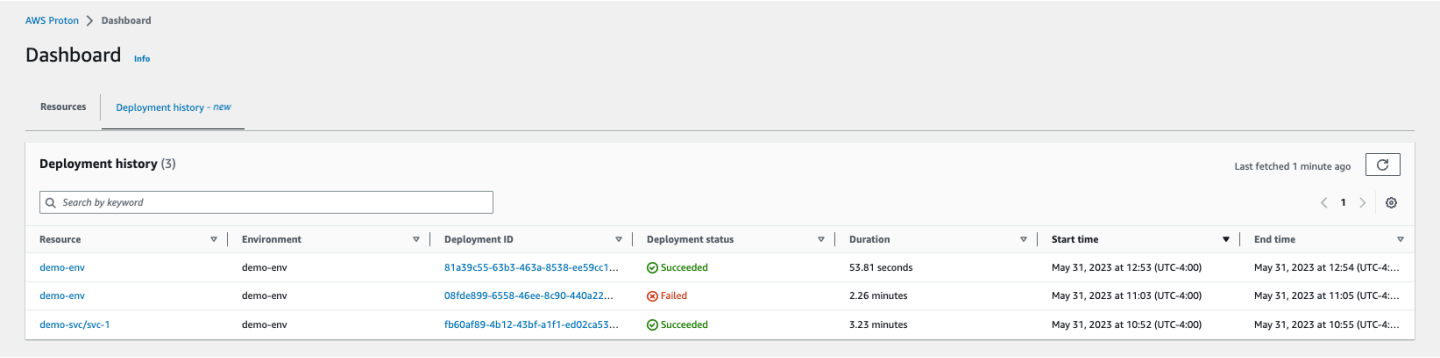


Le premier onglet du tableau de bord affiche le décompte de toutes les ressources de votre compte. L'onglet Ressources indique le nombre d'instances de service, de services, d'environnements et de composants, ainsi que vos modèles de ressources. Il décompose également le nombre de ressources pour chaque type de ressource déployé en fonction de l'état des ressources de ce type. Un tableau des instances de service présente les détails de chaque instance de service : son statut de déploiement, les AWS Proton ressources auxquelles elle est associée, les mises à jour disponibles et certains horodatages.

Vous pouvez filtrer la liste des instances de service en fonction de n'importe quelle propriété de table. Par exemple, vous pouvez filtrer pour voir les instances de service dont les déploiements se situent dans une fenêtre de temps spécifique, ou les instances de service obsolètes par rapport aux recommandations de versions majeures ou mineures.

Choisissez le nom d'une instance de service pour accéder à la page détaillée de l'instance de service, où vous pouvez agir pour effectuer les mises à jour de version appropriées. Choisissez un autre nom de AWS Proton ressource pour accéder à sa page détaillée, ou choisissez un type de ressource pour accéder à la liste de ressources correspondante.

Historique des déploiements



L'onglet Historique des déploiements vous permet de consulter les détails de vos déploiements. Dans le tableau de l'historique des déploiements, vous pouvez suivre l'état du déploiement, ainsi que l'environnement et l'ID de déploiement. Vous pouvez choisir le nom de la ressource ou l'ID de déploiement pour obtenir encore plus de détails, tels qu'un message d'état du déploiement et les sorties de ressources. Le tableau vous permet également de filtrer selon n'importe quelle propriété du tableau.

Sécurité dans AWS Proton

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. Des auditeurs tiers 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 Proton, 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 AWS Proton. Les rubriques suivantes expliquent comment configurer AWS Proton pour répondre à 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 AWS Proton ressources.

Rubriques

- Prévention du problème de l’adjoint confus entre services
- CodeBuild fourniture d'un support Amazon VPC personnalisé

Identity and Access Management pour AWS Proton

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 AWS Proton les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion de l’accès à l’aide de politiques
- Comment AWS Proton fonctionne avec IAM
- Exemples de politiques pour AWS Proton
- AWS politiques gérées pour AWS Proton
- Utilisation de rôles liés à un service pour AWS Proton
- Résolution des problèmes AWS Proton d'identité et d'accès

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 Proton d'identité et d'accès)
- Administrateur du service : déterminez l’accès des utilisateurs et soumettez les demandes d’autorisation (voir Comment AWS Proton fonctionne avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l’accès (voir Exemples de politiques basées sur l'identité pour AWS Proton)

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.

Politiques basées sur l'identité

Les stratégies basées sur l'identité sont des documents de stratégie d'autorisations 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.

Les politiques basées sur l'identité 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.

Politiques basées sur les ressources

Les politiques basées sur les ressources 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.

Les politiques basées sur les ressources sont des politiques en ligne situées 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 (SCPs) — Spécifiez 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 (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir Politiques de contrôle des ressources (RCPs) dans le guide de AWS Organizations l'utilisateur.

- 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 AWS Proton fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à AWS Proton, découvrez les fonctionnalités IAM disponibles. AWS Proton

Fonctionnalités IAM que vous pouvez utiliser avec AWS Proton

Fonctionnalité IAM	AWS Proton soutien
Politiques basées sur l’identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique	Oui
ACLs	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d’identification temporaires	Oui
Autorisations de principal	Oui
Rôles de service	Oui
Rôles liés à un service	Oui

Pour obtenir une vue d'ensemble du fonctionnement de la plupart des fonctionnalités IAM AWS Proton et des autres Services AWS fonctionnalités, consultez le Services AWS guide de l'utilisateur IAM consacré à leur utilisation.

Politiques basées sur l'identité pour AWS Proton

Prend en charge les politiques basées sur l'identité : oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. 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.

Exemples de politiques basées sur l'identité pour AWS Proton

Pour consulter des exemples de politiques AWS Proton basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Proton

Politiques basées sur les ressources au sein de AWS Proton

Prend en charge les politiques basées sur les ressources : non

Les politiques basées sur les ressources 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 AWS Proton

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 AWS Proton actions, reportez-vous à la section Actions définies par AWS Proton dans la référence d'autorisation de service.

Les actions de politique en AWS Proton cours utilisent le préfixe suivant avant l'action :

```
proton
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `List`, incluez l'action suivante :

```
"Action": "proton:List*"
```

Pour consulter des exemples de politiques AWS Proton basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Proton

Ressources politiques pour AWS Proton

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 AWS Proton ressources et leurs caractéristiques ARNs, consultez la section Ressources définies par AWS Proton dans la référence d'autorisation de service. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par AWS Proton.

Pour consulter des exemples de politiques AWS Proton basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Proton

Clés de conditions de politique pour AWS Proton

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 AWS Proton condition, reportez-vous à la section Clés de condition pour AWS Proton 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 Proton.

Pour consulter un exemple condition-key-based de politique visant à limiter l'accès à une ressource, consultezExemples de politiques basées sur des clés conditionnelles pour AWS Proton.

Listes de contrôle d'accès (ACLs) dans AWS Proton

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Les listes de contrôle d'accès (ACLs) sont des listes de bénéficiaires que vous pouvez associer aux ressources. Elles accordent des autorisations aux comptes pour accéder aux ressources auxquelles ils sont associés.

Contrôle d'accès basé sur les attributs (ABAC) avec AWS Proton

Prise en charge d'ABAC (balises dans les politiques) : Oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des 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.

Pour plus d'informations sur le balisage AWS Proton des ressources, consultezAWS Proton ressources et balisage.

Utilisation d'informations d'identification temporaires avec AWS Proton

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.

Autorisations principales interservices pour AWS Proton

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 AWS Proton

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.

Pour de plus amples informations, veuillez consulter AWS Proton Exemples de politiques relatives aux rôles de service IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber AWS Proton les fonctionnalités. Modifiez les rôles de service uniquement lorsque AWS Proton vous recevez des instructions à cet effet.

Rôles liés à un service pour AWS Proton

Prend en charge les rôles liés à un service : oui

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre

Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour de plus amples informations, veuillez consulter Utilisation de rôles liés à un service pour AWS Proton.

Exemples de politiques pour AWS Proton

Vous trouverez des exemples de politiques AWS Proton IAM dans les sections suivantes.

Rubriques

Exemples de politiques basées sur l'identité pour AWS Proton

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources AWS Proton . 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 AWS Proton, y compris le format de ARNs pour chacun des types de ressources, voir Actions, ressources et clés de condition AWS Proton dans la référence d'autorisation de service.

Rubriques

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer AWS Proton des ressources 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.

Liens vers des exemples de politiques basées sur l'identité pour AWS Proton

Liens vers des exemples de politiques basées sur l'identité pour AWS Proton

- AWS politiques gérées pour AWS Proton
- AWS Proton Exemples de politiques relatives aux rôles de service IAM
- Exemples de politiques basées sur des clés conditionnelles pour AWS Proton

AWS Proton Exemples de politiques relatives aux rôles de service IAM

Les administrateurs possèdent et gèrent les ressources AWS Proton créées conformément aux modèles d'environnement et de service. Ils associent à leur compte des rôles de service IAM qui leur permettent AWS Proton de créer des ressources en leur nom. Les administrateurs fournissent les rôles et les AWS Key Management Service clés IAM pour les ressources qui sont ensuite détenues et gérées par les développeurs lors du AWS Proton déploiement de leur application en tant que AWS Proton service dans un AWS Proton environnement. Pour plus d'informations sur le chiffrement des données AWS KMS et le chiffrement des données, consultezProtection des données dans AWS Proton.

Un rôle de service est un rôle Amazon Web Services (IAM) qui permet AWS Proton d'appeler des ressources en votre nom. Si vous spécifiez un rôle de service, AWS Proton utilise les informations d'identification du rôle. Utilisez un rôle de service pour spécifier explicitement les actions qui AWS Proton peuvent être effectuées.

Le rôle de service et sa politique d'autorisation sont créés à partir du service IAM. Pour plus d'informations sur la création d'un rôle de service, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.

AWS Proton rôle de service pour le provisionnement à l'aide CloudFormation

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de AWS Proton service et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de CloudFormation service de l'environnement (`protonServiceRoleArn`paramètre de l'action d'CreateEnvironmentAPI). Ce rôle permet d' AWS Proton effectuer des appels d'API vers d'autres services en votre nom lorsque l'environnement ou l'une des instances de service qui y sont exécutées utilise le provisionnement AWS géré et AWS CloudFormation pour provisionner l'infrastructure.

Nous vous recommandons d'utiliser le rôle IAM et la politique de confiance suivants pour votre rôle AWS Proton de service. Lorsque vous utilisez la AWS Proton console pour créer un environnement et que vous choisissez de créer un nouveau rôle, cette politique s' AWS Proton ajoute au rôle de service qu'elle crée pour vous. Lorsque vous délimitez l'autorisation associée à cette politique, gardez à l'esprit qu'elle AWS Proton échoue en cas d'`Access Denied`erreur.

> **Important**
>
> Sachez que les politiques présentées dans les exemples suivants accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos environnements.

AWS Proton exemple de politique de rôle de service pour CloudFormation

*123456789012*Remplacez-le par votre Compte AWS identifiant.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
```

```
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "NotAction": [
        "organizations:*",
        "account:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeOrganization",
        "account:ListRegions"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton politique de confiance en matière de services

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Politique de rôle du service AWS de provisionnement géré et limitée

Voici un exemple de politique de rôle de AWS Proton service délimitée que vous pouvez utiliser si vous n'avez besoin de AWS Proton services que pour provisionner des ressources S3.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
```

```
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton rôle de service pour le CodeBuild provisionnement

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de AWS Proton service et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de CodeBuild service de l'environnement (`codebuildRoleArn`paramètre de l'action d'CreateEnvironmentAPI). Ce rôle permet d' AWS Proton effectuer des appels d'API vers d'autres services en votre nom lorsque l'environnement ou l'une des instances de service qui y sont exécutées utilise le CodeBuild provisionnement pour provisionner l'infrastructure.

Lorsque vous utilisez la AWS Proton console pour créer un environnement et que vous choisissez de créer un nouveau rôle, AWS Proton une politique avec des privilèges d'administrateur est ajoutée au rôle de service créé pour vous. Lorsque vous créez votre propre rôle et que vous limitez les autorisations, gardez à l'esprit que cela AWS Proton échoue en cas d'`Access Denied`erreur.

> **Important**
>
> Sachez que les politiques associées aux AWS Proton rôles qu'il crée pour vous accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos environnements.

AWS Proton exemple de politique de rôle de service pour CodeBuild

L'exemple suivant fournit des autorisations permettant de CodeBuild provisionner des ressources à l'aide du AWS Cloud Development Kit (AWS CDK).

*123456789012*Remplacez-le par votre Compte AWS identifiant.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
   {
     "Action": [
       "logs:CreateLogStream",
       "logs:CreateLogGroup",
       "logs:PutLogEvents"
     ],
     "Resource": [
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
 Shell-*",
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
 Shell-*:*"
     ],
     "Effect": "Allow"
```

```
    },
    {
      "Action": "proton:NotifyResourceDeploymentStatusChange",
      "Resource": "arn:aws:proton:us-east-1:123456789012:*",
      "Effect": "Allow"
    },
    {
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::123456789012:role/cdk-*-deploy-role-*",
        "arn:aws:iam::123456789012:role/cdk-*-file-publishing-role-*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS Proton CodeBuild politique de confiance

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "CodeBuildTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS Proton rôles de service de pipeline

Pour provisionner des pipelines de services, vous AWS Proton devez disposer d'autorisations pour effectuer des appels d'API vers d'autres services. Les rôles de service requis sont similaires à ceux que vous fournissez lorsque vous créez des environnements. Cependant, les rôles de création de pipelines sont partagés entre tous les services de votre AWS compte, et vous fournissez ces rôles sous forme de paramètres de compte dans la console ou via l'action d'UpdateAccountSettingsAPI.

Lorsque vous utilisez la AWS Proton console pour mettre à jour les paramètres du compte et que vous choisissez de créer un nouveau rôle pour les rôles de CodeBuild service CloudFormation ou pour les rôles de service, les politiques qui s' AWS Proton ajoutent aux rôles de service qu'elle crée pour vous sont les mêmes que celles décrites dans les sections précédentes, AWS-rôle de provisionnement géré etCodeBuild rôle de provisionnement. Lorsque vous délimitez l'autorisation associée à cette politique, gardez à l'esprit qu'elle AWS Proton échoue en cas d'`Access Denied`erreur.

> **Important**
>
> Sachez que les exemples de politiques présentés dans les sections précédentes accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos pipelines.

AWS Proton rôle du composant

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de AWS Proton service et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de CloudFormation composant de l'environnement (`componentRoleArn`paramètre de l'action d'CreateEnvironmentAPI). Ce rôle limite l'infrastructure que les composants directement définis peuvent fournir. Pour plus d'informations sur les composants, consultezÉléments.

L'exemple de politique suivant prend en charge la création d'un composant directement défini qui approvisionne un compartiment Amazon Simple Storage Service (Amazon S3) et une politique d'accès associée.

AWS Proton exemple de politique de rôle de composant

*123456789012*Remplacez-le par votre Compte AWS identifiant.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
```

```
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS Proton politique de confiance en matière de composants

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Exemples de politiques basées sur des clés conditionnelles pour AWS Proton

L'exemple de politique IAM suivant refuse l'accès aux AWS Proton actions correspondant aux modèles spécifiés dans le `Condition` bloc. Notez que ces clés de condition ne sont prises en charge que par les actions répertoriées dans Actions, ressources et clés de condition pour AWS Proton. Pour gérer les autorisations relatives à d'autres actions, par

exemple`DeleteEnvironmentTemplate`, vous devez utiliser le contrôle d'accès au niveau des ressources.

Exemple de politique qui refuse les actions d'un AWS Proton modèle sur un modèle spécifique :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:EnvironmentTemplate":
 ["arn:aws:proton:region_id:123456789012:environment-template/my-environment-
template"]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 ["arn:aws:proton:region_id:123456789012:service-template/my-service-template"]
                }
            }
        }
    ]
}
```

Dans l'exemple de politique suivant, la première instruction au niveau des ressources refuse l'accès aux actions du AWS Proton modèle autres que celles `ListServiceTemplates` qui correspondent au modèle de service répertorié dans le `Resource` bloc. La deuxième déclaration refuse l'accès aux AWS Proton actions correspondant au modèle répertorié dans le `Condition` bloc.

Exemple de politique refusant AWS Proton les actions correspondant à un modèle spécifique :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
        },
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate": [
                        "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
                    ]
                }
            }
        }
    ]
}
```

Le dernier exemple de politique autorise AWS Proton les actions du développeur qui correspondent au modèle de service spécifique répertorié dans le `Condition` bloc.

Exemple de politique permettant aux AWS Proton développeurs d'effectuer des actions correspondant à un modèle spécifique :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListServices",
                "proton:ListServiceInstances",
                "proton:ListEnvironments",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetEnvironment",
                "proton:CreateService",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "proton:DeleteService",
                "codestar-connections:ListConnections"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 "arn:aws:proton:region_id:123456789012:service-template/my-service-template"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService":
 "proton.amazonaws.com"
```

```
                }
            }
        }

    ]
}
```

AWS politiques gérées pour AWS Proton

Pour ajouter des autorisations aux utilisateurs, aux groupes et aux rôles, il est plus facile d'utiliser des politiques AWS gérées que de les rédiger vous-même. Il faut du temps et de l'expertise pour créer des politiques gérées par le client IAM qui ne fournissent à votre équipe que les autorisations dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques AWS gérées. Ces politiques couvrent des cas d'utilisation courants et sont disponibles dans votre Compte AWS. Pour plus d'informations sur les politiques AWS gérées, voir les politiques AWS gérées dans le guide de l'utilisateur IAM.

Services AWS maintenir et mettre à jour les politiques AWS gérées. Vous ne pouvez pas modifier les autorisations dans les politiques AWS gérées. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (utilisateurs, groupes et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique AWS gérée. Les mises à jour des politiques n'endommageront donc pas vos autorisations existantes.

En outre, AWS prend en charge les politiques gérées pour les fonctions professionnelles qui couvrent plusieurs services. Par exemple, la politique ReadOnlyAccess AWS gérée fournit un accès en lecture seule à toutes Services AWS les ressources. Quand un service lance une nouvelle fonctionnalité, AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir la liste des politiques de fonctions professionnelles et leurs descriptions, consultez la page politiques gérées par AWS pour les fonctions de tâche dans le Guide de l'utilisateur IAM.

AWS Proton fournit des politiques IAM gérées et des relations de confiance que vous pouvez associer à des utilisateurs, à des groupes ou à des rôles qui permettent différents niveaux de contrôle sur les ressources et les opérations d'API. Vous pouvez appliquer ces politiques directement ou les utiliser comme points de départ pour créer vos propres politiques.

La relation de confiance suivante est utilisée pour chacune des politiques AWS Proton gérées.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ExampleTrustRelationshipWithProtonConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS politique gérée : AWSProton FullAccess

Vous pouvez les associer `AWSProtonFullAccess` à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations administratives qui permettent un accès complet aux AWS Proton actions et un accès limité aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux administrateurs un accès complet à AWS Proton APIs.
- `iam`— Permet aux administrateurs de transmettre des rôles à AWS Proton. Cela est nécessaire pour AWS Proton pouvoir effectuer des appels d'API vers d'autres services au nom de l'administrateur.
- `kms`— Permet aux administrateurs d'ajouter une autorisation à une clé gérée par le client.

- `codeconnections`— Permet aux administrateurs de répertorier et de transmettre des connexions par code afin qu'elles puissent être utilisées par AWS Proton.

Pour de plus amples informations, veuillez consulter AWSProtonFullAccess.

AWS politique gérée : AWSProton DeveloperAccess

Vous pouvez les associer `AWSProtonDeveloperAccess` à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres AWS actions qui en AWS Proton dépendent. L'étendue de ces autorisations est conçue pour soutenir le rôle d'un développeur qui crée et déploie AWS Proton des services.

Cette politique ne donne pas accès à la création, à la suppression et à la mise à jour de AWS Proton modèles et d'environnements APIs. Si les développeurs ont besoin d'autorisations encore plus limitées que celles prévues par cette politique, nous recommandons de créer une politique personnalisée dont l'étendue est limitée de manière à accorder le moindre privilège.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux contributeurs d'accéder à un ensemble limité de AWS Proton APIs.
- `codeconnections`— Permet aux contributeurs de répertorier et de transmettre des connexions de code afin qu'elles puissent être utilisées par AWS Proton.

Pour de plus amples informations, veuillez consulter AWSProtonDeveloperAccess.

AWS politique gérée : AWSProton ReadOnlyAccess

Vous pouvez les associer `AWSProtonReadOnlyAccess` à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations qui autorisent un accès en lecture seule aux AWS Proton actions et un accès en lecture seule limité aux autres actions de AWS service qui en dépendent. AWS Proton

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux contributeurs d'accéder en lecture seule à. AWS Proton APIs

Pour de plus amples informations, veuillez consulter AWSProtonReadOnlyAccess.

AWS politique gérée : AWSProton SyncServiceRolePolicy

AWS Proton associe cette politique au rôle AWSServiceRoleForProtonSynclié au service qui permet d'effectuer la AWS Proton synchronisation des modèles.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet de AWS Proton synchroniser un accès limité à AWS Proton APIs.
- `codeconnections`— Permet de AWS Proton synchroniser un accès limité à CodeConnections APIs.

Pour de plus amples informations, veuillez consulter AWSProtonSyncServiceRolePolicy.

AWS politique gérée : AWSProton CodeBuildProvisioningBasicAccess

Les autorisations CodeBuild doivent exécuter une version pour le AWS Proton CodeBuild provisionnement. Vous pouvez le rattacher `AWSProtonCodeBuildProvisioningBasicAccess` à votre rôle CodeBuild d'approvisionnement.

Cette politique accorde les autorisations minimales nécessaires au fonctionnement AWS Proton CodeBuild du provisionnement. Il accorde des autorisations qui permettent CodeBuild de générer des journaux de construction. Il autorise également Proton à mettre les sorties Infrastructure as Code (IaC) à la disposition des AWS Proton utilisateurs. Il ne fournit pas les autorisations nécessaires aux outils IaC pour gérer l'infrastructure.

La politique inclut les espaces de noms d'actions clés suivants :

- `logs`- Permet CodeBuild de générer des journaux de construction. Sans cette autorisation, il ne CodeBuild pourra pas démarrer.
- `proton`- Permet à une commande de CodeBuild provisionnement d'appeler `aws proton notify-resource-deployment-status-change` pour mettre à jour les sorties iAAC pour une ressource donnée AWS Proton .

Pour de plus amples informations, veuillez consulter AWSProtonCodeBuildProvisioningBasicAccess.

AWS politique gérée : AWSProton CodeBuildProvisioningServiceRolePolicy

AWS Proton associe cette politique au rôle AWSServiceRoleForProtonCodeBuildProvisioninglié au service qui permet d' AWS Proton effectuer un provisionnement CodeBuild basé.

Cette politique accorde des autorisations qui permettent un accès limité aux actions de AWS service qui AWS Proton dépendent de.

La politique inclut les espaces de noms d'actions clés suivants :

- `cloudformation`— Permet un accès limité au provisionnement AWS Proton CodeBuild basé sur CloudFormation APIs.
- `codebuild`— Permet un accès limité au provisionnement AWS Proton CodeBuild basé sur CodeBuild APIs.
- `iam`— Permet aux administrateurs de transmettre des rôles à AWS Proton. Cela est nécessaire pour AWS Proton pouvoir effectuer des appels d'API vers d'autres services au nom de l'administrateur.
- `servicequotas`— Permet de AWS Proton vérifier la limite de compilation CodeBuild simultanée, ce qui garantit une mise en file d'attente correcte des builds.

Pour de plus amples informations, veuillez consulter AWSProtonCodeBuildProvisioningServiceRolePolicy.

AWS politique gérée : AWSProton ServiceGitSyncServiceRolePolicy

AWS Proton associe cette politique au rôle AWSServiceRoleForProtonServiceSynclié au service qui permet d'effectuer la AWS Proton synchronisation des services.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet de AWS Proton synchroniser un accès limité à AWS Proton APIs.

Pour de plus amples informations, veuillez consulter AWSProtonServiceGitSyncServiceRolePolicy.

AWS Proton mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées AWS Proton 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 Historique du AWS Proton document.

Modifier	Description	Date
AWSProtonCodeBuildProvisioningServiceRolePolicy – Mise à jour d'une stratégie existante	La politique gérée pour le rôle lié à un service qui permet d' AWS Proton effectuer un provisionnement CodeBuild basé accorde désormais les autorisations nécessaires pour appeler les actions CloudFormation `TagResource` et `UntagResource` les actions d'API. Ces autorisations sont nécessaires pour effectuer des opérations de balisage sur les ressources.	15 juin 2024
AWSProtonFullAccess : mise à jour d'une politique existante	La politique gérée permettant au rôle lié à un service d'utiliser la synchronisation Git avec les référentiels Git a été mise à jour pour les ressources portant les deux préfixes de service. Pour plus d'informations, consultez les sections Utilisation de rôles liés à un service pour AWS CodeConnections et politiques gérées.	25 avril 2024
AWSProtonDeveloperAccess : mise à jour d'une politique existante	La politique gérée permettant au rôle lié à un service d'utiliser la synchronisation Git avec	25 avril 2024

Modifier	Description	Date
	les référentiels Git a été mise à jour pour les ressources portant les deux préfixes de service. Pour plus d'informations, consultez les sections Utilisation de rôles liés à un service pour AWS CodeConnections et politiques gérées.	
AWSProtonSyncServiceRolePolicy : mise à jour d'une politique existante	La politique gérée permettant au rôle lié à un service d'utiliser la synchronisation Git avec les référentiels Git a été mise à jour pour les ressources portant les deux préfixes de service. Pour plus d'informations, consultez les sections Utilisation de rôles liés à un service pour AWS CodeConnections et politiques gérées.	25 avril 2024
AWSProtonCodeBuildProvisioningServiceRolePolicy : mise à jour d'une politique existante	AWS Proton a mis à jour cette politique pour ajouter des autorisations afin de garantir que les comptes disposent de la limite de création CodeBuild simultanée nécessaire pour utiliser CodeBuild Provisioning.	12 mai 2023

Modifier	Description	Date
AWSProtonServiceGitSyncServiceRolePolicy : nouvelle politique	AWS Proton a ajouté une nouvelle politique permettant d' AWS Proton effectuer la synchronisation des services. La politique est utilisée dans le rôle AWSServiceRoleForProtonServiceSync lié au service.	31 mars 2023
AWSProtonDeveloperAccess : mise à jour d'une politique existante	AWS Proton a ajouté une nouvelle `GetResourcesSummary` action qui vous permet d'afficher un résumé de vos modèles, des ressources de modèles déployées et des ressources obsolètes.	18 novembre 2022
AWSProtonReadOnlyAccess : mise à jour d'une politique existante	AWS Proton a ajouté une nouvelle `GetResourcesSummary` action qui vous permet d'afficher un résumé de vos modèles, des ressources de modèles déployées et des ressources obsolètes.	18 novembre 2022
AWSProtonCodeBuildProvisioningBasicAccess : nouvelle politique	AWS Proton a ajouté une nouvelle politique qui lui donne CodeBuild les autorisations nécessaires pour exécuter une version pour le AWS Proton CodeBuild provisionnement.	16 novembre 2022

Modifier	Description	Date
AWSProtonSyncServiceRolePolicy : nouvelle politique	AWS Proton a ajouté une nouvelle politique permettant d' AWS Proton effectuer des opérations liées au provisionnement CodeBuild basé. La politique est utilisée dans le rôle AWSServiceRoleForProtonCodeBuildProvisioninglié au service.	2 septembre 2022
AWSProtonFullAccess : mise à jour d'une politique existante	AWS Proton a mis à jour cette politique afin de permettre l'accès aux nouvelles opérations d' AWS Proton API et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022
AWSProtonDeveloperAccess : mise à jour d'une politique existante	AWS Proton mettre à jour cette politique afin de fournir l'accès aux nouvelles opérations d' AWS Proton API et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022

Modifier	Description	Date
AWSProtonReadOnlyAccess : mise à jour d'une politique existante	AWS Proton mettre à jour cette politique afin de fournir l'accès aux nouvelles opérations d' AWS Proton API et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022
AWSProtonSyncServiceRolePolicy : nouvelle politique	AWS Proton a ajouté une nouvelle politique permettant d' AWS Proton effectuer des opérations liées à la synchronisation des modèles. La politique est utilisée dans le rôle AWSServiceRoleForProtonSync lié au service.	23 novembre 2021
AWSProtonFullAccess : nouvelle politique	AWS Proton a ajouté une nouvelle politique pour fournir un accès aux rôles administratifs aux opérations de l' AWS Proton API et à la AWS Proton console.	9 juin 2021
AWSProtonDeveloperAccess : nouvelle politique	AWS Proton a ajouté une nouvelle politique pour fournir un accès aux rôles de développeur aux opérations de l' AWS Proton API et à la AWS Proton console.	9 juin 2021

Modifier	Description	Date
AWSProtonReadOnlyAccess : nouvelle politique	AWS Proton a ajouté une nouvelle politique pour fournir un accès en lecture seule aux opérations de AWS Proton l'API et à la AWS Proton console.	9 juin 2021
AWS Proton a commencé à suivre les modifications.	AWS Proton a commencé à suivre les modifications apportées AWS à ses politiques gérées.	9 juin 2021

Utilisation de rôles liés à un service pour AWS Proton

AWS Proton utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à. AWS Proton Les rôles liés au service sont prédéfinis par AWS Proton et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Rubriques

- Utilisation des rôles pour la AWS Proton synchronisation
- Utilisation des rôles pour le provisionnement CodeBuild basé

Utilisation des rôles pour la AWS Proton synchronisation

AWS Proton utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à. AWS Proton Les rôles liés au service sont prédéfinis par AWS Proton et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration AWS Proton car vous n'avez pas à ajouter manuellement les autorisations nécessaires. AWS Proton définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul AWS Proton peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos AWS Proton ressources car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez la section AWS Services qui fonctionnent avec IAM et recherchez les services dont la valeur est Oui dans la colonne Rôles liés à un service. Sélectionnez un Oui ayant un lien pour consulter la documentation du rôle lié à un service, pour ce service.

Autorisations de rôle liées à un service pour AWS Proton

AWS Proton utilise deux rôles liés à un service nommés AWSServiceRoleForProtonSyncet. AWSServiceRoleForProtonServiceSync

Le rôle lié à un service AWSServiceRoleForProtonSync approuve les services suivants pour endosser le rôle :

- `sync.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSProtonSyncServiceRolePolicy` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire des AWS Proton modèles et des versions de modèles
- Action : utiliser la connexion activée CodeConnections

Pour plus d'informations sur cette politique, consultez AWS politique gérée : AWSProton SyncServiceRolePolicy.

Le rôle lié à un service AWSServiceRoleForProtonServiceSync approuve les services suivants pour endosser le rôle :

- `service-sync.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSProtonServiceGitSyncServiceRolePolicy` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire des AWS Proton services et des instances de service

Pour plus d'informations sur cette politique, consultez AWS politique gérée : AWSProton ServiceGitSyncServiceRolePolicy.

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour AWS Proton

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous configurez un référentiel ou un service à synchroniser AWS Proton dans le AWS Management Console, le AWS CLI ou l' AWS API, vous AWS Proton créez le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous configurez un référentiel ou un service pour la synchronisation AWS Proton, AWS Proton crée à nouveau le rôle lié au service pour vous.

Pour recréer le rôle AWSServiceRoleForProtonSynclié à un service, vous devez configurer un référentiel pour la synchronisation, et pour le recréer AWSServiceRoleForProtonServiceSync, vous devez configurer un service pour la synchronisation.

Modification d'un rôle lié à un service pour AWS Proton

AWS Proton ne vous permet pas de modifier le rôle AWSServiceRoleForProtonSynclié au service. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence au rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Supprimer un rôle lié à un service pour AWS Proton

Vous n'avez pas besoin de supprimer manuellement le rôle AWSServiceRoleForProtonSync. Lorsque vous supprimez tous les référentiels AWS Proton liés pour les synchroniser dans l'API AWS Management Console, le AWS CLI ou l' AWS API, vous AWS Proton nettoyez les ressources et supprimez le rôle lié au service pour vous.

Régions prises en charge pour les rôles AWS Proton liés à un service

AWS Proton prend en charge l'utilisation de rôles liés au service partout Régions AWS où le service est disponible. Pour plus d'informations, consultez Points de terminaison et quotas AWS Proton dans le document Références générales AWS.

Utilisation des rôles pour le provisionnement CodeBuild basé

AWS Proton utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à. AWS Proton Les rôles liés au service sont prédéfinis par AWS Proton et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration AWS Proton car vous n'avez pas à ajouter manuellement les autorisations nécessaires. AWS Proton définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul AWS Proton peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos AWS Proton ressources car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez la section AWS Services qui fonctionnent avec IAM et recherchez les services dont la valeur est Oui dans la colonne Rôles liés à un service. Sélectionnez un Oui ayant un lien pour consulter la documentation du rôle lié à un service, pour ce service.

Autorisations de rôle liées à un service pour AWS Proton

AWS Proton utilise le rôle lié à un service nommé AWSServiceRoleForProtonCodeBuildProvisioning— Un rôle lié à un service pour AWS Proton CodeBuild le provisionnement.

Le rôle lié à un service AWSServiceRoleForProtonCodeBuildProvisioning approuve les services suivants pour endosser le rôle :

- `codebuild.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSProtonCodeBuildProvisioningServiceRolePolicy` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire sur les CloudFormation piles et les transformations
- Action : créer, gérer et lire des CodeBuild projets et des builds

Pour plus d'informations sur cette politique, consultez AWS politique gérée : AWSProton CodeBuildProvisioningServiceRolePolicy.

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour AWS Proton

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un environnement qui utilise le provisionnement CodeBuild basé AWS Proton dans l'API AWS Management Console, le ou l' AWS API AWS CLI, vous AWS Proton créez le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un environnement qui utilise le provisionnement CodeBuild basé sur le provisionnement dans AWS Proton, il AWS Proton crée à nouveau le rôle lié au service pour vous.

Modification d'un rôle lié à un service pour AWS Proton

AWS Proton ne vous permet pas de modifier le rôle AWSServiceRoleForProtonCodeBuildProvisioninglié au service. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Supprimer un rôle lié à un service pour AWS Proton

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Toutefois, vous devez supprimer tous les environnements et services (instances et pipelines) qui utilisent le provisionnement CodeBuild basé AWS Proton avant de pouvoir le supprimer manuellement.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au AWSServiceRoleForProtonCodeBuildProvisioningservice. Pour plus d'informations, consultez la section Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles AWS Proton liés à un service

AWS Proton prend en charge l'utilisation de rôles liés au service partout Régions AWS où le service est disponible. Pour plus d'informations, consultez Points de terminaison et quotas AWS Proton dans le document Références générales AWS.

Résolution des problèmes AWS Proton d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS Proton IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans AWS Proton
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS Proton ressources

Je ne suis pas autorisé à effectuer une action dans AWS Proton

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

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 `proton:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `proton:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS Proton.

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 exécuter une action dans AWS Proton. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction 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 permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS Proton ressources

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 (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises AWS Proton en charge, consultezComment AWS Proton 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.

Analyse de configuration et de vulnérabilité dans AWS Proton

AWS Proton ne fournit pas de correctifs ou de mises à jour pour le code fourni par le client. Les clients sont responsables de la mise à jour et de l'application de correctifs à leur propre code, y compris le code source de leurs services et applications qui s'exécutent, AWS Proton ainsi que le code fourni dans leurs ensembles de modèles de services et d'environnement.

Les clients sont responsables de la mise à jour et de l'application des correctifs aux ressources de l'infrastructure dans leurs environnements et services. AWS Proton ne mettra pas automatiquement à jour ou ne patchera aucune ressource. Les clients sont invités à consulter la documentation relative aux ressources de leur architecture afin de comprendre leurs politiques respectives en matière de correctifs.

Outre la fourniture de mises à jour d'environnement et de service demandées par le client pour les versions mineures des modèles de service et d'environnement, AWS Proton il ne fournit pas de correctifs ou de mises à jour des ressources que les clients définissent dans leurs modèles et ensembles de modèles de services et d'environnements.

Pour plus de détails, consultez les ressources suivantes :

- Modèle de responsabilité partagée
- Amazon Web Services : Présentation des procédures de sécurité

Protection des données dans AWS Proton

AWS Proton est conforme au modèle de responsabilité AWS partagée modèle de de qui inclut des réglementations et des directives pour la protection des données. AWS est chargé de protéger l'infrastructure mondiale qui gère tous les Services AWS. AWS conserve le contrôle des données hébergées sur cette infrastructure, y compris les contrôles de configuration de sécurité pour le traitement du contenu client et des données personnelles. AWS les clients et les partenaires APN,

agissant en tant que responsables du traitement des données ou en tant que sous-traitants, sont responsables de toutes les données personnelles qu'ils déposent dans le AWS Cloud

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des comptes utilisateur individuels avec Gestion des identités et des accès AWS (IAM), afin que chaque utilisateur ne dispose que des autorisations nécessaires pour accomplir 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 recommandons TLS 1.2 ou version ultérieure.
- 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.

Nous vous recommandons vivement de ne jamais saisir d'informations d'identification sensibles, telles que les numéros de compte de vos clients, dans des champs de texte libres tels que le champ Nom. Cela inclut lorsque vous travaillez avec AWS Proton ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous saisissez dans des champs de texte en format libre pour les identificateurs de ressources ou des éléments similaires liés à la gestion des AWS ressources 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.

Pour en savoir plus sur la protection des données, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD sur le Blog sur la sécurité d'AWS .

Chiffrement au repos côté serveur

Si vous choisissez de chiffrer les données sensibles de vos ensembles de modèles au repos dans le compartiment S3 dans lequel vous stockez vos ensembles de modèles, vous devez utiliser une clé SSE-S3 ou SSE-KMS pour permettre de récupérer les ensembles de modèles afin qu'ils puissent être attachés AWS Proton à un modèle enregistré. AWS Proton

Chiffrement en transit

Toutes les communications de service à service sont chiffrées en transit à l'aide de SSL/TLS.

AWS Proton gestion des clés de chiffrement

À l'intérieur AWS Proton, toutes les données des clients sont cryptées par défaut à l'aide d'une clé AWS Proton détenue. Si vous fournissez une AWS KMS clé détenue et gérée par le client, toutes les données du client sont cryptées à l'aide de la clé fournie par le client, comme décrit dans les paragraphes suivants.

Lorsque vous créez un AWS Proton modèle, vous spécifiez votre clé et AWS Proton utilisez vos informations d'identification pour créer une autorisation qui permet AWS Proton d'utiliser votre clé.

Si vous retirez manuellement l'autorisation ou si vous désactivez ou supprimez la clé que vous avez spécifiée, vous ne pourrez pas lire les données chiffrées par la clé spécifiée et renvoyées`ValidationException`. AWS Proton

AWS Proton contexte de chiffrement

AWS Proton prend en charge les en-têtes de contexte de chiffrement. Un contexte de chiffrement est un ensemble facultatif de paires clé-valeur qui peut contenir des informations contextuelles supplémentaires sur les données. Pour des informations générales sur le contexte de chiffrement, consultez la section Concepts AWS Key Management Service - Contexte de chiffrement du Guide du développeur AWS Key Management Service .

Un contexte de chiffrement est un ensemble de paires clé-valeur contenant des données arbitraires non secrètes. Lorsque vous incluez un contexte de chiffrement dans une demande de chiffrement de données, lie AWS KMS cryptographiquement le contexte de chiffrement aux données chiffrées. Pour déchiffrer les données, vous devez transmettre le même contexte de chiffrement.

Les clients peuvent utiliser le contexte de chiffrement pour identifier l'utilisation de leur clé gérée par le client dans les enregistrements et les journaux d'audit. Il apparaît également en texte clair dans les journaux, tels que AWS CloudTrail Amazon CloudWatch Logs.

AWS Proton ne prend en compte aucun contexte de chiffrement spécifié par le client ou externe.

AWS Proton ajoute le contexte de chiffrement suivant.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

Le premier contexte de chiffrement identifie le AWS Proton modèle auquel la ressource est associée et sert également de contrainte pour les autorisations et les autorisations clés gérées par le client.

Le second contexte de chiffrement identifie la AWS Proton ressource cryptée.

Les exemples suivants montrent l'utilisation du contexte de AWS Proton chiffrement.

Développeur créant une instance de service.

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service/my-service/
service-instance/my-service-instance"
}
```

Un administrateur crée un modèle.

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service-template/my-
template"
}
```

Sécurité de l'infrastructure dans AWS Proton

En tant que service géré, AWS Proton il 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 AWS Proton 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.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Pour améliorer l'isolation du réseau, vous pouvez utiliser AWS PrivateLink les méthodes décrites dans la section suivante.

AWS Proton et points de terminaison VPC d'interface ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre VPC et créer un point de AWS Proton terminaison VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLinkune technologie qui vous permet d'accéder en privé AWS Proton APIs sans passerelle Internet, appareil NAT, connexion VPN ou AWS Direct Connect connexion. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec elles. AWS Proton APIs Le trafic entre votre VPC et celui qui AWS Proton ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux.

Pour de plus amples informations, consultez Points de terminaison VPC (AWS PrivateLink) dans le Guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de AWS Proton terminaison VPC

Avant de configurer un point de terminaison VPC d'interface pour AWS Proton, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC.

AWS Proton permet d'appeler toutes ses actions d'API depuis votre VPC.

Les politiques de point de terminaison VPC sont prises en charge pour. AWS Proton Par défaut, l'accès complet à AWS Proton est autorisé via le point de terminaison. Pour plus d'informations, consultez Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

Création d'un point de terminaison VPC d'interface pour AWS Proton

Vous pouvez créer un point de terminaison VPC pour le AWS Proton service à l'aide de la console Amazon VPC ou du (). AWS Command Line Interface AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Créez un point de terminaison VPC à l' AWS Proton aide du nom de service suivant :

- com.amazonaws. *region*.proton

Si vous activez le DNS privé pour le point de terminaison, vous pouvez envoyer des demandes d'API AWS Proton en utilisant son nom DNS par défaut pour la région, par exemple,`proton.region.amazonaws.com`.

Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison VPC pour AWS Proton

Vous pouvez attacher une stratégie de point de terminaison à votre point de terminaison d'un VPC qui contrôle l'accès à AWS Proton. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

Exemple : politique de point de terminaison VPC pour les actions AWS Proton

Voici un exemple de politique de point de terminaison pour AWS Proton. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès aux AWS Proton actions répertoriées pour tous les principaux sur toutes les ressources.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServices",
        "proton:ListServiceInstances",
        "proton:ListEnvironments",
        "proton:GetServiceTemplate",
```

```
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetEnvironment",
        "proton:CreateService",
        "proton:UpdateService",
        "proton:UpdateServiceInstance",
        "proton:UpdateServicePipeline",
        "proton:DeleteService"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Connexion et surveillance AWS Proton

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS Proton et des performances de vos autres AWS solutions. AWS fournit les outils de surveillance suivants pour surveiller l'exécution de vos instances AWS Proton, signaler tout problème et prendre des mesures automatiques le cas échéant.

À l'heure actuelle, AWS Proton il n'est pas intégré à Amazon CloudWatch Logs ou AWS Trusted Advisor. Les administrateurs peuvent configurer et utiliser CloudWatch pour surveiller les autres Services AWS , comme indiqué dans leurs modèles de service et d'environnement. AWS Proton est intégré à AWS CloudTrail.

- 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.
- 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 Amazon CloudWatch Logs.

- AWS CloudTrailcapture les appels d'API et les événements associés effectués par vous ou en votre nom Compte AWS et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, 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.
- Amazon EventBridge est un service de bus d'événements sans serveur qui permet de connecter facilement vos applications à des données provenant de diverses sources. EventBridge fournit un flux de données en temps réel à partir de vos propres applications Software-as-a-Service (SaaS) Services AWS et achemine ces données vers des cibles telles que Lambda. Cela vous permet de surveiller les événements qui se produisent dans les services et de créer des architectures basées sur les événements. Pour plus d'informations, veuillez consulter Automatisez AWS Proton avec EventBridge et le Guide de l'utilisateur EventBridge .

Résilience dans AWS Proton

L'infrastructure AWS mondiale est construite autour Région AWS de zones de disponibilité. Région AWS s 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 Région AWS zones de disponibilité et les zones de disponibilité, consultez la section Infrastructure AWS globale.

Outre l'infrastructure AWS mondiale, AWS Proton propose des fonctionnalités qui vous aident à répondre à vos besoins en matière de résilience et de sauvegarde des données.

AWS Proton sauvegardes

AWS Proton conserve une sauvegarde de toutes les données des clients. En cas de panne totale, cette sauvegarde peut être utilisée pour restaurer AWS Proton les données du client à partir d'un état valide antérieur.

Bonnes pratiques en matière de sécurité pour AWS Proton

AWS Proton fournit des 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.

Rubriques

Utilisation du contrôle d'accès IAM

IAM est un outil Service AWS que vous pouvez utiliser pour gérer les utilisateurs et leurs autorisations dans AWS. Vous pouvez utiliser IAM AWS Proton pour spécifier les AWS Proton actions que les administrateurs et les développeurs peuvent effectuer, telles que la gestion de modèles, d'environnements ou de services. Vous pouvez utiliser les rôles de service IAM pour permettre AWS Proton de passer des appels vers d'autres services en votre nom.

Pour plus d'informations sur les rôles IAM AWS Proton et les rôles IAM, consultezIdentity and Access Management pour AWS Proton.

Mettez en œuvre l'accès avec le moindre privilège. Pour plus d'informations, consultez la section Politiques et autorisations dans IAM dans le guide de l'Gestion des identités et des accès AWS utilisateur.

N'intégrez pas d'informations d'identification dans vos modèles et ensembles de modèles

Plutôt que d'intégrer des informations sensibles dans vos CloudFormation modèles et ensembles de modèles, nous vous recommandons d'utiliser des références dynamiques dans votre modèle de pile.

Les références dynamiques constituent un moyen compact et puissant de référencer des valeurs externes stockées et gérées dans d'autres services, tels que le AWS Systems Manager Parameter Store ou AWS Secrets Manager. Lorsque vous utilisez une référence dynamique, CloudFormation récupère la valeur de la référence spécifiée lorsque cela est nécessaire lors des opérations de pile et de modification des ensembles de modifications, et transmet la valeur à la ressource appropriée. Cependant, CloudFormation ne stocke jamais la valeur de référence réelle. Pour plus d'informations, consultez la section Utilisation de références dynamiques pour spécifier des valeurs de modèle dans le Guide de CloudFormation l'utilisateur.

AWS Secrets Manager vous aide à chiffrer, stocker et récupérer en toute sécurité des informations d'identification pour vos bases de données et d'autres services. Le AWS Systems Manager Parameter Store fournit un stockage hiérarchique sécurisé pour la gestion des données de configuration.

Pour plus d'informations sur la définition des paramètres du modèle, consultez https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html le guide de CloudFormation l'utilisateur.

Utilisez le chiffrement pour protéger les données sensibles

À l'intérieur AWS Proton, toutes les données des clients sont cryptées par défaut à l'aide d'une clé AWS Proton détenue.

En tant que membre de l'équipe de la plateforme, vous pouvez fournir une clé gérée par le client AWS Proton pour chiffrer et sécuriser vos données sensibles. Chiffrez les données sensibles au repos dans votre compartiment S3. Pour de plus amples informations, veuillez consulter Protection des données dans AWS Proton.

AWS CloudTrail À utiliser pour afficher et enregistrer les appels d'API

AWS CloudTrail suit toute personne effectuant des appels d'API dans votre Compte AWS. Les appels d'API sont enregistrés chaque fois que quelqu'un utilise l' AWS Proton API, la AWS Proton console ou AWS Proton AWS CLI les commandes. Activez la journalisation et spécifiez un compartiment Amazon S3 pour y stocker les journaux. Ainsi, si nécessaire, vous pouvez vérifier qui a effectué quel AWS Proton appel sur votre compte. Pour de plus amples informations, veuillez consulter Connexion et surveillance AWS Proton.

Prévention du problème de l'adjoint confus entre services

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 la confusion des adjoints. L'usurpation d'identité entre services 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 auprès des principaux fournisseurs de services qui ont obtenu l'accès aux ressources de votre compte.

Nous recommandons d'utiliser les clés de contexte de condition `aws:SourceAccount`globale `aws:SourceArn`et les clés contextuelles dans les politiques de ressources afin de limiter les autorisations qui AWS Proton accordent un autre service à la ressource. 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. Si vous utilisez les deux clés de contexte de condition globale et que la valeur `aws:SourceArn` contient l'ID de compte, la valeur `aws:SourceAccount` et le compte dans la valeur `aws:SourceArn` doivent utiliser le même ID de compte lorsqu'ils sont utilisés dans la même instruction de politique. 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.

La valeur de `aws:SourceArn` doit être une ressource qui AWS Proton stocke.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'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::proton:*:`*`123456789012`*`:environment/*`.

L'exemple suivant montre comment vous pouvez utiliser les touches de contexte de condition `aws:SourceAccount` globale `aws:SourceArn` et globale AWS Proton pour éviter le problème de confusion des adjoints.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ExampleProtonConfusedDeputyPreventionPolicy",
        "Effect": "Allow",
        "Principal": {"Service": "proton.amazonaws.com"},
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
            }
        }
    }
}
```

CodeBuild fourniture d'un support Amazon VPC personnalisé

AWS Proton CodeBuild Le provisionnement exécute des commandes CLI arbitraires fournies par le client dans CodeBuild un projet situé dans AWS Proton le compte Environment. Ces commandes gèrent généralement les ressources à l'aide d'un outil d'infrastructure en tant que code (IaC), tel que CDK. Si vous disposez de ressources dans un Amazon VPC, il est CodeBuild possible que vous ne puissiez pas y accéder. Pour ce faire, CodeBuild prend en charge la possibilité de s'exécuter au sein d'un Amazon VPC spécifique. Voici quelques exemples de cas d'utilisation :

- Récupérez les dépendances à partir de référentiels d'artefacts internes auto-hébergés, tels que pour `PyPI` Python, `Maven` pour Java et pour Node.js `npm`
- CodeBuild doit accéder à un serveur Jenkins dans un Amazon VPC particulier pour enregistrer un pipeline.
- Accédez aux objets d'un compartiment Amazon S3 configuré pour autoriser l'accès uniquement via un point de terminaison Amazon VPC.
- Exécutez des tests d'intégration depuis votre build par rapport aux données d'une base de données Amazon RDS isolée sur un sous-réseau privé.

Pour plus d'informations, consultez la documentation sur CodeBuild les VPC.

Si vous souhaitez que le CodeBuild provisioning s'exécute dans un VPC personnalisé AWS Proton , cette solution est simple. Tout d'abord, vous devez ajouter l'ID du VPC, les sous-réseaux et les groupes de sécurité au modèle d'environnement. Ensuite, vous entrez ces valeurs dans les spécifications d'environnement. Cela se traduira par la création d'un CodeBuild projet qui cible un VPC donné.

Mise à jour du modèle d'environnement

Schema

L'ID VPC, les sous-réseaux et les groupes de sécurité doivent être ajoutés au schéma du modèle afin qu'ils puissent exister dans les spécifications de l'environnement.

Un exemple `schema.yaml` :

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentInputType"
  types:
    EnvironmentInputType:
      type: object
      properties:
        codebuild_vpc_id:
          type: string
        codebuild_subnets:
          type: array
          items:
            type: string
        codebuild_security_groups:
          type: array
          items:
            type: string
```

Cela ajoute trois nouvelles propriétés qui seront utilisées par le manifeste :

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifeste

Pour configurer les paramètres Amazon VPC dans CodeBuild, une propriété facultative appelée `project_properties` est disponible dans le manifeste du modèle. Le contenu `project_properties` de est ajouté à la CloudFormation pile qui crée le CodeBuild projet. Cela permet d'ajouter non seulement les propriétés Amazon VPC, mais également toutes les CloudFormationCodeBuild CloudFormation propriétés prises en charge, telles que le délai de génération. Les mêmes données fournies sont mises à `proton-inputs.json` disposition pour les valeurs de`project_properties`.

Ajoutez cette section à votre `manifest.yaml` :

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Voici à quoi `manifest.yaml` peut ressembler le résultat :

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy -- --force
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Création de l'environnement

Lorsque vous créez un environnement avec votre modèle compatible CodeBuild Provisioning VPC, vous devez fournir l'identifiant Amazon VPC, les sous-réseaux et les groupes de sécurité.

Pour obtenir la liste de tous les Amazon VPC de votre IDs région, exécutez la commande suivante :

```
aws ec2 describe-vpcs
```

Pour obtenir une liste de tous les sous-réseaux IDs, exécutez :

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> N'incluez que les sous-réseaux privés. CodeBuild échouera si vous fournissez des sous-réseaux publics. Les sous-réseaux publics ont une route par défaut vers une passerelle Internet, contrairement aux sous-réseaux privés.

Exécutez la commande suivante pour obtenir le groupe de sécurité IDs. Ils IDs peuvent également être obtenus par le biais de AWS Management Console :

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Les valeurs ressembleront à :

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

Garantir CodeBuild les autorisations

Le support Amazon VPC nécessite certaines autorisations, telles que la possibilité de créer une interface réseau élastique.

Si l'environnement est créé dans la console, entrez ces valeurs dans l'assistant de création d'environnement. Si vous souhaitez créer l'environnement par programmation, voici ce `spec.yaml` qui suit :

```
proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton ressources et balisage

AWS Proton les ressources auxquelles est attribué un Amazon Resource Name (ARN) incluent les modèles d'environnement et leurs versions principales et secondaires, les modèles de service et leurs versions principale et mineure, les environnements, les services, les instances de service, les composants et les référentiels. Vous pouvez étiqueter ces ressources pour vous aider à les organiser et à les identifier. Vous pouvez utiliser des balises pour classer les ressources en fonction de leur objectif, de leur propriétaire, de leur environnement ou d'autres critères. Pour de plus amples informations, consultez Politiques de balisage . Pour suivre et gérer vos AWS Proton ressources, vous pouvez utiliser les fonctionnalités de balisage décrites dans les sections suivantes.

AWS balisage

Vous pouvez attribuer des métadonnées à vos AWS ressources sous forme de balises. Chaque balise est composée d'une clé définie par le client et d'une valeur facultative. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources.

> **Important**
>
> N'ajoutez pas de données d'identification personnelle (PII) ou d'autres informations confidentielles ou sensibles dans les étiquettes. Les étiquettes accessibles à de nombreux Services AWS, y compris la facturation. Les étiquettes ne sont pas destinées à être utilisées pour des données privées ou sensibles.

Chaque balise se compose de deux parties.

- Une clé de balise (par exemple, `CostCenter`, `Environment` ou `Project`). Les touches de tag distinguent les majuscules et minuscules.
- Une valeur de balise (facultatif) (par exemple, `111122223333` ou`Production`). Les valeurs de balise sont sensibles à la casse, tout comme les clés de balise.

Les exigences de base en matière de dénomination et d'utilisation suivantes s'appliquent aux balises.

- Chaque ressource peut avoir un maximum de 50 balises créées par l'utilisateur.

> **Note**
>
> Les balises créées par le système qui commencent par le `aws:` préfixe sont réservées à AWS l'usage et ne sont pas prises en compte dans cette limite. Vous ne pouvez pas modifier ou supprimer une balise commençant par le préfixe `aws:`.

- Pour chaque ressource, chaque clé d'identification doit être unique, et chaque clé d'identification peut avoir une seule valeur.
- La clé de balise doit comporter au minimum 1 et au maximum 128 caractères Unicode en UTF-8.
- La valeur de la balise doit être au minimum de 1 et au maximum de 256 caractères Unicode en UTF-8.
- Les caractères autorisés dans les balises sont les lettres, les chiffres, les espaces représentables en UTF-8 et les caractères suivants :* _. :/= + - @.

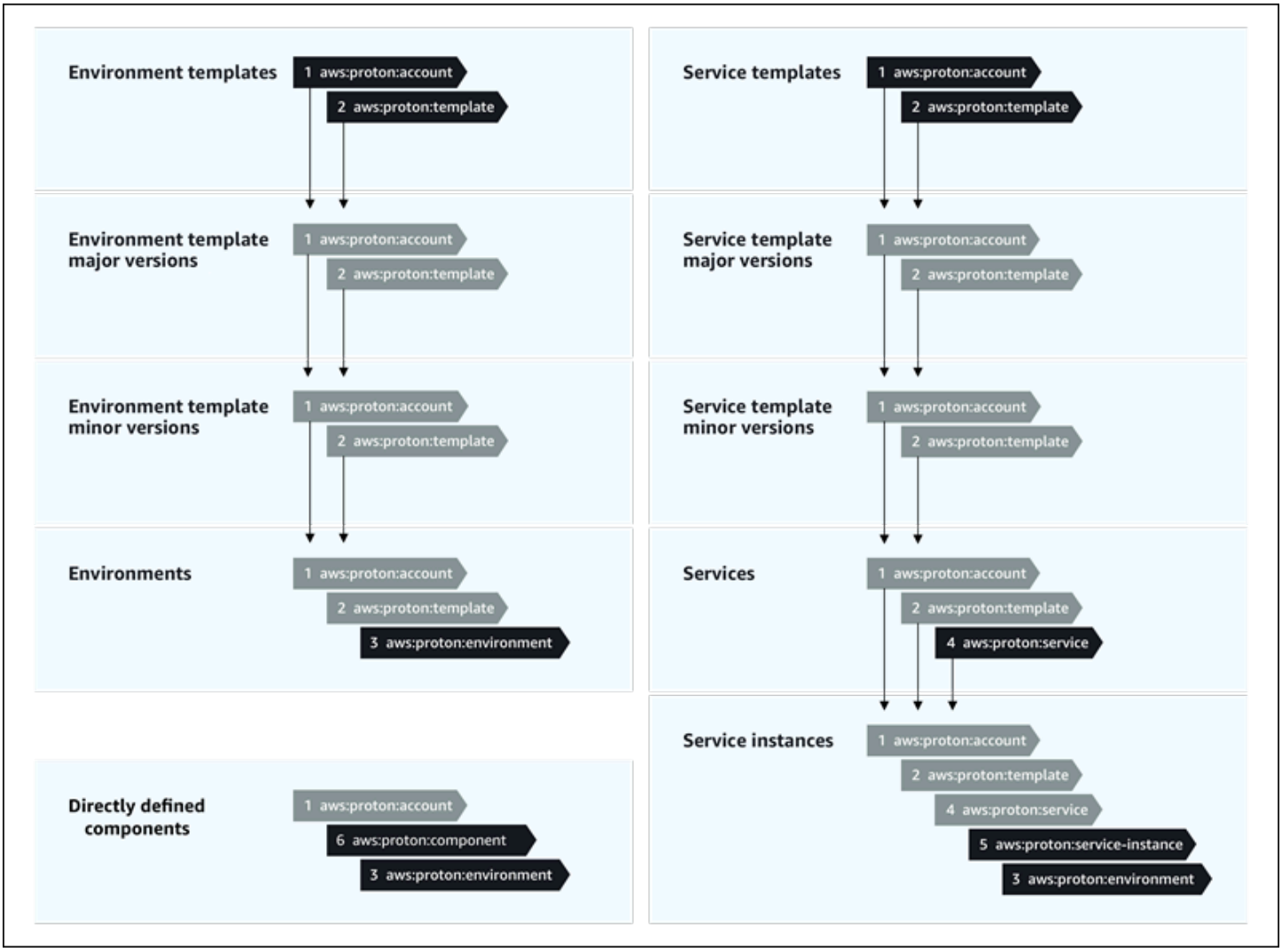
AWS Proton balisage

Avec AWS Proton, vous pouvez utiliser à la fois les balises que vous créez et les balises AWS Proton générées automatiquement pour vous.

AWS Proton AWS balises gérées

Lorsque vous créez une AWS Proton ressource, des balises AWS gérées sont AWS Proton automatiquement générées pour votre nouvelle ressource, comme indiqué dans le schéma suivant. AWS les balises gérées se propagent ensuite à d'autres AWS Proton ressources basées sur votre nouvelle ressource. Par exemple, les balises gérées d'un modèle d'environnement se propagent vers ses versions, et les balises gérées d'un service se propagent vers ses instances de service.

> **Note**
>
> AWS les balises gérées ne sont pas générées pour les connexions aux comptes d'environnement. Pour de plus amples informations, veuillez consulter the section called "Connexions aux comptes".



Propagation des balises vers les ressources provisionnées

Si les ressources provisionnées, telles que celles définies dans les modèles de service et d'environnement, prennent en charge le AWS balisage, les balises gérées se propagent sous forme de balises AWS gérées par le client vers les ressources provisionnées. Ces balises ne se propageront pas à une ressource provisionnée qui ne prend pas en charge AWS le balisage.

AWS Proton applique des balises à vos ressources en fonction AWS Proton des comptes, des modèles enregistrés et des environnements déployés, ainsi que des services et des instances de services, comme décrit dans le tableau suivant. Vous pouvez utiliser des balises AWS gérées pour afficher et gérer vos AWS Proton ressources, mais vous ne pouvez pas les modifier.

AWS clé de balise gérée	Clé gérée par le client propagée	Description
`aws:proton:account`	`proton:account`	Le AWS compte qui crée et déploie les AWS Proton ressources.
`aws:proton:template`	`proton:template`	L'ARN d'un modèle sélection né.
`aws:proton:environment`	`proton:environment`	L'ARN d'un environnement sélectionné.
`aws:proton:service`	`proton:service`	L'ARN d'un service sélection né.
`aws:proton:service-instance`	`proton:service-instance`	L'ARN d'une instance de service sélectionnée.
`aws:proton:component`	`proton:component`	L'ARN d'un composant sélectionné.

Voici un exemple de balise AWS gérée pour une AWS Proton ressource.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

Voici un exemple de balise gérée par le client appliquée à une ressource provisionnée qui a été propagée à partir d'une balise AWS gérée.

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

Avec AWS-managed provisioning, AWS Proton applique des balises propagées directement aux ressources provisionnées.

Avec le provisionnement autogéré, AWS Proton met à disposition les balises propagées ainsi que les fichiers iAc rendus qu'il soumet dans la pull request (PR) de provisionnement. Les balises sont fournies dans la variable string map`proton_tags`. Nous vous recommandons de faire

référence à cette variable dans votre configuration Terraform pour y inclure des AWS Proton balises. `default_tags` Cela propage les AWS Proton balises à toutes les ressources provisionnées.

L'exemple suivant montre cette méthode de propagation de balises dans un modèle Terraform d'environnement.

Voici la définition de la `proton_tags` variable :

proton.environment.variables.tf :

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

Voici comment les valeurs de balise sont attribuées à cette variable :

proton.auto.tfvars.json :

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

Et voici comment vous pouvez ajouter des AWS Proton balises à votre configuration Terraform afin qu'elles soient ajoutées aux ressources provisionnées :

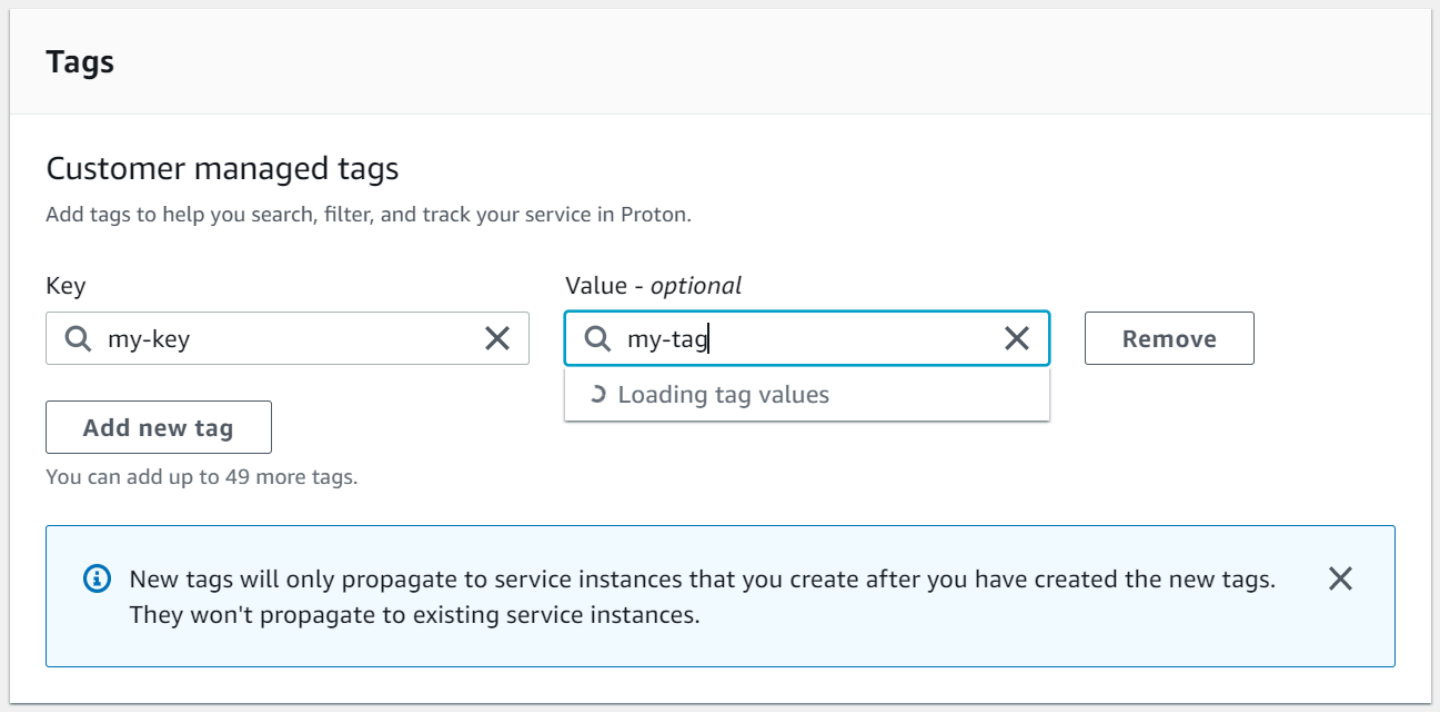
```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Tags gérés par le client

Chaque AWS Proton ressource dispose d'un quota maximum de 50 balises gérées par le client. Les balises gérées par le client se propagent aux AWS Proton ressources enfants de la même manière que les balises AWS gérées, sauf qu'elles ne se propagent pas aux AWS Proton ressources existantes ou aux ressources provisionnées. Si vous appliquez une nouvelle balise à une AWS Proton ressource contenant des ressources enfants existantes et que vous souhaitez que les ressources enfants existantes soient étiquetées avec la nouvelle balise, vous devez étiqueter chaque ressource enfant existante manuellement, à l'aide de la console ou AWS CLI.

Création de balises à l'aide de la console et de la CLI

Lorsque vous créez une AWS Proton ressource à l'aide de la console, vous avez la possibilité de créer des balises gérées par le client sur la première ou la deuxième page de la procédure de création, comme indiqué dans l'instantané de console suivant. Choisissez Ajouter une nouvelle étiquette, entrez la clé et la valeur et continuez.



Après avoir créé une nouvelle ressource à l'aide de la AWS Proton console, vous pouvez consulter la liste des balises AWS gérées et gérées par le client sur la page détaillée.

Création ou modification d'un tag

1. Dans la AWS Proton console, ouvrez une page détaillée AWS Proton des ressources où vous pouvez voir une liste de balises.
2. Choisissez Gérer les balises.
3. Sur la page Gérer les balises, vous pouvez afficher, créer, supprimer et modifier des balises. Vous ne pouvez pas modifier les balises AWS gérées répertoriées en haut de la page. Cependant, vous pouvez ajouter et modifier les balises gérées par le client à l'aide de champs d'édition, listés après les balises AWS gérées.

   Choisissez Ajouter un nouveau tag pour créer un nouveau tag.
4. Entrez une clé et une valeur pour le nouveau tag.
5. Pour modifier un tag, entrez une valeur dans le champ de valeur du tag pour une clé sélectionnée.
6. Pour supprimer un tag, choisissez Supprimer pour le tag sélectionné.
7. Lorsque vous avez terminé vos modifications, choisissez Enregistrer les modifications.

Créez des balises à l'aide du AWS Proton AWS CLI

Vous pouvez afficher, créer, supprimer et modifier des balises à l'aide du AWS Proton AWS CLI.

Vous pouvez créer ou modifier une balise pour une ressource, comme indiqué dans l'exemple suivant.

```
$ aws proton tag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tags '[{"key":"mykey1","value":"myval1"},{"key":"mykey2","value":"myval2"}]'
```

Vous pouvez supprimer une balise pour une ressource, comme indiqué dans l'exemple suivant.

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

Vous pouvez répertorier les balises d'une ressource comme indiqué dans le dernier exemple.

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

Résolution des problèmes AWS Proton

Apprenez à résoudre les problèmes liés à AWS Proton.

Rubriques

Erreurs de déploiement faisant référence à des paramètres CloudFormation dynamiques

Si des erreurs de déploiement font référence à vos variables CloudFormation dynamiques, vérifiez qu'elles ont échappé à Jinja. Ces erreurs peuvent être causées par une mauvaise interprétation de vos variables dynamiques par Jinja. La syntaxe des paramètres CloudFormation dynamiques est très similaire à la syntaxe Jinja que vous utilisez avec vos AWS Proton paramètres.

Exemple de syntaxe de variable CloudFormation dynamique :

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

Exemple de AWS Proton syntaxe Jinja pour les paramètres :

`'{{ service_instance.environment.outputs.env-outputs }}'`.

Pour éviter ces erreurs d'interprétation, Jinja échappe à vos paramètres CloudFormation dynamiques, comme indiqué dans les exemples suivants.

Cet exemple est tiré du guide de CloudFormation l'utilisateur. Les segments AWS Secrets Manager secret-name et json-key peuvent être utilisés pour récupérer les informations de connexion stockées dans le secret.

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
```

```
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
```

Pour échapper aux paramètres CloudFormation dynamiques, vous pouvez utiliser deux méthodes différentes :

- Placez un bloc entre `{% raw %} and {% endraw %}` :

```
'{% raw %}'
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
'{% endraw %}'
```

- Placez un paramètre entre `"{{ }}"` :

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}' }}"
      MasterUserPassword:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}' }}"
```

Pour plus d'informations, voir Jinja escaping.

AWS Proton quotas

Le tableau suivant répertorie les AWS Proton quotas. Toutes les valeurs sont par AWS compte, par AWS région prise en charge.

Quota de ressources	Limite par défaut	Ajustable?
Taille maximale du bundle de modèles	10 Mo	✕ Non
Taille maximale du modèle de fichier manifeste	2 Mo	✕ Non
Taille maximale du fichier de schéma modèle	2 Mo	✕ Non
Taille maximale de chaque fichier modèle	2 Mo	✕ Non
Longueur maximale de chaque nom de modèle	100 caractères	✕ Non
Nombre maximum de fichiers CloudFormation modèles par bundle	1	✕ Non
Nombre maximum de modèles enregistrés par compte, modèles de service et d'environnement combinés	1 000	✓ Oui
Nombre maximum de versions de modèle enregistrées par modèle	1 000	✓ Oui
Nombre maximal de fichiers par bundle de CodeBuild provisioning	500	✕ Non
Nombre maximum d'environnements par compte	1 000	✓ Oui
Nombre maximum de services par compte	1 000	✓ Oui
Nombre maximum d'instances de service par service	20	✓ Oui
Nombre maximum de composants par compte	1 000	✓ Oui

Quota de ressources	Limite par défaut	Ajustable?
Nombre maximal de connexions à un compte d'environnement par compte d'environnement	1 000	✓ Oui

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation en lien avec la dernière version de AWS Proton et les commentaires des clients. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Version de l'API : 2020-07-20

Modification	Description	Date
Avis de fin de support	Le 7 octobre 2026, AWS le support de AWS Proton.	7 octobre 2025
Mise à jour de politique gérée	`AWSProtonCodeBuildProvisioningServiceRolePolicy` la politique a été mise à jour.	15 juin 2024
Mise à jour de politique gérée	`AWSProtonDeveloperAccess` la politique a été mise à jour.	25 avril 2024
Mise à jour de politique gérée	`AWSProtonFullAccess` la politique a été mise à jour.	25 avril 2024
Mise à jour de politique gérée	`AWSProtonSyncServiceRolePolicy` la politique a été mise à jour.	25 avril 2024
Mise à jour de politique gérée	`AWSProtonCodeBuildProvisioningServiceRolePolicy` la politique a été mise à jour.	12 mai 2023
Configurations de synchronisation des services.	AWS Proton ajoute la prise en charge des configurations de synchronisation des services.	31 mars 2023

CodeBuild	AWS Proton ajoute le support pour le CodeBuildprovisionnement.	16 novembre 2022
Mise à jour de politique gérée	Ajout d'une `AWSProtonCodeBuildProvisioningBasicAccess` politique qui donne CodeBuild les autorisations nécessaires pour exécuter une version pour le AWS Proton CodeBuild provisionnement.	11 novembre 2022
Propagation des balises Terraform	Ajout de la propagation des balises Terraform au chapitre sur le balisage.	16 septembre 2022
Guide de migration d'API	Suppression du guide de migration de l'API pré-GA.	12 août 2022
AWS Proton objets	Ajout d'une rubrique sur AWS Proton les objets et leur relation avec d'autres AWS objets tiers. Voir AWS Proton les objets.	29 juillet 2022
Clarifications relatives aux référentiels liés	Clarification de l'objectif des référentiels liés (enregistrés) et de leur utilisation dans le guide.	18 juillet 2022
Fusion de guides	Fusion des deux guides de l'administrateur et de l'utilisateur distincts en un seul guide, le Guide de AWS Proton l'utilisateur.	30 juin 2022

Mise à jour de politique gérée	Politiques gérées mises à jour pour permettre l'accès aux nouvelles opérations d' AWS Proton API et pour résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console. Voir les politiques AWS gérées pour AWS Proton.	20 juin 2022
Commencer à utiliser la CLI	Mise à jour Commencer AWS CLI avec un nouveau didacticiel utilisant la nouvelle bibliothèque de modèles.	14 juin 2022
Composants directement définis	Ajout du chapitre Composants et modifications connexes apportées tout au long du guide.	1er juin 2022
AWS Proton bibliothèque de modèles	Ajout de la rubrique « Bibliothèque de AWS Proton modèles ».	6 mai 2022
Disponibilité générale de Terraform (GA)	Le provisionnement par pull request a été renommé en provisionnement autogéré. Ajout d'une rubrique sur les méthodes de provisionnement.	23 mars 2022
Balisage des référentiels	Ajout de la prise en charge du balisage des ressources du référentiel. Consultez la section Créer un lien vers votre dépôt.	23 mars 2022

Mise à jour de la documentation	Ajout du balisage de connexion au compte d'environnement.	26 novembre 2021
Synchronisation des modèles et aperçu de Terraform	Ajout de la gestion automatique des versions des modèles avec la fonction de synchronisation des modèles pour une disponibilité générale et le provisionnement des pull requests avec Terraform en version préliminaire. Guide de migration des API de retour.	24 novembre 2021
mises à jour de documentation	Ajout d'un EventBridgedidacticiel, d'un flux de travail de démarrage, d'un mode de AWS Proton fonctionnement et d'améliorations de la section relative aux ensembles de modèles.	17 septembre 2021
AWS Proton lancement des panneaux d'aide de la console	Des panneaux d'aide ont été ajoutés à la console. La suppression de la version du modèle de console ne supprime plus les versions inférieures. Le guide de migration de l'API a été supprimé.	8 septembre 2021
AWS Proton version de disponibilité générale (GA)	Ajout d'environnements multicomptes, de EventBridge surveillance, declés de condition IAM, de prise en charge de l'idempotencie et de quotas accrus.	9 juin 2021

Ajoutez et supprimez des instances de service pour un service et utilisez l'infrastructure externe existante pour les environnements avec AWS Proton	Cette version préliminaire publique inclut des mises à jour qui vous permettent d'ajouter et de supprimer des instances de service dans un service, d'utiliser votre infrastructure externe existante dans un AWS Proton environnement et d'annuler des déploiements d'environnement, d'instance de service et de pipeline. AWS Proton prend désormais en charge PrivateLink. Une validation de suppression supplémentaire a été ajoutée pour éviter qu'une version mineure ne soit supprimée par erreur alors qu'une ressource l'utilise.	27 avril 2021
Marquage avec AWS Proton	La version préliminaire publique 2 inclut le AWS Proton balisage et la possibilité de lancer des services sans pipeline de services.	5 mars 2021
Première version	La version préliminaire publique est désormais disponible dans certaines régions.	1er décembre 2020

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.