aws

Guida per gli sviluppatori

Deadline Cloud

Deadline Cloud: Guida per gli sviluppatori

Table of Contents

Cos'è Deadline Cloud? 1
Open Job Description 2
Concetti e terminologia 2
Risorse agricole 2
Risorse per l'esecuzione del lavoro 3
Altri concetti e terminologia importanti 5
Guida all'architettura 8
Fonte del lavoro 10
Workflow interattivo 10
Flusso di lavoro automatico 10
Invio del lavoro 10
Inviatore integrato con DCC 11
Definizione del lavoro personalizzata 11
Gestione delle applicazioni 12
Deadline: canale conda gestito dal cloud per flotte gestite dai servizi (SMF) 12
Canale conda autogestito 12
Gestione personalizzata delle applicazioni 12
Licenze dell'applicazione 13
Flotte gestite dai servizi e licenze basate sull'utilizzo 13
Flotte gestite dal cliente e licenze basate sull'utilizzo 13
Licenze personalizzate 14
Accesso alle risorse 14
Allegati Job 14
Accesso allo storage personalizzato 15
Monitoraggio del lavoro e gestione dell'output 15
Monitoraggio Deadline Cloud 15
Applicazione di monitoraggio personalizzata 16
Soluzione di monitoraggio automatizzata 16
Gestione dell'infrastruttura dei lavoratori 16
Flotte gestite dai servizi 16
Flotte gestite dai clienti 17
Architetture di esempio 17
Studio di produzione tradizionale 17
Studio nel cloud 20

Automazione di ECommerce 21
Whitelabel/OEM/B2C Cliente 24
Cos'è un carico di lavoro Deadline Cloud 27
In che modo i carichi di lavoro derivano dalla produzione 27
Gli ingredienti di un carico di lavoro 28
Portabilità del carico di lavoro 29
Nozioni di base 32
Crea una fattoria 32
Passaggi successivi 36
Esegui l'agente lavoratore 36
Passaggi successivi 39
Invia offerte di lavoro 39
Invia il simple_job campione 40
Invia con un parametro 43
Crea un job simple_file_job 44
Fasi successive 47
Invia offerte di lavoro con allegati 47
Configura la coda per gli allegati dei lavori 48
Invia con allegati di lavoro 51
Come vengono archiviati gli allegati di lavoro 53
Fasi successive 56
Aggiungi una flotta gestita dai servizi 57
Fasi successive 59
Pulisci le risorse agricole 59
Crea un lavoro 63
Pacchetti Job 64
Elementi del modello Job 67
Suddivisione delle attività 70
Elementi dei valori dei parametri 73
Elementi di riferimento delle risorse 75
Utilizzo dei file nei lavori 78
Esempio di infrastruttura di progetto 79
Profili di archiviazione e mappatura dei percorsi 81
Allegati Job 89
Invio di file con un lavoro 90
Ottenere file di output da un lavoro 101

Utilizzo dei file in un passaggio dipendente 105
Creare limiti di risorse per i lavori 107
Interruzione ed eliminazione dei limiti 109
Crea un limite 110
Associa un limite e una coda 110
Invia un lavoro che richiede dei limiti 111
Invio di un processo 112
Da un terminale 113
Da una sceneggiatura 114
Dall'interno delle candidature 115
Pianifica i lavori 117
Configurazioni di pianificazione 117
Determina la compatibilità della flotta 120
Ridimensionamento della flotta 122
Sessioni 122
Dipendenze tra fasi 125
Modifica i lavori 127
Flotte gestite dai clienti 133
Crea un CMF 133
Configurazione dell'host di lavoro 139
Configurare un ambiente Python 140
Installa l'agente di lavoro 140
Configura l'agente di lavoro 142
Crea utenti e gruppi di lavoro 143
Protezione dell'host di lavoro 146
Gestione dell’accesso 148
Concessione dell'accesso 149
Revocare l'accesso 150
Installa il software per i lavori 150
Installa gli adattatori DCC 151
Configura le credenziali 151
Flusso di dati host dei lavoratori 155
Endpoint e protocolli 155
Operazioni API utilizzate dai lavoratori 156
Altri dati trasmessi 157
Opzioni di connettività privata 158

Metti alla prova il tuo host di lavoro 158
Crea un AMI 161
Prepara l'istanza 161
Costruisci il AMI 163
Crea un'infrastruttura per la flotta 164
Ridimensiona automaticamente la tua flotta 169
Controllo dello stato della flotta 174
Flotte gestite dai servizi 175
Connetti le risorse VPC al tuo SMF 175
Come funzionano gli endpoint di risorse VPC 176
Prerequisiti 176
Configura un endpoint di risorse VPC 177
Accesso alle risorse VPC 177
Autenticazione e sicurezza 178
Considerazioni tecniche 178
Risoluzione dei problemi 178
Allegati Job 179
Scegliete una modalità di filesystem 179
Ottimizza le prestazioni di trasferimento 180
Scarica i risultati del lavoro 181
Implementa e configura software personalizzato sui lavoratori 182
Scegli un metodo di distribuzione 182
Configurazione dei lavori utilizzando ambienti di coda 183
Controlla l'ambiente di lavoro 184
Fornisci candidature per i tuoi lavori 200
Crea un canale conda usando S3 203
Compila e testa pacchetti localmente 204
Pubblica pacchetti su un canale conda Amazon S3 210
Configura le autorizzazioni per la coda di produzione per pacchetti conda personalizzati 216
Aggiungi un canale conda a un ambiente di coda 217
Crea un pacchetto conda per un'applicazione o un plug-in 218
Crea una ricetta di costruzione di conda per Blender 221
Crea una ricetta conda per Maya 223
Crea una ricetta conda per l'adattatore Maya 226
Crea una ricetta conda per il plugin MtoA 228
Automatizza la creazione di pacchetti con Deadline Cloud 230

Script di configurazione dell'host 234
Risoluzione dei problemi 237
Utilizzo di licenze software 241
Combinazione di BYOL e UBL 241
Come funzionano le licenze combinate 241
Esempio: utilizzo delle licenze BYOL Cinema 4D con fallback UBL 242
Considerazioni sulla concessione di licenze combinate 243
Connect le flotte SMF a un server di licenze 243
Fase 1: Configurare l'ambiente di coda 244
Fase 2: (Facoltativo) Configurazione dell'istanza del proxy di licenza 254
CloudFormation Fase 3: configurazione del modello 255
Connect le flotte CMF a un endpoint di licenza 265
Fase 1: Creare un gruppo di sicurezza 266
Passaggio 2: configura l'endpoint della licenza 266
Fase 3: Connettere un'applicazione di rendering a un endpoint 267
Fase 4: Eliminare un endpoint di licenza 271
Utilizzo di agenti AI 272
Monitoraggio 275
CloudTrail registri 276
Deadline Cloud eventi relativi ai dati in CloudTrail 278
Deadline Cloud eventi gestionali in CloudTrail 280
Deadline Cloud esempi di eventi 283
Monitoraggio con CloudWatch 284
CloudWatch metriche 285
Allarmi raccomandati 288
Gestione degli eventi utilizzando EventBridge 289
Eventi Deadline Cloud 290
Invio di eventi Deadline Cloud 290
Riferimento ai dettagli sugli eventi 291
Interrogazione di dati aggregati relativi alle statistiche di sessione 307
Avvio di una richiesta di aggregazione 307
Recupero dei risultati 308
Recupero dei metadati utente tramite UserID 309
Per mappare un ID utente 309
Come trovare il tuo ID Identity Store 310
Verifica della mappatura degli utenti 311

Risorse aggiuntive ... 311
Sicurezza ... 312
Protezione dei dati ... 313
Crittografia dei dati a riposo ... 314
Crittografia dei dati in transito ... 314
Gestione delle chiavi ... 315
Inter-network privacy del traffico ... 325
Disattivazione ... 325
Identity and Access Management ... 326
Destinatari ... 327
Autenticazione con identità ... 327
Gestione dell'accesso tramite policy ... 329
Come funziona Deadline Cloud con IAM ... 331
Identity-based esempi di politiche ... 336
AWS politiche gestite ... 346
Ruoli di servizio ... 350
Risoluzione dei problemi ... 364
Convalida della conformità ... 366
Resilienza ... 366
Sicurezza dell'infrastruttura ... 367
Analisi della configurazione e delle vulnerabilità ... 367
Cross-service confusa prevenzione sostitutiva ... 368
AWS PrivateLink ... 369
Considerazioni ... 370
Deadline Cloud endpoint ... 370
Creare endpoint ... 371
Ambienti di rete con restrizioni ... 372
AWS Endpoint API da inserire nella lista consentita ... 372
Domini Web da inserire nella lista consentita ... 372
Environment-specific endpoint da inserire nella lista consentita ... 373
Best practice di sicurezza ... 374
Protezione dei dati ... 374
autorizzazioni IAM ... 375
Esegui lavori come utenti e gruppi ... 375
Rete ... 375
Dati sul lavoro ... 376

Struttura dell'azienda 376
Code di allegati Job 377
Bucket software personalizzati 380
Operatori ospitanti 380
Script di configurazione dell'host 382
Workstation 382
Verificare il software scaricato 383
Cronologia dei documenti 390
..... cccxci

Cos'è AWS Deadline Cloud?

AWS Deadline Cloud è un AWS servizio completamente gestito che ti consente di avere una farm di elaborazione scalabile attiva e funzionante in pochi minuti. Fornisce una console di amministrazione per la gestione di utenti, aziende agricole, code per la pianificazione dei lavori e flotte di lavoratori che si occupano dell'elaborazione.

Questa guida per sviluppatori è destinata agli sviluppatori di pipeline, strumenti e applicazioni in un'ampia gamma di casi d'uso, tra cui:

- Gli sviluppatori di pipeline e i direttori tecnici possono integrare Deadline Cloud APIs e le funzionalità nelle loro pipeline di produzione personalizzate.
- I fornitori di software indipendenti possono integrare Deadline Cloud nelle loro applicazioni, consentendo agli artisti e agli utenti della creazione di contenuti digitali di inviare lavori di rendering di Deadline Cloud senza problemi dalle loro postazioni di lavoro.
- Gli sviluppatori di servizi Web e basati su cloud possono integrare il rendering di Deadline Cloud nelle loro piattaforme, consentendo ai clienti di fornire risorse per visualizzare virtualmente i prodotti.

Forniamo strumenti che ti consentono di lavorare direttamente in qualsiasi fase della tua pipeline:

- Un'interfaccia a riga di comando che puoi usare direttamente o tramite script.
- L' AWS SDK per 11 linguaggi di programmazione più diffusi.
- Un'interfaccia web basata su REST che puoi chiamare dalle tue applicazioni.

Puoi anche usarne altre Servizi AWS nelle tue applicazioni personalizzate. Ad esempio, puoi usare:

- AWS CloudFormationper automatizzare la creazione e la rimozione di fattorie, code e flotte.
- Amazon CloudWatch raccoglierà metriche per i posti di lavoro.
- Amazon Simple Storage Service per archiviare e gestire risorse digitali e risultati di lavoro.
- AWS IAM Identity Centerper gestire utenti e gruppi per le tue aziende agricole.

Open Job Description

Deadline Cloud utilizza la specifica Open Job Description (OpenJD) per specificare i dettagli di un lavoro. OpenJD è stato sviluppato per definire lavori portabili tra soluzioni. Lo si usa per definire un job che è un insieme di comandi eseguiti su host di lavoro.

Puoi creare un modello di lavoro OpenJD utilizzando un mittente fornito da Deadline Cloud oppure puoi utilizzare qualsiasi strumento che desideri per creare il modello. Dopo aver creato il modello, lo invii a Deadline Cloud. Se usi un mittente, si occuperà dell'invio del modello. Se hai creato il modello in un altro modo, richiami un'azione da riga di comando di Deadline Cloud oppure puoi utilizzare una di queste per inviare il AWS SDKs lavoro. In entrambi i casi, Deadline Cloud aggiunge il lavoro alla coda specificata e pianifica il lavoro.

Concetti e terminologia per Deadline Cloud

Per aiutarti a iniziare a usare AWS Deadline Cloud, questo argomento spiega alcuni dei suoi concetti e della terminologia chiave.

Risorse agricole

Questo diagramma mostra come le risorse agricole di Deadline Cloud interagiscono.

Farm

Una farm contiene tutte le altre risorse relative all'invio e all'esecuzione dei lavori. Le aziende agricole sono indipendenti l'una dall'altra, il che le rende utili per separare gli ambienti di produzione.

Queue

Una coda contiene i lavori per la programmazione sulle flotte associate. Gli utenti possono inviare lavori a una coda e gestirne la priorità e lo stato all'interno della coda. Una coda deve essere associata a una flotta con un'associazione queue-fleet affinché i relativi lavori vengano eseguiti e le code possono essere associate a più flotte.

Parco istanze

Una flotta contiene la capacità di elaborazione per l'esecuzione dei lavori. Le flotte possono essere gestite dal servizio o dal cliente. Le flotte gestite dai servizi funzionano in Deadline Cloud

e includono funzionalità integrate come scalabilità automatica, licenze e accesso al software. Le flotte gestite dai clienti vengono eseguite sulle tue risorse di elaborazione come EC2 istanze Amazon o server locali.

Budget

Un budget stabilisce le soglie di spesa per la tua attività lavorativa e ti consente di intraprendere azioni quando vengono raggiunte le soglie, come interrompere la pianificazione dei lavori.

Ambiente di coda

Un ambiente di coda definisce gli script che vengono eseguiti su ciascun lavoratore per configurare o eliminare l'ambiente del carico di lavoro. Sono utili per impostare le variabili di ambiente, installare software e configurare lo storage delle risorse.

Profilo di archiviazione

Un profilo di archiviazione è una configurazione per un gruppo di host e workstation che indica dove si trovano i dati nel file system. Deadline Cloud utilizza i profili di archiviazione per mappare i percorsi quando si eseguono lavori su host configurati in modo diverso, ad esempio un lavoro inviato da Windows e su cui è in esecuzione. Linux

Limite

Un limite consente di tenere traccia dell'utilizzo di risorse condivise, come le licenze fluttuanti, e di controllare il modo in cui vengono allocate tra i lavori. I limiti sono associati alle code con associazioni di limiti di coda.

Monitoraggio

Il monitor configura l'URL per l'applicazione web di monitoraggio Deadline Cloud, consentendo agli utenti finali di monitorare e gestire i lavori. È possibile accedervi in un browser o tramite l'applicazione desktop di monitoraggio Deadline Cloud.

Risorse per l'esecuzione del lavoro

Questo diagramma mostra come le risorse di lavoro di Deadline Cloud interagiscono.

Processo

Un lavoro è un insieme di lavori che un utente invia a Deadline Cloud per essere pianificato ed eseguito sui lavoratori disponibili. Un lavoro può renderizzare una scena 3D o eseguire una

simulazione. I lavori vengono creati a partire da modelli di lavoro riutilizzabili, che definiscono l'ambiente e i processi di runtime e i parametri specifici del processo. I job contengono fasi e attività che definiscono il lavoro da eseguire e possono essere configurati con priorità, numero massimo di lavoratori e impostazioni per i nuovi tentativi.

Priorità del lavoro

La priorità del lavoro è l'ordine approssimativo in cui Deadline Cloud elabora un lavoro in una coda. È possibile impostare la priorità del lavoro tra 1 e 100, i lavori con una priorità numerica più alta vengono generalmente elaborati per primi. I lavori con la stessa priorità vengono elaborati nell'ordine di ricezione.

Proprietà processo

Le proprietà del lavoro sono impostazioni che definisci quando invii un lavoro di rendering. Alcuni esempi includono l'intervallo di fotogrammi, il percorso di output, gli allegati dei lavori, la fotocamera renderizzabile e altro ancora. Le proprietà variano in base al DCC da cui viene inviato il rendering.

Fase

Un passo fa parte di un processo che fornisce un modello per l'esecuzione di molte attività identiche ad eccezione dei valori dei parametri dell'attività. I passaggi possono dipendere da altri passaggi, il che consente di creare flussi di lavoro complessi con percorsi di esecuzione sequenziali o paralleli. Nei lavori di rendering, un passaggio spesso definisce il comando per il rendering di un frame e utilizza il numero di frame come parametro dell'attività.

Processo

Un'attività è l'unità di lavoro più piccola in Deadline Cloud. Le attività fanno parte delle fasi e vengono eseguite dai lavoratori, che rappresentano le singole operazioni che devono essere eseguite come parte di un lavoro. Le attività possono essere configurate con parametri specifici e vengono assegnate ai lavoratori in base alle loro capacità e disponibilità. Nei lavori di rendering, un'attività spesso esegue il rendering di un singolo frame.

Worker

I lavoratori fanno parte di una flotta ed eseguono le attività relative ai lavori. I lavoratori possono essere configurati con funzionalità specifiche come acceleratori GPU, architettura della CPU e sistema operativo. Nelle flotte gestite dai servizi, i lavoratori vengono creati automaticamente man mano che la flotta si espande continuamente.

Istanza

Le flotte utilizzano istanze per le risorse della CPU. Un'istanza è un'istanza Amazon EC2 Performance. Deadline Cloud utilizza istanze On-Demand e Spot.

Istanza On-Demand

Le istanze On-Demand hanno un prezzo al secondo, non hanno alcun impegno a lungo termine e non verranno interrotte.

Istanza Spot

Le istanze Spot offrono una capacità non riservata che puoi utilizzare a un prezzo scontato, ma che possono essere interrotte da richieste On-Demand.

Attendi e salva

La funzione Wait and Save consente una pianificazione ritardata dei lavori per ridurre i costi e può essere interrotta da richieste On-Demand e Spot. Wait and Save è disponibile solo nelle flotte gestite dai servizi Deadline Cloud.

Wait and Save serve per gestire l'esecuzione dei carichi di lavoro di visual computing in Deadline Cloud. AWS Consulta i termini AWS del servizio per i dettagli.

Sessione

Una sessione rappresenta la sequenza di lavoro di un lavoratore su un lavoro. Durante una singola sessione, a un lavoratore possono essere assegnate più attività che esegue una dopo l'altra. Le sessioni spesso prevedono azioni di configurazione che configurano gli ambienti e caricano le risorse prima di eseguire le azioni delle attività.

Azione della sessione

Un'azione di sessione rappresenta operazioni specifiche eseguite durante una sessione, come la configurazione dell'ambiente, l'esecuzione di un'attività e la sincronizzazione delle risorse.

Altri concetti e terminologia importanti

Esploratore di utilizzo

Usage explorer è una funzionalità di Deadline Cloud monitor. Fornisce una stima approssimativa dei costi e dell'utilizzo.

Responsabile del budget

Il gestore del budget fa parte del monitor Deadline Cloud. Usa il gestore del budget per creare e gestire i budget. Puoi anche usarlo per limitare le attività in modo da rispettare il budget.

Libreria client Deadline Cloud

La libreria client open source include un'interfaccia a riga di comando e una libreria per la gestione di Deadline Cloud. La funzionalità include l'invio di pacchetti di lavoro basati sulla specifica Open Job Description a Deadline Cloud, il download degli output degli allegati di lavoro e il monitoraggio della fattoria utilizzando l'interfaccia a riga di comando (CLI).

Applicazione per la creazione di contenuti digitali (DCC)

Le applicazioni per la creazione di contenuti digitali (DCCs) sono prodotti di terze parti in cui è possibile creare contenuti digitali. Deadline Cloud dispone di integrazioni integrate con molte piattaforme DCCs come Autodesk Maya, Blender e Maxon Cinema 4D, che consentono di inviare offerte di lavoro dall'interno del DCC ed eseguirne il rendering su flotte gestite dai servizi con software e licenze preconfigurati.

Allegati Job

Gli allegati di lavoro sono una funzionalità di Deadline Cloud che consente di caricare e scaricare risorse come parte di un lavoro, ad esempio texture, modelli 3D e impianti di illuminazione. Gli allegati Job vengono archiviati in Amazon S3 ed evitano la necessità di uno storage di rete condiviso.

Modello del processo

Un modello di lavoro definisce l'ambiente di runtime e tutti i processi eseguiti come parte di un job di Deadline Cloud.

Inviatore di Deadline Cloud

Un mittente di Deadline Cloud è un plug-in per un DCC che consente agli utenti di inviare facilmente lavori dall'interno del DCC.

Endpoint della licenza

Un endpoint di licenza rende disponibili le licenze basate sull'utilizzo di Deadline Cloud per prodotti di terze parti all'interno del tuo VPC. Questo modello prevede il pagamento in base al consumo e ti viene addebitato in base al numero di ore e minuti che utilizzi. Gli endpoint delle licenze non sono collegati alle farm e possono essere utilizzati indipendentemente.

Tag

Un tag è un'etichetta che puoi assegnare a una AWS risorsa. Ogni tag è composto da una chiave e da un valore opzionale definiti dall'utente. Con i tag, puoi classificare AWS le tue risorse in diversi modi, ad esempio per scopo, proprietario o ambiente.

Licenze basate sull'utilizzo (UBL)

Le licenze basate sull'utilizzo (UBL) sono un modello di licenza su richiesta disponibile per determinati prodotti di terze parti. Questo modello è pagato in base al consumo e ti viene addebitato il numero di ore e minuti che utilizzi.

Guida all'architettura cloud di Deadline

Questo argomento fornisce linee guida e best practice per progettare e creare render farm affidabili, sicure, efficienti ed economiche per i carichi di lavoro utilizzando Deadline Cloud. L'utilizzo di queste linee guida può aiutare a creare carichi di lavoro stabili ed efficienti, consentendo di concentrarsi sull'innovazione, sulla riduzione dei costi e sul miglioramento dell'esperienza dei clienti.

Questo contenuto è destinato ai Chief Technology Officer (CTOs), agli architetti, agli sviluppatori e ai membri del team operativo.

Un flusso di lavoro di end-to-end rendering richiede soluzioni a più livelli del processo, come la generazione di lavori, l'accesso alle risorse e il monitoraggio dei lavori. Deadline Cloud offre diverse soluzioni per ogni livello del processo di rendering. Selezionando tra le opzioni di Deadline Cloud in ogni livello, puoi progettare un flusso di lavoro adatto al tuo caso d'uso.

Per ogni livello, dovrai decidere quale approccio è il migliore per il tuo caso d'uso. Queste non sono definizioni di scenario rigide e non sono l'unico modo per utilizzare Deadline Cloud. Si tratta invece di un insieme di concetti di alto livello per aiutarti a capire come Deadline Cloud potrebbe adattarsi alla tua azienda o al tuo flusso di lavoro. Puoi separare i carichi di lavoro di Deadline Cloud nei seguenti livelli: Job Source, Job Submission, Application Management, Application Licensing, Asset Access, Output Management e Worker Infrastructure Management.

In generale, è possibile utilizzare mix-and-match qualsiasi scenario in un livello con qualsiasi altro scenario in un altro livello, ad eccezione delle combinazioni specifiche specificate di seguito.


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

Fonte del lavoro

La fonte del lavoro è il punto di accesso in cui i nuovi lavori entreranno nel sistema che sarà reso da Deadline Cloud. Ad alto livello, ci sono due fonti principali di lavoro: l'interattività umana e i sistemi informatici automatizzati.

Workflow interattivo

In questo scenario, un artista o un altro ruolo creativo è il principale generatore di lavoro da elaborare nella farm Deadline Cloud. Di solito il risultato di questi lavori è un artefatto principale per il progetto o il team più ampio. Svolgono il loro lavoro utilizzando software come uno strumento di creazione di contenuti digitali (DCC) standard del settore. Inviano manualmente i lavori alla Deadline Cloud farm e successivamente visualizzano i risultati per esaminarli. La workstation stessa non è gestita da. AWS

Nella maggior parte dei casi, questi artisti utilizzano i mittenti integrati di Deadline Cloud e il monitor Deadline Cloud nei livelli di applicazione e monitoraggio del carico di lavoro.

Flusso di lavoro automatico

In questo scenario, un sistema programmatico di proprietà del cliente è il principale generatore di posti di lavoro nella cloud farm di Deadline. Potrebbe trattarsi della generazione di asset in una pipeline di vendita al dettaglio, ad esempio un video giradischi generato da un modello o da una scansione 3D. Potrebbe trattarsi della composizione automatica della grafica delle trasmissioni e delle schede dei giocatori per lo sport. Il tema di questo scenario è che un individuo non invia manualmente ogni lavoro a Deadline Cloud, ma il lavoro viene generato come parte di un sistema più ampio.

Con i lavori automatizzati, è meno comune utilizzare gli inviatori integrati di Deadline Cloud e il monitor Deadline Cloud. Spesso le definizioni dei lavori si basano sullo sviluppo di applicazioni personalizzate scritte dall'utente e i risultati dei lavori confluiscono automaticamente in un sistema di Digital Asset Management (DAM) o in un sistema Media Asset Management (MAM) per l'approvazione e la distribuzione.

Invio del lavoro

Le offerte di lavoro vengono inviate a Deadline Cloud utilizzando OpenJobDescriptionmodelli. OpenJobDescription è una specifica aperta flessibile per la definizione di processi di elaborazione

in batch portabili tra diverse implementazioni di sistemi di pianificazione. Il file di definizione del Job descrive i parametri del lavoro, le fasi del lavoro, come viene parametrizzato un passaggio in base agli input del lavoro, nonché lo script effettivo che verrà eseguito su un Worker per eseguire l'elaborazione. L'idea di Workload Submission è come vengono create queste definizioni di lavoro, chi le crea e come vengono inviate.

Inviatore integrato con DCC

Un mittente integrato di Deadline Cloud è un software che collega Deadline Cloud a un DCC o pacchetto software standard del settore. Il mittente integrato determina come trasformare i dati e la configurazione per un carico di lavoro di rendering, composito o altro in un modello di lavoro, cosa che può essere compresa da Deadline Cloud. Molti mittenti integrati vengono creati e gestiti dal team di Deadline Cloud o dal creatore del pacchetto software, ma se non ne esiste già uno per l'applicazione desiderata, puoi creare e gestire il tuo mittente. Il team di Deadline Cloud ne supporta solo DCCs un numero limitato.

I flussi di lavoro interattivi di solito coinvolgono mittenti integrati, ma non sempre. Per i flussi di lavoro automatizzati basati su modelli, un flusso di lavoro comune prevede che un artista configuri un modello di lavoro nel proprio DCC ed esegua un'esportazione una tantum del job bundle. Questo pacchetto di lavori definisce come eseguire quel particolare tipo di lavoro su Deadline Cloud in modo parametrizzato. Questo pacchetto di lavori può essere integrato nello scenario Automated Workflow per scopi di automazione.

Definizione del lavoro personalizzata

Per applicazioni e flussi di lavoro personalizzati, è possibile controllare completamente il modo in cui queste definizioni di lavoro vengono create e inviate a Deadline Cloud. Ad esempio, un sito di e-commerce potrebbe chiedere ai venditori di caricare modelli 3D dell'oggetto che vendono. Dopo questo caricamento, la piattaforma di e-commerce potrebbe generare dinamicamente una definizione di lavoro da inviare a Deadline Cloud per generare automaticamente un'animazione giradischi su uno sfondo comune utilizzando un'illuminazione comune in modo che corrisponda agli altri oggetti 3D disponibili sul sito. Durante lo sviluppo della piattaforma di e-commerce, uno sviluppatore di software creava una definizione di lavoro, la incorporava nella piattaforma di e-commerce con i parametri eventualmente forniti dai venditori e codificava la piattaforma per inviare questo lavoro durante il flusso di lavoro di caricamento dei prodotti della piattaforma.

Deadline Cloud fornisce una serie di definizioni di lavoro di esempio nell'archivio degli esempi su github.

Gestione delle applicazioni

Dopo che un lavoro è stato inviato a Deadline Cloud e assegnato a un lavoratore, lo script della definizione del lavoro viene eseguito sul lavoratore. Nella maggior parte dei casi, questo script richiamerà un'applicazione per eseguire l'elaborazione effettiva, ad esempio un renderer, un composito, una codifica, un filtro o qualsiasi altra di una serie di attività ad alta intensità di calcolo. La gestione delle applicazioni consiste nel garantire che la versione necessaria del software richiesto sia disponibile per i lavoratori.

Puoi gestire le applicazioni utilizzando qualsiasi sistema di gestione dei pacchetti che preferisci, ma Deadline Cloud fornisce una serie di strumenti per abilitare facilmente l'uso dei pacchetti conda. Conda è un gestore di pacchetti e un sistema di gestione dell'ambiente open source, multipiattaforma e indipendente dal linguaggio.

Deadline: canale conda gestito dal cloud per flotte gestite dai servizi (SMF)

Quando si utilizzano flotte gestite dal servizio, un canale conda gestito da Deadline Cloud viene automaticamente configurato e configurato per essere utilizzato dai dipendenti. Il servizio Deadline Cloud fornisce una serie di applicazioni e rendering DCC partner in questo canale conda. Per ulteriori informazioni, consulta Creare un ambiente di coda nella guida per l'utente di Deadline Cloud. Questi pacchetti vengono aggiornati automaticamente dal servizio Deadline Cloud e non richiedono alcuna manutenzione da parte dell'utente. Questo canale conda è disponibile solo quando si utilizzano flotte gestite dal servizio e non è disponibile quando si utilizzano flotte gestite dal cliente.

Canale conda autogestito

Se non sei in grado di utilizzare il canale conda gestito da Deadline Cloud, devi determinare come installare, applicare patch e gestire in altro modo le applicazioni sulla tua flotta Deadline Cloud. Un'opzione è creare un canale conda da configurare e gestire. Questo interagirà più strettamente con il canale conda gestito da Deadline Cloud. Ad esempio, puoi utilizzare un DCC dal canale conda gestito da Deadline Cloud ma portare il tuo pacchetto che contiene un plug-in DCC specifico. Per ulteriori informazioni su questo processo, consulta Creare un canale conda utilizzando S3.

Gestione personalizzata delle applicazioni

Per la gestione delle applicazioni, il requisito di Deadline Cloud è che l'applicazione sia disponibile nel PATH quando lo script del lavoro viene eseguito sul lavoratore.

Se crei e gestisci già pacchetti Rez, puoi utilizzare un ambiente di coda per installare le applicazioni dai repository Rez. Un esempio di ambiente di coda può essere trovato su AWS Deadline Cloud org. GitHub

Se gestisci già le applicazioni su flotte gestite dal cliente con lavoratori longevi o in immagini di sistema, non è necessario alcun ambiente di coda per la gestione delle applicazioni. Assicurati che la candidatura compaia nel percorso dell'utente del lavoro e invia il lavoro.

Licenze dell'applicazione

Molti carichi di lavoro eseguiti comunemente su Deadline Cloud richiedono la licenza software del fornitore del software. Queste applicazioni sono spesso concesse in licenza per postazione, per CPU o per host. È tua responsabilità assicurarti che l'utilizzo di software di terze parti su Deadline Cloud sia conforme al contratto di licenza di terze parti. Se utilizzi software open source, software personalizzato o altro software senza licenza, non è richiesta la configurazione di questo livello. Tieni presente che Deadline Cloud supporta solo le licenze di rendering e non supporta le licenze per workstation.

Flotte gestite dai servizi e licenze basate sull'utilizzo

Quando si utilizzano flotte gestite dai servizi Deadline Cloud, le licenze basate sull'utilizzo (UBL) vengono configurate automaticamente per il software supportato. I lavori eseguiti su flotte gestite dai servizi dispongono automaticamente di variabili di ambiente impostate per le applicazioni supportate per indirizzarle a utilizzare i server di licenza Deadline Cloud. Quando utilizzi Deadline Cloud UBL, ti viene addebitato solo il numero di ore di utilizzo dell'applicazione con licenza.

Flotte gestite dal cliente e licenze basate sull'utilizzo

Le licenze basate sull'utilizzo (UBL) di Deadline Cloud sono disponibili anche quando non si utilizzano flotte gestite dai servizi. In questo scenario, configurerai gli endpoint di licenza Deadline Cloud che forniscono indirizzi IP nelle sottoreti VPC selezionate che forniscono l'accesso ai server di licenza Deadline Cloud. Dopo aver configurato le variabili di ambiente specifiche del software appropriate sui lavoratori e aver configurato la connettività di rete dai lavoratori agli indirizzi IP degli endpoint di licenza, i lavoratori possono effettuare il check-out e il check-in delle licenze per il software supportato. Le tariffe orarie per le licenze sono le stesse di quando si utilizza UBL in flotte gestite dai servizi.

Licenze personalizzate

Potresti utilizzare un'applicazione che non è supportata da Deadline Cloud UBL o potresti avere licenze preesistenti ancora valide. In questo scenario, sei responsabile della configurazione del percorso di rete dai tuoi dipendenti (gestiti dal cliente o dal servizio) ai server di licenza. Per ulteriori informazioni sulle licenze personalizzate, consulta. Connect flotte gestite dai servizi a un server di licenze personalizzato

Accesso alle risorse

Dopo che un lavoro è stato inviato a un lavoratore e l'applicazione è stata configurata, il lavoratore deve essere configurato per accedere ai dati relativi agli asset richiesti per il lavoro. Potrebbero trattarsi di dati 3D, dati di texture, dati di animazione, fotogrammi video o qualsiasi altro tipo di dati utilizzati nel lavoro.

Inizia a pensare a dove sono attualmente archiviati i tuoi dati. Potrebbe trovarsi sul disco rigido della workstation, su uno strumento di collaborazione utente, sul controllo del codice sorgente, su un file system condiviso in locale o nel cloud, su Amazon S3 o in qualsiasi altra posizione.

Quindi, considera cosa è necessario affinché un lavoratore acceda a questi dati. Questi dati sono disponibili solo sulla rete aziendale? Quale identità o credenziali sono necessarie per accedere ai dati? La fonte di dati è scalabile per supportare il lavoro con il numero di lavoratori che prevedi di elaborare il lavoro?

Allegati Job

Il meccanismo di accesso alle risorse più semplice per iniziare è rappresentato dagli allegati di lavoro di Deadline Cloud. Quando un lavoro viene inviato utilizzando gli allegati del lavoro, i dati richiesti dal lavoro vengono caricati in un bucket Amazon S3 insieme a un file manifest che specifica i file richiesti dal lavoro. Con gli allegati di lavoro, non sono necessarie complicate configurazioni di rete o di storage condiviso. I file vengono caricati una sola volta, quindi i caricamenti successivi vengono completati più rapidamente. Dopo che un lavoratore ha terminato l'elaborazione di un lavoro, i dati di output vengono caricati su Amazon S3 in modo che possano essere scaricati dall'artista o da un altro cliente. Job Attachments è adatto a flotte di qualsiasi dimensione ed è semplice e veloce da installare e utilizzare.

Job attachments non è lo strumento migliore per tutte le situazioni. Se i dati sono già attivi AWS, gli allegati di lavoro aggiungono una copia aggiuntiva dei dati, compresi i tempi di trasferimento e

i costi di archiviazione associati. Gli allegati del lavoro richiedono che il lavoro possa specificare completamente i dati richiesti al momento dell'invio, in modo che i dati possano essere caricati.

Per utilizzare gli allegati dei lavori, la coda di Deadline Cloud deve avere un bucket di allegati di lavoro associato e il ruolo in coda deve essere utilizzato per fornire l'accesso a quel bucket. Per impostazione predefinita, gli inviatori integrati di Deadline Cloud supportano tutti gli allegati di lavoro. Se non utilizzi un mittente integrato di Deadline Cloud, gli allegati di lavoro possono essere utilizzati con il tuo software personalizzato integrando la libreria python di Deadline Cloud.

Accesso allo storage personalizzato

Se non utilizzi gli allegati relativi alle offerte di lavoro, hai la responsabilità di garantire che i lavoratori abbiano accesso ai dati necessari per le mansioni. Deadline Cloud fornisce una serie di strumenti per supportare questo obiettivo e per mantenere la portabilità dei lavori. Potresti voler utilizzare una soluzione di archiviazione personalizzata quando disponi già di uno spazio di archiviazione di rete condiviso per artisti e lavoratori, preferisci utilizzare un servizio esterno come LucidLink, o per altri motivi.

Utilizzate i profili di storage per modellare i file system sulla workstation e sugli host di lavoro. Ogni profilo di storage descrive il sistema operativo e il layout del file system di una delle configurazioni di sistema. Utilizzando i profili di archiviazione, quando un artista che utilizza una workstation Windows invia un lavoro che viene elaborato da un Linux lavoratore, Deadline Cloud assicura la mappatura del percorso in modo che il lavoratore possa accedere all'archiviazione dei dati che hai configurato.

Quando si utilizzano flotte gestite dai servizi Deadline Cloud, gli script di configurazione dell'host e gli endpoint di risorse VPC consentono ai lavoratori di montare e accedere direttamente allo storage condiviso o ad altri servizi disponibili nel tuo VPC.

Monitoraggio del lavoro e gestione dell'output

Dopo che i lavori inviati a Deadline Cloud sono stati completati con successo, una persona o un processo scaricherà l'output del lavoro da utilizzare nel flusso di lavoro aziendale al di fuori di Deadline Cloud. Dopo un fallimento del lavoro, i registri dei lavori e le informazioni di monitoraggio aiutano a diagnosticare i problemi.

Monitoraggio Deadline Cloud

L'applicazione di monitoraggio Deadline Cloud è disponibile sul Web e per desktop. Questa soluzione è ideale per gli studi che utilizzano flussi di lavoro interattivi per un'ampia gamma di DCCs utilizzi

degli allegati di lavoro per l'archiviazione. Il monitor ti supporta solo quando utilizzi IAM Identity Center. IAM Identity Center è un prodotto Workforce Identity, non una soluzione di consumer identity (B2C), quindi non è appropriata per molti scenari B2C.

Applicazione di monitoraggio personalizzata

Se desideri personalizzare l'esperienza di monitoraggio dei tuoi utenti, stai creando un prodotto B2C o creando un sistema altamente specializzato utilizzando Deadline Cloud, scegli di creare un'applicazione di monitoraggio personalizzata. Puoi utilizzare l'API AWS Deadline Cloud per creare questa applicazione personalizzata, combinando il contesto del tuo flusso di lavoro complessivo con i concetti di Deadline Cloud. Ad esempio, il tuo prodotto B2C potrebbe avere un proprio concetto di progetto configurato dagli utenti e l'applicazione può inserire i lavori di Deadline Cloud nella stessa interfaccia.

Soluzione di monitoraggio automatizzata

In alcuni scenari, non è necessaria alcuna applicazione di monitoraggio dedicata per Deadline Cloud. Questo scenario è comune nei flussi di lavoro automatizzati in cui Deadline Cloud viene utilizzato per il rendering automatico delle risorse in una pipeline, come la grafica di trasmissione per sport o notizie. In questo scenario, l'API Deadline Cloud e EventBridge gli eventi vengono utilizzati per l'integrazione con un sistema di gestione delle risorse multimediali esterno per le approvazioni e lo spostamento dei dati alla fase successiva del processo.

Gestione dell'infrastruttura dei lavoratori

Le flotte Deadline Cloud sono un gruppo di server (lavoratori) in grado di elaborare i lavori inviati a una coda di Deadline Cloud e sono l'infrastruttura principale di qualsiasi farm Deadline Cloud.

Flotte gestite dai servizi

In una flotta gestita dai servizi, Deadline Cloud si assume la responsabilità degli host dei lavoratori, del sistema operativo, del networking, delle patch, della scalabilità automatica e di altri fattori legati alla gestione di una render farm. Tu specifichi il numero minimo e massimo di lavoratori che desideri, insieme alle specifiche di sistema richieste per la tua candidatura e Deadline Cloud si occuperà del resto. Le flotte gestite dai servizi sono l'unica opzione di flotta che può utilizzare i canali conda gestiti da Deadline Cloud per gestire facilmente le applicazioni DCC del settore. Inoltre, Deadline Cloud UBL viene configurato automaticamente con flotte gestite dai servizi. Le flotte Wait and Save per carichi di lavoro a basso costo e tolleranti ai ritardi sono disponibili solo utilizzando flotte gestite dai servizi.

Flotte gestite dai clienti

Utilizzate flotte gestite dal cliente quando avete bisogno di un maggiore controllo sugli host dei lavoratori e sul loro ambiente. Le flotte gestite dai clienti sono le più adatte quando si utilizza Deadline Cloud in locale. Per ulteriori informazioni, consulta Crea e utilizza flotte gestite dai clienti di Deadline Cloud.

Architetture di esempio

Studio di produzione tradizionale

Lo studio di produzione tradizionale richiede un'infrastruttura di elaborazione, archiviazione e rete significativa in grado di estendersi su più sedi fisiche fino ai carichi di lavoro di fornitura dei servizi. Ogni singolo pacchetto software e fornitore ha requisiti hardware, software, di rete e di licenza unici che devono essere soddisfatti durante la risoluzione dei conflitti di versione, compatibilità e risorse.

È comune avere requisiti di infrastruttura separati per le postazioni di lavoro degli artisti, i nodi di rendering, lo storage di rete, i server di licenza, i sistemi di code di lavoro, gli strumenti di monitoraggio e la gestione delle risorse. Gli studi in genere devono mantenere più versioni di strumenti DCC, renderer, plug-in e strumenti personalizzati mentre gestiscono accordi di licenza complessi all'interno della propria render farm. L'infrastruttura dello studio diventa più complicata se si considerano gli ambienti di sviluppo, controllo della qualità e produzione.

Una tipica implementazione di Deadline Cloud che utilizza opzioni gestite dai servizi risolve o riduce molte di queste sfide attraverso:

- Invio interattivo dei lavori tramite flusso di lavoro tramite inviatori DCC integrati
- Gestione delle applicazioni tramite canali conda gestiti da Deadline Cloud
- Licenze basate sull'utilizzo configurate automaticamente per il software supportato
- Gestione delle risorse tramite allegati di lavoro
- Monitoraggio tramite l'applicazione di monitoraggio Deadline Cloud
- Gestione dell'infrastruttura tramite flotte gestite dai servizi

Con questo approccio, gli artisti possono inviare lavori direttamente dai loro strumenti DCC familiari a una render farm cloud scalabile senza gestire infrastrutture complesse. Il servizio gestisce

automaticamente la distribuzione del software, le licenze, il trasferimento dei dati e la scalabilità dell'infrastruttura. Gli artisti possono monitorare il proprio lavoro tramite un'interfaccia Web o un'applicazione desktop e gli output vengono automaticamente archiviati in Amazon S3 per un facile accesso.

Con questa configurazione, gli studi possono creare ambienti di sviluppo e produzione in pochi minuti, pagare solo per l'elaborazione e le licenze che utilizzano e concentrarsi sul lavoro creativo piuttosto che sulla gestione dell'infrastruttura. L'approccio gestito dai servizi offre il percorso più rapido per adottare il rendering su cloud, mantenendo al contempo flussi di lavoro familiari per gli artisti.


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

Studio nel cloud

I moderni studi di effetti visivi e animazione stanno spostando sempre più spesso l'intera pipeline verso il cloud, comprese le workstation per artisti. Questo approccio elimina la necessità di un'infrastruttura locale, consente la collaborazione globale e offre una scalabilità perfetta sia per il lavoro interattivo che per il rendering. Tuttavia, introduce anche nuove sfide nella gestione delle risorse cloud, nell'assicurare l'accesso ai dati a bassa latenza e nell'integrazione delle workstation basate sul cloud con le render farm.

Un tipico studio nativo del cloud richiede un approccio unificato alla gestione delle workstation cloud, allo storage condiviso, all'infrastruttura di rendering e alla distribuzione del software su tutti questi componenti. Gli approcci tradizionali hanno spesso portato a sistemi complessi e gestiti manualmente che faticavano a bilanciare prestazioni, costi e flessibilità.

Una distribuzione Deadline Cloud per uno studio nativo del cloud può essere implementata utilizzando:

- Invio interattivo del flusso di lavoro tramite inviatori DCC integrati su workstation cloud
- Gestione delle applicazioni tramite nodi di rendering conda gestiti da Deadline Cloud
- Licenze basate sull'utilizzo configurate automaticamente per il software supportato
- Accesso personalizzato allo storage tramite Windows File Server FSx per i dati di progetto condivisi
- Monitoraggio tramite l'applicazione di monitoraggio Deadline Cloud
- Gestione dell'infrastruttura mediante flotte gestite dai servizi

Questo approccio consente agli artisti di lavorare su postazioni di lavoro basate su cloud con accesso diretto allo storage condiviso ad alte prestazioni e di inviare facilmente i lavori alla Deadline Cloud farm. Lo studio può gestire l'implementazione del software su entrambe le workstation e i nodi di rendering utilizzando gli stessi canali conda, garantendo coerenza e riducendo il sovraccarico di manutenzione.

I vantaggi principali di questa configurazione includono:

- Collaborazione globale con artisti in grado di accedere alle postazioni di lavoro da qualsiasi luogo
- Ambienti software coerenti tra workstation e nodi di rendering
- Storage condiviso ad alte prestazioni accessibile sia alle workstation che ai nodi di rendering
- Scalabilità flessibile delle risorse di elaborazione interattive e in batch
- Gestione centralizzata di tutta l'infrastruttura di studio nel cloud

La configurazione dello storage in questo scenario prevede in genere:

- FSx per Windows File Server per i dati di progetto, accessibile sia dalle workstation cloud che dai lavoratori di Deadline Cloud
- Profili di archiviazione in Deadline Cloud per gestire la mappatura dei percorsi tra le workstation e i nodi di rendering
- Montaggio diretto delle FSx condivisioni sui lavoratori di Deadline Cloud utilizzando endpoint di risorse VPC e script di configurazione dell'host

Questo approccio cloud-native consente agli studi di eliminare l'infrastruttura locale, permettendo una rapida scalabilità per progetti di qualsiasi dimensione e mantenendo al contempo flussi di lavoro familiari agli artisti. Offre la flessibilità necessaria per utilizzare una combinazione di risorse gestite dal servizio e dal cliente, ottimizzando sia la facilità di gestione che i requisiti prestazionali specifici.

Sfruttando le workstation cloud insieme a Deadline Cloud, gli studi possono realizzare una pipeline di produzione completamente integrata e accessibile a livello globale che si adatta perfettamente da piccoli team a grandi produzioni.

Automazione di ECommerce

La moderna piattaforma di e-commerce richiede la generazione automatizzata di asset su larga scala per fornire una visualizzazione completa dei prodotti su milioni di articoli. Gli approcci tradizionali richiederebbero investimenti significativi in infrastrutture per elaborare grandi volumi di modelli 3D in supporti di prodotto standardizzati, spesso con il risultato di sistemi sottodimensionati che creano arretrati di elaborazione o di sistemi sovradimensionati con capacità inattiva.

Un tipico flusso di lavoro di e-commerce automatizzato deve gestire l'elaborazione del caricamento dei prodotti, la convalida dei modelli 3D, la gestione della render farm, l'elaborazione degli output e l'integrazione con i sistemi di informazione sui prodotti. La gestione di questi flussi di lavoro richiede tradizionalmente il coordinamento di più applicazioni di rendering, risorse di calcolo e pipeline di elaborazione dei dati, garantendo al contempo una qualità costante e mantenendo l'efficienza dei costi su larga scala.

Un'implementazione di Deadline Cloud per l'automazione dell'e-commerce può essere implementata utilizzando:

- Invio automatico dei lavori al flusso di lavoro tramite l'integrazione di API personalizzate nell'applicazione di inserimento di e-commerce esistente

- Definizioni di lavoro personalizzate adattate alla visualizzazione standardizzata del prodotto
- Gestione delle applicazioni tramite canali conda gestiti da Deadline Cloud
- Licenze basate sull'utilizzo configurate automaticamente per il software supportato
- Integrazione diretta con Amazon S3 per la gestione degli asset
- Applicazione di monitoraggio personalizzata integrata con i sistemi di gestione dei prodotti esistenti
- Flotte gestite dai servizi per una scalabilità elastica

Questo approccio consente l'elaborazione di migliaia di prodotti al giorno, generando automaticamente visualizzazioni di prodotto standardizzate come le animazioni dei giradischi. L'infrastruttura gestita dai servizi si ridimensiona automaticamente per soddisfare la domanda variabile, mantenendo al contempo l'efficienza dei costi attraverso il riutilizzo da parte dei lavoratori e l'implementazione ottimizzata delle applicazioni.


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

Whitelabel/OEM/B2C Cliente

Il software DCC (Digital Content Creation) tradizionale richiede in genere agli utenti di mantenere la propria infrastruttura di rendering o di elaborare i rendering localmente sulla propria workstation, con conseguenti investimenti hardware significativi o lunghi tempi di attesa che interrompono i flussi di lavoro creativi. Per i fornitori di software, fornire funzionalità di rendering su cloud richiedeva tradizionalmente la creazione e la manutenzione di infrastrutture e sistemi di fatturazione complessi.

Un'implementazione di Deadline Cloud integrata nel software B2C consente un rendering cloud senza interruzioni direttamente all'interno dell'interfaccia familiare dell'utente. Questa integrazione combina:

- Invio interattivo dei lavori tramite flusso di lavoro integrato nell'applicazione DCC
- Canali conda gestiti da Deadline Cloud per la distribuzione delle applicazioni di rendering
- Licenze basate sull'utilizzo configurate automaticamente
- Gestione delle risorse tramite allegati di lavoro con storage gestito dal fornitore
- Monitoraggio personalizzato integrato direttamente nell'interfaccia DCC
- Flotte gestite dai servizi condivise tra gli utenti

Questo approccio consente agli utenti finali di inviare i rendering al cloud con un solo clic dall'interno del software, senza gestire account, infrastrutture o configurazioni complesse. Il fornitore del software mantiene un ambiente multi-tenant in cui:

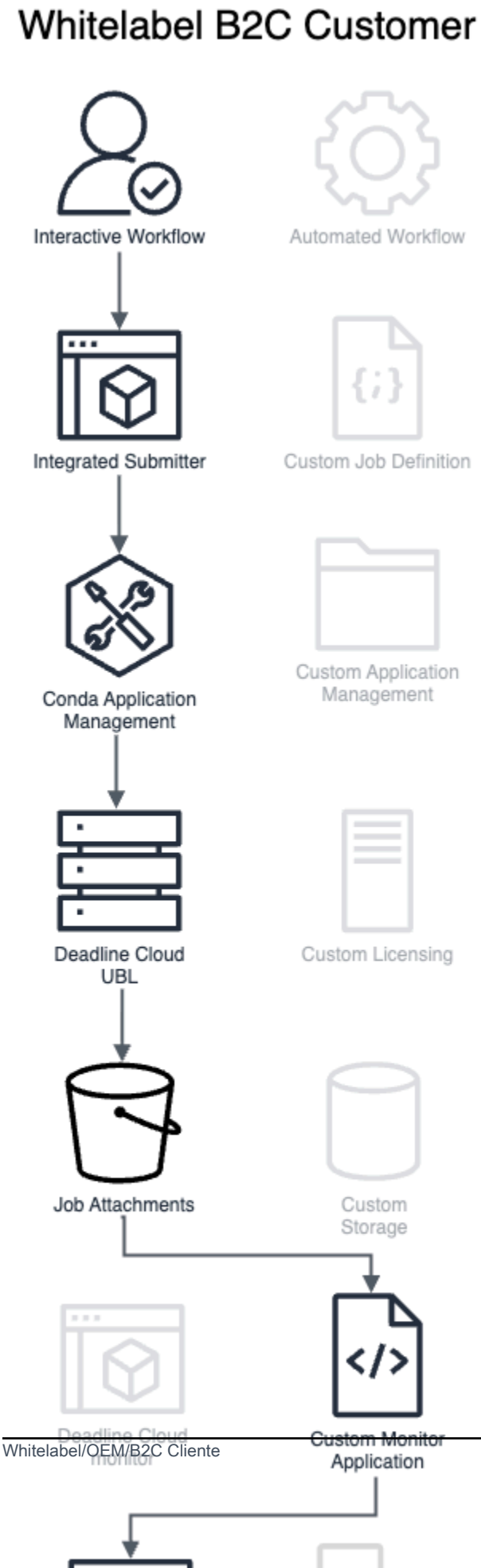
- Gli utenti si autenticano tramite le credenziali software esistenti
- I lavori vengono indirizzati automaticamente a code dedicate per utente
- Le risorse sono isolate in modo sicuro utilizzando prefissi di archiviazione controllati da IAM
- La fatturazione viene gestita tramite i sistemi esistenti del fornitore
- Lo stato del lavoro e gli output vengono trasmessi direttamente all'applicazione dell'utente

L'approccio alla flotta condivisa garantisce prestazioni ottimali mantenendo un pool di dipendenti sempre aggiornato, riducendo al minimo i tempi di avvio e massimizzando l'utilizzo delle risorse da parte di tutta la base di utenti. Questa configurazione consente ai fornitori di software di offrire il rendering su cloud come funzionalità di prodotto senza interruzioni anziché come servizio separato che richiede configurazioni o account aggiuntivi.

Gli utenti finali traggono vantaggio da:

- Invio con un clic dalla loro interfaccia familiare
- Pay-as-you-go prezzi senza gestione dell'infrastruttura
- Tempi rapidi di avvio dei lavori grazie all'infrastruttura condivisa
- Scaricamento e organizzazione automatici dei rendering completati
- Esperienza coerente su tutte le piattaforme

Questo modello di integrazione consente ai fornitori di software di fornire funzionalità di rendering di livello aziendale a tutta la loro base di utenti, mantenendo al contempo un'esperienza semplice e intuitiva che sembra tipica della loro applicazione.


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

Cos'è un carico di lavoro Deadline Cloud

Con AWS Deadline Cloud, puoi inviare offerte di lavoro per eseguire le tue applicazioni nel cloud ed elaborare dati per la produzione di contenuti o approfondimenti fondamentali per la tua attività. Deadline Cloud utilizza Open Job Description (OpenJD) come sintassi per i modelli di lavoro, una specifica progettata per le esigenze delle pipeline di calcolo visivo ma applicabile a molti altri casi d'uso. Alcuni esempi di carichi di lavoro includono il rendering di grafica computerizzata, la simulazione fisica e la fotogrammetria.

I carichi di lavoro spaziano da semplici pacchetti di lavoro che gli utenti inviano a una coda con la CLI o una GUI generata automaticamente, a plug-in di invio integrati che generano dinamicamente un pacchetto di lavoro per un carico di lavoro definito dall'applicazione.

In che modo i carichi di lavoro derivano dalla produzione

Per comprendere i carichi di lavoro nei contesti di produzione e come supportarli con Deadline Cloud, considera come si sono evoluti. La produzione può comportare la creazione di effetti visivi, animazioni, giochi, immagini di cataloghi di prodotti, ricostruzioni 3D per il Building Information Modeling (BIM) e altro ancora. Questi contenuti vengono in genere creati da un team di specialisti artistici o tecnici che eseguono una varietà di applicazioni software e script personalizzati. I membri del team si scambiano i dati tra loro utilizzando una pipeline di produzione. Molte attività eseguite dalla pipeline implicano calcoli intensivi che richiederebbero giorni se eseguiti sulla workstation di un utente.

Alcuni esempi di attività in queste pipeline di produzione includono:

- Utilizzo di un'applicazione di fotogrammetria per elaborare fotografie scattate da un set cinematografico per ricostruire una mesh digitale strutturata.
- Esecuzione di una simulazione di particelle in una scena 3D per aggiungere livelli di dettaglio all'effetto visivo di un'esplosione per uno show televisivo.
- Inserimento dei dati di un livello di gioco nel formato necessario per il rilascio esterno e applicazione delle impostazioni di ottimizzazione e compressione.
- Rendering di un set di immagini per un catalogo di prodotti che include variazioni di colore, sfondo e illuminazione.
- Esecuzione di uno script sviluppato su misura su un modello 3D per applicare un look personalizzato e approvato da un regista.

Queste attività coinvolgono molti parametri da regolare per ottenere un risultato artistico o per ottimizzare la qualità dell'output. Spesso è disponibile un'interfaccia grafica per selezionare i valori dei parametri con un pulsante o un menu per eseguire il processo localmente all'interno dell'applicazione. Quando un utente esegue il processo, l'applicazione e probabilmente il computer host stesso non possono essere utilizzati per eseguire altre operazioni perché utilizza lo stato dell'applicazione in memoria e può consumare tutte le risorse di CPU e memoria del computer host.

In molti casi il processo è rapido. Nel corso della produzione, la velocità del processo rallenta all'aumentare dei requisiti di qualità e complessità. Un test del personaggio che ha richiesto 30 secondi durante lo sviluppo può facilmente trasformarsi in 3 ore se applicato al personaggio di produzione finale. Grazie a questa progressione, un carico di lavoro nato all'interno di una GUI può diventare troppo grande per essere contenuto. Il trasferimento su Deadline Cloud può aumentare la produttività degli utenti che eseguono questi processi perché riacquistano il pieno controllo della propria workstation e possono tenere traccia di più iterazioni dal monitor Deadline Cloud.

Esistono due livelli di supporto a cui puntare quando si sviluppa il supporto per un carico di lavoro in Deadline Cloud:

- Trasferimento del carico di lavoro dalla workstation dell'utente a una farm Deadline Cloud senza parallelismo o accelerazione. Ciò potrebbe sottoutilizzare le risorse di calcolo disponibili nella farm, ma la possibilità di spostare operazioni lunghe su un sistema di elaborazione in batch consente agli utenti di fare di più con la propria workstation.
- Ottimizzazione del parallelismo del carico di lavoro in modo che utilizzi la scala orizzontale della Deadline Cloud farm per un completamento rapido.

A volte è ovvio come far funzionare un carico di lavoro in parallelo. Ad esempio, ogni fotogramma di un rendering di grafica computerizzata può essere eseguito indipendentemente. Tuttavia, è importante non rimanere bloccati su questo parallelismo. Tieni invece presente che trasferire un carico di lavoro di lunga durata su Deadline Cloud offre vantaggi significativi, anche quando non esiste un modo ovvio per suddividere il carico di lavoro.

Gli ingredienti di un carico di lavoro

Per specificare un carico di lavoro Deadline Cloud, implementa un job bundle che gli utenti inviano a una coda con la CLI di Deadline Cloud. Gran parte del lavoro necessario per creare un pacchetto di lavoro consiste nella stesura del modello di lavoro, ma ci sono altri fattori, come la modalità di fornitura delle applicazioni richieste dal carico di lavoro. Ecco gli aspetti essenziali da considerare quando si definisce un carico di lavoro per Deadline Cloud:

- L'applicazione da eseguire. Il processo deve essere in grado di avviare i processi applicativi e richiede pertanto un'installazione dell'applicazione disponibile e tutte le licenze utilizzate dall'applicazione, ad esempio l'accesso a un server di licenze flottante. In genere fa parte della configurazione della farm e non è incorporata nel job bundle stesso.
  - Configurazione dei lavori utilizzando ambienti di coda
  - Connect le flotte gestite dai clienti a un endpoint di licenza
- Definizioni dei parametri Job. L'esperienza dell'utente nell'invio del lavoro è fortemente influenzata dai parametri forniti. I parametri di esempio includono file di dati, directory e configurazione dell'applicazione.
  - Valori dei parametri, elementi per i job bundle.
- Flusso di dati sui file. Quando un processo viene eseguito, legge l'input dai file forniti dall'utente, quindi scrive l'output come nuovi file. Per utilizzare gli allegati del lavoro e le funzionalità di mappatura dei percorsi, il job deve specificare i percorsi delle directory o dei file specifici per questi input e output.
  - Utilizzo dei file nei lavori
- Lo script dello step. Lo script step esegue il file binario dell'applicazione con le opzioni della riga di comando corrette per applicare i parametri di processo forniti. Gestisce anche dettagli come la mappatura dei percorsi se i file di dati del carico di lavoro includono riferimenti di percorso assoluti anziché relativi.
  - Elementi del modello di lavoro per i pacchetti di lavoro

Portabilità del carico di lavoro

Un carico di lavoro è portatile quando può essere eseguito su più sistemi diversi senza modificarlo ogni volta che si invia un lavoro. Ad esempio, potrebbe essere eseguito su diverse render farm su cui sono montati diversi file system condivisi o su sistemi operativi diversi come Linux oWindows. Quando si implementa un pacchetto di lavori portatile, è più facile per gli utenti eseguire il lavoro nella propria farm specifica o adattarlo ad altri casi d'uso.

Ecco alcuni modi in cui puoi rendere portatile il tuo job bundle.

- Specificate in modo completo i file di dati di input necessari per un carico di lavoro, utilizzando i parametri del `PATH` lavoro e i riferimenti alle risorse nel job bundle. Questo approccio rende il lavoro trasferibile alle farm basate su file system condivisi e alle farm che creano copie dei dati di input, come la funzionalità degli allegati dei lavori di Deadline Cloud.

- Rendi i riferimenti ai percorsi dei file di input del lavoro rilocabili e utilizzabili su diversi sistemi operativi. Ad esempio, quando gli utenti inviano lavori dalle Windows workstation per eseguirli su un parco macchine. Linux
  - Utilizzate riferimenti relativi ai percorsi dei file, in modo che se la directory che li contiene viene spostata in una posizione diversa, i riferimenti si risolvono comunque. Alcune applicazioni, come Blender, supportano la scelta tra percorsi relativi e assoluti.
  - Se non puoi usare percorsi relativi, supporta i metadati di mappatura dei percorsi OpenJD e traduci i percorsi assoluti in base al modo in cui Deadline Cloud fornisce i file al lavoro.
- Implementa i comandi in un lavoro utilizzando script portatili. Python e bash sono due esempi di linguaggi di scripting che possono essere usati in questo modo. Dovreste considerare la possibilità di fornirli entrambi su tutti gli host di lavoro delle vostre flotte.
  - Usa il binario dell'interprete di script, ad esempio `python` o`bash`, con il nome del file di script come argomento. Questo approccio funziona su tutti i sistemi operativiWindows, incluso, a differenza dell'utilizzo di un file di script con il bit di esecuzione impostato. Linux
  - Scrivi script bash portatili applicando queste pratiche:
    - Espandi i parametri del percorso del modello tra virgolette singole per gestire percorsi con spazi e separatori di Windows percorso.
    - Quando è in esecuzioneWindows, presta attenzione ai problemi relativi alla traduzione automatica dei percorsi di MingW. Ad esempio, trasforma un AWS CLI comando like `aws logs tail /aws/deadline/...` in un comando simile a un registro `aws logs tail "C:/Program Files/Git/aws/deadline/..."` e non esegue correttamente la coda. Imposta la variabile `MSYS_NO_PATHCONV=1` per disattivare questo comportamento.
    - Nella maggior parte dei casi, lo stesso codice funziona su tutti i sistemi operativi. Quando il codice deve essere diverso, usa un `if/else` costrutto per gestire i casi.

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

  - È possibile scrivere script Python portatili utilizzando `pathlib` per gestire le differenze di percorso del file system ed evitare funzionalità specifiche del funzionamento. La documentazione di Python include annotazioni per questo, ad esempio nella documentazione della libreria dei segnali. Linux-il supporto di funzionalità specifiche è contrassegnato come «Disponibilità: Linux».

- Utilizza i parametri del processo per specificare i requisiti dell'applicazione. Utilizzate convenzioni coerenti che l'amministratore della farm può applicare negli ambienti di coda.
  - Ad esempio, è possibile utilizzare i `RezPackages` parametri `CondaPackages` e/o nel job, con un valore di parametro predefinito che elenca i nomi e le versioni dei pacchetti applicativi richiesti dal job. Quindi, è possibile utilizzare uno degli ambienti di coda conda o Rez di esempio per fornire un ambiente virtuale per il lavoro.

Guida introduttiva alle risorse di Deadline Cloud

Per iniziare a creare soluzioni personalizzate per AWS Deadline Cloud, devi configurare le tue risorse. Queste includono una fattoria, almeno una coda per l'azienda agricola e almeno una flotta di lavoratori per la manutenzione della coda. Puoi creare le tue risorse utilizzando la console Deadline Cloud oppure puoi utilizzare il. AWS Command Line Interface

In questo tutorial, lo utilizzerai AWS CloudShell per creare una semplice farm per sviluppatori ed eseguire il worker agent. Potrai quindi inviare ed eseguire un semplice lavoro con parametri e allegati, aggiungere una flotta gestita dai servizi e ripulire le risorse della fattoria quando hai finito.

Le sezioni seguenti illustrano le diverse funzionalità di Deadline Cloud e il modo in cui funzionano e interagiscono. Seguire questi passaggi è utile per sviluppare e testare nuovi carichi di lavoro e personalizzazioni.

Per istruzioni su come configurare la tua fattoria utilizzando la console, consulta Guida introduttiva nella Guida per l'utente di Deadline Cloud.

Argomenti

- Crea una cloud farm di Deadline
- Esegui l'agente di lavoro Deadline Cloud
- Invia con Deadline Cloud
- Invia offerte di lavoro con allegati di lavoro in Deadline Cloud
- Aggiungi una flotta gestita dai servizi alla tua farm di sviluppatori in Deadline Cloud
- Pulisci le risorse della tua azienda agricola in Deadline Cloud

Crea una cloud farm di Deadline

Per creare la tua farm per sviluppatori e le risorse in coda in AWS Deadline Cloud, usa AWS Command Line Interface (AWS CLI), come mostrato nella procedura seguente. Inoltre, creerai un ruolo AWS Identity and Access Management (IAM) e una flotta gestita dal cliente (CMF) e assocerai la flotta alla tua coda. Quindi puoi configurare AWS CLI e confermare che la tua farm sia configurata e funzioni come specificato.

Puoi utilizzare questa farm per esplorare le funzionalità di Deadline Cloud, quindi sviluppare e testare nuovi carichi di lavoro, personalizzazioni e integrazioni di pipeline.

Per creare una fattoria

1. Aprire una AWS CloudShell sessione. Utilizzerai la CloudShell finestra per inserire i comandi AWS Command Line Interface (AWS CLI) per eseguire gli esempi di questo tutorial. Mantieni la CloudShell finestra aperta mentre procedi.
2. Crea un nome per la tua fattoria e aggiungi il nome della fattoria a`~/.bashrc`. In questo modo sarà disponibile per altre sessioni terminali.

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

3. Crea la risorsa della fattoria e aggiungi il relativo ID della fattoria a`~/.bashrc`.

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

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

4. Crea la risorsa della coda e aggiungi il relativo ID di coda a `~/.bashrc`.

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

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

5. Crea un ruolo IAM per la flotta. Questo ruolo fornisce agli host dei lavoratori del tuo parco macchine le credenziali di sicurezza necessarie per eseguire i lavori dalla tua coda.

```
aws iam create-role \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --assume-role-policy-document \
        '{
```

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

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

6. Crea la flotta gestita dal cliente (CMF) e aggiungi il relativo ID della flotta a. `~/.bashrc`

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

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
    --farm-id \$DEV_FARM_ID \
    --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
```

```
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. Associa il CMF alla tua coda.

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

8. Installa l'interfaccia a riga di comando di Deadline Cloud.

```
pip install deadline
```

9. Per impostare la farm predefinita sull'ID della fattoria e la coda sull'ID della coda che hai creato in precedenza, usa il comando seguente.

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

10. (Facoltativo) Per confermare che la fattoria è configurata in base alle specifiche, utilizzate i seguenti comandi:

    - Elenca tutte le fattorie — **`deadline farm list`**
    - Elenca tutte le code nella farm predefinita: **`deadline queue list`**
    - Elenca tutte le flotte nella fattoria predefinita: **`deadline fleet list`**
    - Ottieni la fattoria predefinita: **`deadline farm get`**
    - Ottieni la coda predefinita: **`deadline queue get`**
    - Ottieni tutte le flotte associate alla coda predefinita: **`deadline fleet get`**

Passaggi successivi

Dopo aver creato la tua fattoria, puoi far funzionare l'operatore Deadline Cloud sugli host della tua flotta per elaborare i lavori. Per informazioni, consulta Esegui l'agente di lavoro Deadline Cloud.

Esegui l'agente di lavoro Deadline Cloud

Prima di poter eseguire i lavori che invii alla coda nella tua farm di sviluppatori, devi eseguire il worker agent di AWS Deadline Cloud in modalità sviluppatore su un worker host.

Nel resto di questo tutorial, eseguirai AWS CLI operazioni sulla tua farm di sviluppatori utilizzando due schede. AWS CloudShell Nella prima scheda, puoi inviare offerte di lavoro. Nella seconda scheda, puoi eseguire l'agente di lavoro.

> **Note**
>
> Se lasci la CloudShell sessione inattiva per più di 20 minuti, scade il timeout e interrompe l'agente di lavoro. Per riavviare l'agente di lavoro, segui le istruzioni nella procedura seguente.

Prima di poter avviare un worker agent, devi configurare una farm, una coda e una flotta di Deadline Cloud. Consultare Crea una cloud farm di Deadline.

Per eseguire l'agente di lavoro in modalità sviluppatore

1. Con la fattoria ancora aperta nella prima CloudShell scheda, apri una seconda CloudShell scheda, quindi crea le `demoenv-persist` cartelle `demoenv-logs` e.

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

2. Scarica e installa i pacchetti Deadline Cloud worker agent da PyPI:

   > **Note**
   >
   > Abilitato Windows, è necessario che i file dell'agente siano installati nella directory globale dei pacchetti del sito di Python. Gli ambienti virtuali Python non sono attualmente supportati.

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

3. Per consentire all'agente di lavoro di creare le directory temporanee per i lavori in esecuzione, crea una directory:

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

4. Esegui il worker agent Deadline Cloud in modalità sviluppatore con `DEV_FARM_ID` le variabili `DEV_CMF_ID` che hai aggiunto a. `~/.bashrc`

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

Quando l'agente di lavoro inizializza e quindi esegue il polling del funzionamento dell'`UpdateWorkerSchedule`API, viene visualizzato il seguente output:

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

5. Seleziona la prima CloudShell scheda, quindi elenca i lavoratori della flotta.

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

Viene visualizzato un output come il seguente:

```
Displaying 1 of 1 workers starting at 0

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

In una configurazione di produzione, il worker agent di Deadline Cloud richiede la configurazione di più utenti e directory di configurazione come utente amministrativo sulla macchina host. Puoi ignorare queste impostazioni perché stai eseguendo lavori nella tua farm di sviluppo, a cui solo tu puoi accedere.

Passaggi successivi

Ora che un worker agent è in esecuzione sui tuoi host di lavoro, puoi inviare lavori ai tuoi lavoratori. È possibile:

- Invia con Deadline Cloudutilizzando un semplice pacchetto di lavori OpenJD.
- Invia offerte di lavoro con allegati di lavoro in Deadline Cloudche condividono file tra postazioni di lavoro che utilizzano sistemi operativi diversi.

Invia con Deadline Cloud

Per eseguire lavori Deadline Cloud sui tuoi host di lavoro, crei e utilizzi un pacchetto di lavori Open Job Description (OpenJD) per configurare un lavoro. Il pacchetto configura il lavoro, ad esempio specificando i file di input per un lavoro e dove scrivere l'output del lavoro. Questo argomento include esempi di modi in cui è possibile configurare un job bundle.

Prima di poter seguire le procedure in questa sezione, è necessario completare quanto segue:

- Crea una cloud farm di Deadline
- Esegui l'agente di lavoro Deadline Cloud

Per utilizzare AWS Deadline Cloud per eseguire lavori, utilizza le seguenti procedure. Usa la prima AWS CloudShell scheda per inviare lavori alla tua farm di sviluppatori. Usa la seconda CloudShell scheda per visualizzare l'output del worker agent.

Argomenti

- Invia il simple_job campione
- Invia un messaggio simple_job con un parametro
- Crea un pacchetto di job simple_file_job con file I/O
- Fasi successive

Invia il simple_job campione

Dopo aver creato una fattoria e aver avviato l'agente operaio, puoi inviare il simple_job campione a Deadline Cloud.

Per inviare il simple_job campione a Deadline Cloud

1. Scegli la tua prima CloudShell scheda.
2. Scarica l'esempio da GitHub.

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

3. Vai alla directory degli esempi del job bundle.

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

4. Invia il simple_job campione.

```
deadline bundle submit simple_job
```

5. Scegli la seconda CloudShell scheda per visualizzare l'output di registrazione relativo alle chiamate`BatchGetJobEntities`, all'attivazione di una sessione e all'esecuzione di un'azione di sessione.

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
 # [session-053d77cef82648fe2] Starting new Session.
 [queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
 resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
 'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
 'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
 {'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
 75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
 params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
 'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'},
 'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
```

```
 '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
 request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
 [session-053d77cef82648fea9c69827182] Appended new SessionActions.
 (ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
 [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
 [session-053d77cef82648fea9c69827182] Running as the Worker Agent's
 user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
 [session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
 no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
 Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: local
 file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

> **Note**
>
> Viene mostrato solo l'output di registrazione del worker agent. Esiste un registro separato per la sessione che esegue il lavoro.

6. Scegli la tua prima scheda, quindi controlla i file di registro scritti dal worker agent.

   a. Passa alla directory dei registri del worker agent e visualizzane il contenuto.

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

   b. Stampa il primo file di registro creato dal worker agent.

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

Questo file contiene l'output dell'agente di lavoro su come ha chiamato l'API Deadline Cloud per creare una risorsa worker nel parco macchine e poi ha assunto il ruolo del parco macchine.

c. Stampa l'output del file di registro quando l'agente lavoratore si unisce alla flotta.

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

Questo registro contiene output su tutte le azioni intraprese dall'agente di lavoro, ma non contiene output sulle code da cui esegue i lavori, ad eccezione IDs di quelle risorse.

d. Stampa i file di registro per ogni sessione in una directory con lo stesso nome dell'id della risorsa della coda.

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

Se il processo ha esito positivo, l'output del file di registro sarà simile al seguente:

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

2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
2024-03-27 16:00:22,404 INFO
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,405 INFO --------- Running Task
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Phase: Setup
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
```

```
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO --------- Session Cleanup
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a
```

7. Stampa le informazioni sul lavoro.

   ```
   deadline job get
   ```

   Quando inviate il lavoro, il sistema lo salva come predefinito in modo da non dover inserire l'ID del lavoro.

Invia un messaggio simple_job con un parametro

Puoi inviare lavori con parametri. Nella procedura seguente, si modifica il simple_job modello per includere un messaggio personalizzato, si invia il file di registro della simple_job sessione e si stampa il file di registro della sessione per visualizzare il messaggio.

Per inviare l'simple_jobesempio con un parametro

1. Seleziona la tua prima CloudShell scheda, quindi vai alla directory degli esempi di job bundle.

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

2. Stampa il contenuto del simple_job modello.

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

   La `parameterDefinitions` sezione con il `Message` parametro dovrebbe avere l'aspetto seguente:

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

3. Inviate l'simple_jobesempio con un valore di parametro, quindi attendete che il processo finisca di funzionare.

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

4. Per visualizzare il messaggio personalizzato, visualizza il file di registro della sessione più recente.

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

Crea un pacchetto di job simple_file_job con file I/O

Un lavoro di rendering deve leggere la definizione della scena, renderizzare un'immagine a partire da essa e quindi salvare l'immagine in un file di output. È possibile simulare questa azione facendo in modo che il job calcoli l'hash dell'input anziché renderizzare un'immagine.

Per creare un pacchetto di lavoro simple_file_job con file I/O

1. Seleziona la prima CloudShell scheda, quindi vai alla directory degli esempi di job bundle.

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

2. Crea una copia `simple_job` con il nuovo nome`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Modifica il modello di lavoro come segue:

> **Note**
>
> Ti consigliamo di utilizzarlo nano per questi passaggi. Se si preferisce utilizzareVim, è necessario impostarne la modalità di incolla utilizzando`:set paste`.

   a. Apri il modello in un editor di testo.

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

   b. Aggiungi quanto segue `type`e `objectType`, e `dataFlow`e `parameterDefinitions`.

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

c. Aggiungere il seguente comando di `bash` script alla fine del file che legge dal file di input e scrive nel file di output.

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

L'aggiornamento `template.yaml` dovrebbe corrispondere esattamente a quanto segue:

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      runnable: true
```

```
      data: |
        #!/usr/bin/env bash
        echo "{{Param.Message}}"

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

> **Note**
>
> Se desideri regolare la spaziatura in`template.yaml`, assicurati di utilizzare spazi anziché rientri.

d. Salvate il file e uscite dall'editor di testo.

4. Fornite i valori dei parametri per i file di input e output per inviare il simple_file_job.

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

5. Stampa le informazioni sul lavoro.

```
deadline job get
```

- Verrà visualizzato un output come il seguente:

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

- Sebbene abbiate fornito solo percorsi relativi, per i parametri è impostato il percorso completo. AWS CLI Unisce la directory di lavoro corrente a tutti i percorsi forniti come parametri quando i percorsi hanno il tipoPATH.
- L'agente di lavoro in esecuzione nell'altra finestra del terminale rileva ed esegue il lavoro. Questa azione crea il `hash.txt` file, che è possibile visualizzare con il seguente comando.

```
cat hash.txt
```

Questo comando stamperà un output simile al seguente.

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

Fasi successive

Dopo aver appreso come inviare lavori semplici utilizzando la CLI di Deadline Cloud, puoi scoprire:

- Invia offerte di lavoro con allegati di lavoro in Deadline Cloudper imparare a eseguire lavori su host che eseguono sistemi operativi diversi.
- Aggiungi una flotta gestita dai servizi alla tua farm di sviluppatori in Deadline Cloudper eseguire i tuoi lavori su host gestiti da Deadline Cloud.
- Pulisci le risorse della tua azienda agricola in Deadline Cloudper chiudere le risorse che hai usato per questo tutorial.

Invia offerte di lavoro con allegati di lavoro in Deadline Cloud

Molte farm utilizzano file system condivisi per condividere file tra gli host che inviano i job e quelli che eseguono i job. Ad esempio, nell'`simple_file_jo`besempio precedente, il file system locale è condiviso tra le finestre del AWS CloudShell terminale, che vengono eseguite nella scheda uno in cui si invia il lavoro, e nella scheda due in cui si esegue il worker agent.

Un file system condiviso è vantaggioso quando la postazione di lavoro del mittente e gli host del lavoratore si trovano sulla stessa rete locale. Se memorizzi i dati in locale vicino alle workstation che vi accedono, l'utilizzo di una farm basata sul cloud significa dover condividere i file system tramite una VPN ad alta latenza o sincronizzare i file system nel cloud. Nessuna di queste opzioni è facile da configurare o utilizzare.

AWS Deadline Cloud offre una soluzione semplice con allegati di lavoro, simili agli allegati delle e-mail. Con gli allegati di lavoro, alleghi dati al tuo lavoro. Deadline Cloud gestisce quindi i dettagli del trasferimento e dell'archiviazione dei dati di lavoro nei bucket Amazon Simple Storage Service (Amazon S3).

I flussi di lavoro per la creazione di contenuti sono spesso iterativi, il che significa che un utente invia lavori con un piccolo sottoinsieme di file modificati. Poiché i bucket Amazon S3 archiviano gli allegati del lavoro in uno storage content-addressable, il nome di ogni oggetto si basa sull'hash dei dati dell'oggetto e il contenuto di un albero di directory viene archiviato in un formato di file manifesto allegato a un lavoro.

Prima di poter seguire le procedure in questa sezione, devi completare quanto segue:

- Crea una cloud farm di Deadline
- Esegui l'agente di lavoro Deadline Cloud

Per eseguire lavori con allegati di lavoro, completare i passaggi seguenti.

Argomenti

- Aggiungi una configurazione degli allegati di lavoro alla tua coda
- Invia simple_file_job con allegati di lavoro
- Informazioni su come vengono archiviati gli allegati di lavoro in Amazon S3
- Fasi successive

Aggiungi una configurazione degli allegati di lavoro alla tua coda

Per abilitare gli allegati dei lavori nella tua coda, aggiungi una configurazione degli allegati di lavoro alla risorsa di coda del tuo account.

Per aggiungere una configurazione degli allegati di lavoro alla tua coda

1. Scegli la tua prima CloudShell scheda, quindi inserisci uno dei seguenti comandi per utilizzare un bucket Amazon S3 per gli allegati dei lavori.
   - Se non disponi di un bucket Amazon S3 privato esistente, puoi creare e utilizzare un nuovo bucket S3.

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

```
    $LOCATION_CONSTRAINT \
    --acl private \
    --bucket ${DEV_FARM_BUCKET}
```

- Se disponi già di un bucket Amazon S3 privato, puoi utilizzarlo sostituendolo *MY_BUCKET_NAME* con il nome del tuo bucket.

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

2. Dopo aver creato o scelto il bucket Amazon S3, aggiungi il nome del bucket a per renderlo disponibile `~/.bashrc` per altre sessioni di terminale.

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

3. Crea un ruolo AWS Identity and Access Management (IAM) per la coda.

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess \
    --policy-document \
            '{
                "Version": "2012-10-17",
                "Statement": [
                {
                    "Action": [
                        "s3:GetObject*",
                        "s3:GetBucket*",
                        "s3:List*",
```

```
                    "s3:DeleteObject*",
                    "s3:PutObject",
                    "s3:PutObjectLegalHold",
                    "s3:PutObjectRetention",
                    "s3:PutObjectTagging",
                    "s3:PutObjectVersionTagging",
                    "s3:Abort*"
                ],
                "Resource": [
                    "arn:aws:s3:::'$DEV_FARM_BUCKET'",
                    "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
                ],
                "Effect": "Allow"
            }
        ]
        }'
```

4. Aggiorna la coda per includere le impostazioni degli allegati di lavoro e il ruolo IAM.

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

5. Conferma di aver aggiornato la coda.

```
deadline queue get
```

Viene visualizzato un output come il seguente:

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

```
...
```

Invia simple_file_job con allegati di lavoro

Quando utilizzi gli allegati di lavoro, i pacchetti di lavoro devono fornire a Deadline Cloud informazioni sufficienti per determinare il flusso di dati del lavoro, ad esempio l'utilizzo dei parametri. PATH Nel caso disimple_file_job, hai modificato il `template.yaml` file per indicare a Deadline Cloud che il flusso di dati si trova nel file di input e nel file di output.

Dopo aver aggiunto la configurazione degli allegati di lavoro alla coda, puoi inviare l'esempio di simple_file_job con gli allegati del lavoro. Dopo aver eseguito questa operazione, è possibile visualizzare la registrazione e l'output del lavoro per confermare che l'operazione con gli allegati del lavoro funziona. simple_file_job

Per inviare il pacchetto di lavoro simple_file_job con gli allegati del lavoro

1. Scegli la tua prima CloudShell scheda, quindi apri la cartella. `JobBundle-Samples`
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```
3. Invia simple_file_job alla coda. Quando ti viene richiesto di confermare il caricamento, inserisci. **y**

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```
4. Per visualizzare l'output del registro della sessione di trasferimento dati degli allegati del lavoro, eseguite il comando seguente.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```
5. Elenca le azioni della sessione eseguite all'interno della sessione.

```
aws deadline list-session-actions \
```

```
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID
```

Viene mostrato un output come il seguente:

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

La prima azione della sessione ha scaricato gli allegati del lavoro di input, mentre la seconda azione esegue l'attività come nei passaggi precedenti e quindi ha caricato gli allegati del lavoro di output.

6. Elenca la directory di output.

```
ls *.txt
```

Un output come questo `hash.txt` esiste nella directory, ma `hash-jobattachments.txt` non esiste perché il file di output del processo non è stato ancora scaricato.

7. Scarica l'output del processo più recente.

```
deadline job download-output
```

8. Visualizza l'output del file scaricato.

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

Viene mostrato un output come il seguente:

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

Informazioni su come vengono archiviati gli allegati di lavoro in Amazon S3

Puoi utilizzare AWS Command Line Interface (AWS CLI) per caricare o scaricare dati per gli allegati di lavoro, che vengono archiviati nei bucket Amazon S3. Capire come Deadline Cloud archivia gli allegati di lavoro su Amazon S3 ti aiuterà a sviluppare carichi di lavoro e integrazioni di pipeline.

Per controllare come vengono archiviati gli allegati dei lavori di Deadline Cloud in Amazon S3

1. Scegli la tua prima CloudShell scheda, quindi apri la directory degli esempi di job bundle.

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

2. Ispeziona le proprietà del lavoro.

```
deadline job get
```

Viene mostrato un output come il seguente:

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
```

```
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
template.yaml
  OutFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
jobattachments.txt
attachments:
  manifests:
  - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
    rootPathFormat: posix
    outputRelativeDirectories:
    - .
    inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
    inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
  fileSystem: COPIED
```

Il campo degli allegati contiene un elenco di strutture manifeste che descrivono i percorsi dei dati di input e output utilizzati dal job durante l'esecuzione. Guarda `rootPath` per vedere il percorso della directory locale sul computer che ha inviato il lavoro. Per visualizzare il suffisso dell'oggetto Amazon S3 che contiene un file manifesto, consulta il. `inputManifestFile` Il file manifest contiene i metadati per un'istantanea ad albero di directory dei dati di input del processo.

3. Stampa bene l'oggetto manifest di Amazon S3 per vedere la struttura della directory di input per il lavoro.

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

Viene mostrato un output come il seguente:

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "2ec297b04c59c4741ed97ac8fb83080c",
```

```
            "mtime": 1698186190000000,
            "path": "simple_job/template.yaml",
            "size": 445
        }
        ],
        "totalSize": 445
    }
```

4. Crea il prefisso Amazon S3 che contiene i manifest per gli allegati del processo di output ed elenca l'oggetto al suo interno.

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

Gli allegati del lavoro di output non sono referenziati direttamente dalla risorsa del lavoro, ma vengono invece inseriti in un bucket Amazon S3 basato sulla risorsa della fattoria. IDs

5. Ottieni la chiave dell'oggetto manifesto più recente per l'ID di azione della sessione specifica, quindi stampa in modo semplice gli oggetti del manifesto.

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

`hash-jobattachments.txt`Nell'output vedrete le proprietà del file, come le seguenti:

```
{
```

```
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
    {
        "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
        "mtime": 1698785252554950,
        "path": "hash-jobattachments.txt",
        "size": 182
    }
    ],
    "totalSize": 182
}
```

Il job avrà un solo oggetto manifesto per operazione eseguita, ma in generale è possibile avere più oggetti per operazione eseguita.

6. Visualizza l'output di storage Amazon S3 indirizzabile ai contenuti sotto il prefisso. `Data`

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

Viene mostrato un output come il seguente:

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

Fasi successive

Dopo aver appreso come inviare offerte di lavoro con allegati utilizzando la CLI di Deadline Cloud, puoi scoprire:

- Invia con Deadline Cloudper imparare a eseguire lavori utilizzando un pacchetto OpenJD sui tuoi host di lavoro.
- Aggiungi una flotta gestita dai servizi alla tua farm di sviluppatori in Deadline Cloudper eseguire i tuoi lavori su host gestiti da Deadline Cloud.

- Pulisci le risorse della tua azienda agricola in Deadline Cloudper chiudere le risorse che hai usato per questo tutorial.

Aggiungi una flotta gestita dai servizi alla tua farm di sviluppatori in Deadline Cloud

AWS CloudShell non fornisce una capacità di elaborazione sufficiente per testare carichi di lavoro più grandi. Inoltre, non è configurato per funzionare con lavori che distribuiscono le attività su più host di lavoro.

Invece di utilizzarla CloudShell, puoi aggiungere una flotta gestita dai servizi di Auto Scaling (SMF) alla tua farm di sviluppatori. Un SMF offre una capacità di elaborazione sufficiente per carichi di lavoro più grandi ed è in grado di gestire lavori che richiedono la distribuzione delle attività lavorative su più host di lavoro.

Prima di aggiungere un SMF, devi configurare una farm, una coda e una flotta Deadline Cloud. Per informazioni, consulta Crea una cloud farm di Deadline.

Per aggiungere una flotta gestita dai servizi alla tua farm di sviluppatori

1. Scegli la tua prima AWS CloudShell scheda, quindi crea la flotta gestita dai servizi e aggiungi il relativo ID della flotta a. `.bashrc` Questa azione lo rende disponibile per altre sessioni terminali.

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

```
                    },
                    "osFamily": "linux",
                    "cpuArchitectureType": "x86_64"
                },
                "instanceMarketOptions": {
                    "type": "spot"
                }
            }
        }'

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

2. Associate l'SMF alla vostra coda.

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

3. Invia simple_file_job alla coda. Quando ti viene richiesto di confermare il caricamento, inserisci. **y**

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

4. Conferma che l'SMF funzioni correttamente.

```
deadline fleet get
```

   - L'operatore potrebbe impiegare alcuni minuti per iniziare. Ripeti il `deadline fleet get` comando finché non vedi che la flotta è in funzione.
   - La flotta `queueFleetAssociationsStatus` per i servizi gestiti sarà. `ACTIVE`
   - La SMF `autoScalingStatus` cambierà da a. `GROWING` `STEADY`

   Il tuo stato sarà simile al seguente:

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

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

5. Visualizza il registro del lavoro che hai inviato. Questo registro viene memorizzato in un registro in Amazon CloudWatch Logs, non nel CloudShell file system.

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

Fasi successive

Dopo aver creato e testato una flotta gestita dai servizi, dovresti rimuovere le risorse che hai creato per evitare addebiti inutili.

- Pulisci le risorse della tua azienda agricola in Deadline Cloudper chiudere le risorse che hai usato per questo tutorial.

Pulisci le risorse della tua azienda agricola in Deadline Cloud

Per sviluppare e testare nuovi carichi di lavoro e integrazioni di pipeline, puoi continuare a utilizzare la farm per sviluppatori Deadline Cloud che hai creato per questo tutorial. Se non hai più bisogno della tua farm di sviluppatori, puoi eliminarne le risorse, tra cui farm, fleet, queue, ruoli AWS Identity and Access Management (IAM) e log in Amazon CloudWatch Logs. Dopo aver eliminato queste risorse, dovrai ricominciare il tutorial per utilizzarle. Per ulteriori informazioni, consulta Guida introduttiva alle risorse di Deadline Cloud.

Per ripulire le risorse della farm degli sviluppatori

1. Scegli la tua prima CloudShell scheda, quindi interrompi tutte le associazioni queue-fleet relative alla tua coda.

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

2. Elenca le associazioni delle flotte in coda.

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

Potrebbe essere necessario eseguire nuovamente il comando fino a quando l'output non riporta i risultati`"status": "STOPPED"`, quindi è possibile procedere al passaggio successivo. Il completamento di questo processo può richiedere diversi minuti.

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

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

3. Elimina tutte le associazioni queue-fleet per la tua coda.

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

4. Elimina tutte le flotte associate alla coda.

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

5. Elimina la coda.

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

6. Eliminare la fattoria.

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

7. Elimina altre AWS risorse per la tua fattoria.

a. Elimina il ruolo fleet AWS Identity and Access Management (IAM).

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

b. Elimina il ruolo IAM in coda.

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

c. Elimina i gruppi di log di Amazon CloudWatch Logs. Ogni coda e flotta ha il proprio gruppo di log.

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

Crea lavori da inviare a Deadline Cloud

Invii offerte di lavoro a Deadline Cloud utilizzando i pacchetti di lavoro. Un job bundle è una raccolta di file, tra cui un modello di lavoro Open Job Description (OpenJD) e qualsiasi file di asset necessario per eseguire il rendering del lavoro.

Il modello di lavoro descrive come i lavoratori elaborano e accedono agli asset e fornisce lo script eseguito dal lavoratore. I Job bundle consentono ad artisti, direttori tecnici e sviluppatori di pipeline di inviare facilmente lavori complessi a Deadline Cloud dalle loro workstation locali o dalla render farm locale. I Job bundle sono particolarmente utili per i team che lavorano su effetti visivi, animazioni o altri progetti di rendering multimediale su larga scala che richiedono risorse di elaborazione scalabili e su richiesta.

È possibile creare il job bundle utilizzando il file system locale per archiviare i file e un editor di testo per creare il modello di lavoro. Dopo aver creato il pacchetto, invia il lavoro a Deadline Cloud utilizzando la CLI di Deadline Cloud o uno strumento come un mittente di Deadline Cloud

Puoi archiviare le tue risorse in un file system condiviso tra i tuoi dipendenti oppure puoi utilizzare gli allegati di lavoro di Deadline Cloud per automatizzare lo spostamento delle risorse nei bucket S3, dove i dipendenti possono accedervi. Gli allegati Job aiutano anche a riportare l'output dai lavori alle workstation.

Le seguenti sezioni forniscono istruzioni dettagliate sulla creazione e l'invio di pacchetti di lavoro a Deadline Cloud.

Argomenti

- Modelli Open Job Description (OpenJD) per Deadline Cloud
- Utilizzo dei file nei lavori
- Usa gli allegati del lavoro per condividere file
- Creare limiti di risorse per i lavori
- Come inviare un'offerta di lavoro a Deadline Cloud
- Pianifica lavori in Deadline Cloud
- Modifica un lavoro in Deadline Cloud

Modelli Open Job Description (OpenJD) per Deadline Cloud

Un job bundle è uno degli strumenti che usi per definire i lavori per AWS Deadline Cloud. Raggruppano un modello di Open Job Description (OpenJD) con informazioni aggiuntive come file e directory che i lavori utilizzano con gli allegati del lavoro. Utilizzi l'interfaccia a riga di comando (CLI) di Deadline Cloud per utilizzare un pacchetto di lavori per inviare lavori per l'esecuzione di una coda.

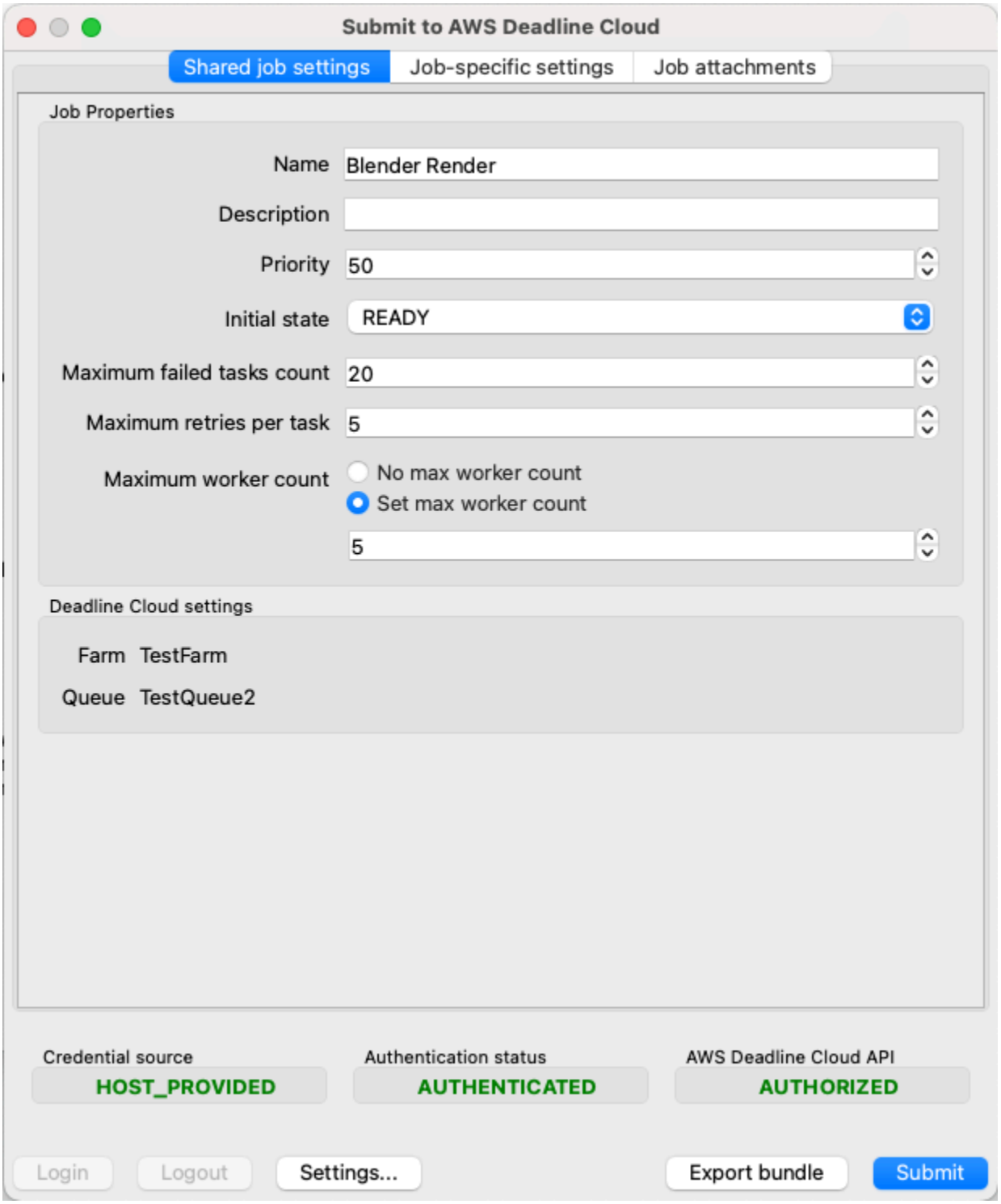
Un job bundle è una struttura di directory che contiene un modello di lavoro OpenJD, altri file che definiscono il lavoro e file specifici del lavoro richiesti come input per il lavoro. È possibile specificare i file che definiscono il lavoro come file YAML o JSON.

L'unico file richiesto è o. `template.yaml` `template.json` Puoi anche includere i seguenti file:

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

Utilizza un pacchetto di lavori per l'invio di lavori personalizzati con la CLI di Deadline Cloud e un allegato di lavoro, oppure puoi utilizzare un'interfaccia grafica di invio. Ad esempio, quello che segue è l'esempio di Blender di. GitHub Per eseguire l'esempio, utilizzate il seguente comando nella directory di esempio di Blender:

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



Il pannello delle impostazioni specifiche del lavoro viene generato dalle `userInterface` proprietà dei parametri del lavoro definiti nel modello di lavoro.

Per inviare un lavoro utilizzando la riga di comando, è possibile utilizzare un comando simile al seguente

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

Oppure puoi usare la `deadline.client.api.create_job_from_job_bundle` funzione nel pacchetto `deadline` Python.

Tutti i plugin per l'invio dei lavori forniti con Deadline Cloud, come il plug-in Autodesk Maya, generano un pacchetto di lavori per l'invio e quindi utilizzano il pacchetto Deadline Cloud Python per inviare il lavoro a Deadline Cloud. Puoi vedere i pacchetti di offerte di lavoro inviati nella directory della cronologia dei lavori della tua postazione di lavoro o utilizzando un mittente. È possibile trovare la directory della cronologia delle mansioni con il seguente comando:

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

Quando il tuo lavoro è in esecuzione su un lavoratore Deadline Cloud, ha accesso a variabili di ambiente che forniscono informazioni sul lavoro. Le variabili di ambiente sono:

Nome della variabile	Disponibilità
DEADLINE_FARM_ID	Tutte le operazioni
DEADLINE_FLOTTE_ID	Tutte le operazioni
ID_DEADLINE_WORKER	Tutte le operazioni
ID_CODA DI SCADENZA	Tutte le operazioni
ID_DEADLINE_JOB_ID	Tutte le operazioni
ID_FASE DI SCADENZA	Azioni relative alle attività
DEADLINE_SESSION_ID	Tutte le operazioni
ID_ATTIVITÀ_SCADENZA	Azioni relative alle attività
DEADLINE_SESSIONACTION_ID	Tutte le operazioni

Argomenti

- Elementi del modello di lavoro per i pacchetti di lavoro
- Suddivisione delle attività per i modelli di lavoro
- Valori dei parametri, elementi per i job bundle.
- Elementi di riferimento delle risorse per i pacchetti di lavoro

Elementi del modello di lavoro per i pacchetti di lavoro

Il modello di lavoro definisce l'ambiente di runtime e i processi che vengono eseguiti come parte di un job di Deadline Cloud. È possibile creare parametri in un modello in modo che possa essere utilizzato per creare lavori che differiscono solo nei valori di input, proprio come una funzione in un linguaggio di programmazione.

Quando invii un lavoro a Deadline Cloud, questo viene eseguito in qualsiasi ambiente di coda applicato alla coda. Gli ambienti di coda vengono creati utilizzando la specifica degli ambienti esterni Open Job Description (OpenJD). Per i dettagli, consulta il modello Environment nel repository OpenJD. GitHub

Per un'introduzione alla creazione di un lavoro con un modello di lavoro OpenJD, vedi Introduzione alla creazione di un lavoro nel repository OpenJD. GitHub Ulteriori informazioni sono disponibili in Come vengono eseguiti i lavori. Sono disponibili esempi di modelli di lavoro nella directory del GitHub repository OpenJD. `samples`

È possibile definire il modello di lavoro in formato YAML (`template.yaml`) o in formato JSON (). `template.json` Gli esempi in questa sezione sono mostrati in formato YAML.

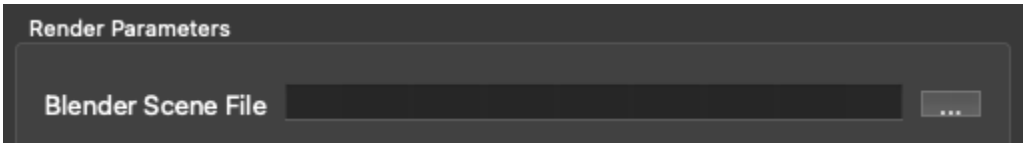
Ad esempio, il modello di lavoro per l'`blender_render`esempio definisce un parametro di input `BlenderSceneFile` come percorso di file:

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
```

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

La `userInterface` proprietà definisce il comportamento delle interfacce utente generate automaticamente sia per la riga di comando che utilizza il `deadline bundle gui-submit` comando sia all'interno dei plugin di invio dei lavori per applicazioni come Autodesk Maya.

In questo esempio, il widget dell'interfaccia utente per l'immissione di un valore per il `BlenderSceneFile` parametro è una finestra di dialogo per la selezione dei file che mostra solo i file. `.blend`



Per ulteriori esempi di utilizzo dell'`userIntefaceelemento`, consultate l'esempio gui_control_showcase nel repository su. deadline-cloud-samples GitHub

Le `dataFlow` proprietà `objectType` and controllano il comportamento degli allegati delle offerte di lavoro quando invii un'offerta di lavoro da un pacchetto di offerte di lavoro. In questo caso, `objectType: FILE dataFlow:IN` significa che il valore di `BlenderSceneFile` è un file di input per gli allegati del lavoro.

Al contrario, la definizione del `OutputDir` parametro ha `objectType: DIRECTORY` e`dataFlow:` OUT:

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
    label: Output Directory
    groupLabel: Render Parameters
  default: "./output"
  description: Choose the render output directory.
```

Il valore del `OutputDir` parametro viene utilizzato dagli allegati del lavoro come directory in cui il lavoro scrive i file di output.

Per ulteriori informazioni sulle `dataFlow` proprietà `objectType` e, vedere JobPathParameterDefinitionla specifica Open Job Description

Il resto dell'esempio di modello di `blender_render` lavoro definisce il flusso di lavoro del lavoro come un singolo passaggio, con ogni fotogramma dell'animazione renderizzato come un'attività separata:

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

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

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

Ad esempio, se il valore del `Frames` parametro è`1-10`, definisce 10 attività. Ogni task ha un valore diverso per il `Frame` parametro. Per eseguire un'attività:

1. Ad esempio, tutti i riferimenti alle variabili nella `data` proprietà del file incorporato vengono espansi`--render-frame 1`.
2. Il contenuto della `data` proprietà viene scritto su un file nella directory di lavoro della sessione su disco.
3. Il onRun comando dell'attività si risolve in `bash` *`location of embedded file`* e quindi viene eseguito.

Per ulteriori informazioni su file incorporati, sessioni e percorsi mappati, vedere How job are run nella specifica Open Job Description.

Esistono altri esempi di modelli di lavoro nel repository deadline-cloud-samples/job_bundles, oltre agli esempi di modelli forniti con la specifica Open Job Descriptions.

Suddivisione delle attività per i modelli di lavoro

La suddivisione in blocchi delle attività consente di raggruppare più attività in un'unica unità di lavoro denominata chunk. In un lavoro di rendering, ad esempio, ciò significa che Deadline Cloud può inviare più frame insieme anziché un frame per chiamata di comando. Ciò riduce il sovraccarico di avvio delle applicazioni per ogni attività e riduce la durata totale del lavoro. Per i dettagli, consulta Esecuzione di più frame alla volta nel wiki di OpenJD.

OpenJD supporta estensioni che aggiungono funzionalità opzionali ai modelli di lavoro. La suddivisione in blocchi delle attività viene abilitata aggiungendo l'estensione. `TASK_CHUNKING` Per utilizzare la suddivisione in blocchi, aggiungi l'estensione al tuo modello di lavoro e utilizza il tipo di parametro dell'`CHUNK[INT]`attività. Invia lavori suddivisi in blocchi utilizzando lo stesso comando. `deadline bundle submit` Ad esempio, il seguente modello di lavoro esegue il rendering dei frame in blocchi di 10:

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

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

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

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

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

In questo esempio, Deadline Cloud divide i 100 frame in blocchi come, e così via. `1-10` `11-20` La `{{Task.Param.Frame}}` variabile si espande in un'espressione di intervallo come. `1-10` Poiché

`rangeConstraint` è impostato su`CONTIGUOUS`, l'intervallo è sempre in `start-end` formato. Lo script analizza questo intervallo e passa i frame iniziale e finale a Blender usando le `-e` opzioni `-s` e with. `--render-anim`

La `chunks` proprietà supporta i seguenti campi:

- `defaultTaskCount`— (Obbligatorio) Quante attività combinare in un unico blocco. Il valore massimo è 150.
- `rangeConstraint`— (Obbligatorio) Se`CONTIGUOUS`, un blocco è sempre un intervallo contiguo come. `1-10` Se`NONCONTIGUOUS`, un blocco può essere un insieme arbitrario, ad esempio. `1,3,7-10`
- `targetRuntimeSeconds`— (Facoltativo) Il tempo di esecuzione previsto in secondi per ogni blocco. Deadline Cloud può regolare dinamicamente la dimensione dei blocchi per avvicinarsi a questo obiettivo una volta completati alcuni blocchi.

Per altri esempi di suddivisione in blocchi delle attività, inclusi esempi di base e di Blender con blocchi contigui e non contigui, consulta gli esempi di suddivisione in blocchi delle attività nell'archivio degli esempi di Deadline Cloud su. GitHub

> **Requisiti della flotta gestita dal cliente**
>
> La suddivisione delle attività richiede una versione di Worker Agent compatibile. Se utilizzi flotte gestite dai clienti, assicurati che i tuoi worker agent siano aggiornati prima di inviare lavori con chunking. Le flotte gestite dai servizi utilizzano sempre una versione di Worker Agent compatibile.

> **Scaricamento dell'output per lavori suddivisi in blocchi**
>
> Quando scarichi l'output per una singola attività in un lavoro suddiviso in blocchi, Deadline Cloud scarica l'output per l'intero blocco. Ad esempio, se i frame 1-10 sono stati elaborati insieme, il download dell'output per il frame 3 include tutti i frame 1-10. Questa funzionalità richiede la `deadline-cloud` versione 0.53.3 o successiva.

Valori dei parametri, elementi per i job bundle.

È possibile utilizzare il file dei parametri per impostare i valori di alcuni parametri del processo nel modello di lavoro o gli argomenti della richiesta di CreateJoboperazione nel job bundle in modo da non dover impostare valori quando si invia un lavoro. L'interfaccia utente per l'invio dei lavori consente di modificare questi valori.

È possibile definire il modello di lavoro in formato YAML (`parameter_values.yaml`) o in formato JSON (). `parameter_values.json` Gli esempi in questa sezione sono mostrati in formato YAML.

In YAML, il formato del file è:

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

Ogni elemento dell'`parameterValues`elenco deve essere uno dei seguenti:

- Un parametro di lavoro definito nel modello di lavoro.
- Un parametro di lavoro definito in un ambiente di coda per la coda a cui si invia il lavoro..
- Un parametro speciale passato all'`CreateJob`operazione durante la creazione di un lavoro.
  - `deadline:priority`— Il valore deve essere un numero intero. Viene passato all'`CreateJob`operazione come parametro prioritario.
  - `deadline:targetTaskRunStatus`— Il valore deve essere una stringa. Viene passato all'`CreateJob`operazione come parametro targetTaskRunStatus.
  - `deadline:maxFailedTasksCount`— Il valore deve essere un numero intero. Viene passato all'`CreateJob`operazione come parametro maxFailedTasksCount.
  - `deadline:maxRetriesPerTask`— Il valore deve essere un numero intero. Viene passato all'`CreateJob`operazione come parametro maxRetriesPerTask.
  - `deadline:maxWorkercount`— Il valore deve essere un numero intero. Viene passato all'`CreateJob`operazione come maxWorkerCountparametro.

Un modello di lavoro è sempre un modello anziché un processo specifico da eseguire. Un file di valori dei parametri consente a un job bundle di fungere da modello se alcuni parametri non hanno valori definiti in questo file o come invio di job specifico se tutti i parametri hanno valori.

Ad esempio, l'esempio blender_render non ha un file di parametri e il relativo modello di lavoro definisce parametri senza valori predefiniti. Questo modello deve essere usato come modello per creare lavori. Dopo aver creato un job utilizzando questo job bundle, Deadline Cloud scrive un nuovo job bundle nella directory della cronologia dei lavori.

Ad esempio, quando invii un lavoro con il seguente comando:

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

Il nuovo job bundle contiene un `parameter_values.yaml` file che contiene i parametri specificati:

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

È possibile creare lo stesso lavoro con il seguente comando:

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

> **Note**
>
> Il job bundle inviato viene salvato nella directory della cronologia dei lavori. Puoi trovare la posizione di quella directory con il seguente comando:
>
> ```
> deadline config get settings.job_history_dir
> ```

Elementi di riferimento delle risorse per i pacchetti di lavoro

Puoi utilizzare gli allegati di lavoro di Deadline Cloud per trasferire file avanti e indietro tra la tua workstation e Deadline Cloud. Il file di riferimento delle risorse elenca i file e le directory di input, nonché le directory di output per gli allegati. Se non elencate tutti i file e le directory in questo file, potete selezionarli quando inviate un lavoro con il comando. `deadline bundle gui-submit`

Questo file non ha effetto se non si utilizzano gli allegati del lavoro.

È possibile definire il modello di lavoro in formato YAML (`asset_references.yaml`) o in formato JSON (). `asset_references.json` Gli esempi in questa sezione sono mostrati in formato YAML.

In YAML, il formato del file è:

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

    outputs:
        # Directories on the submitting workstation where the job writes output files
        # if running locally.
```

```
        directories:
        - list of directory paths

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

Quando si seleziona il file di input o output da caricare su Amazon S3, Deadline Cloud confronta il percorso del file con i percorsi elencati nei profili di archiviazione. Ogni posizione del file system `SHARED` di tipo in un profilo di archiviazione astrae una condivisione di file di rete montata sulle workstation e sugli host di lavoro. Deadline Cloud carica solo i file che non si trovano su una di queste condivisioni di file.

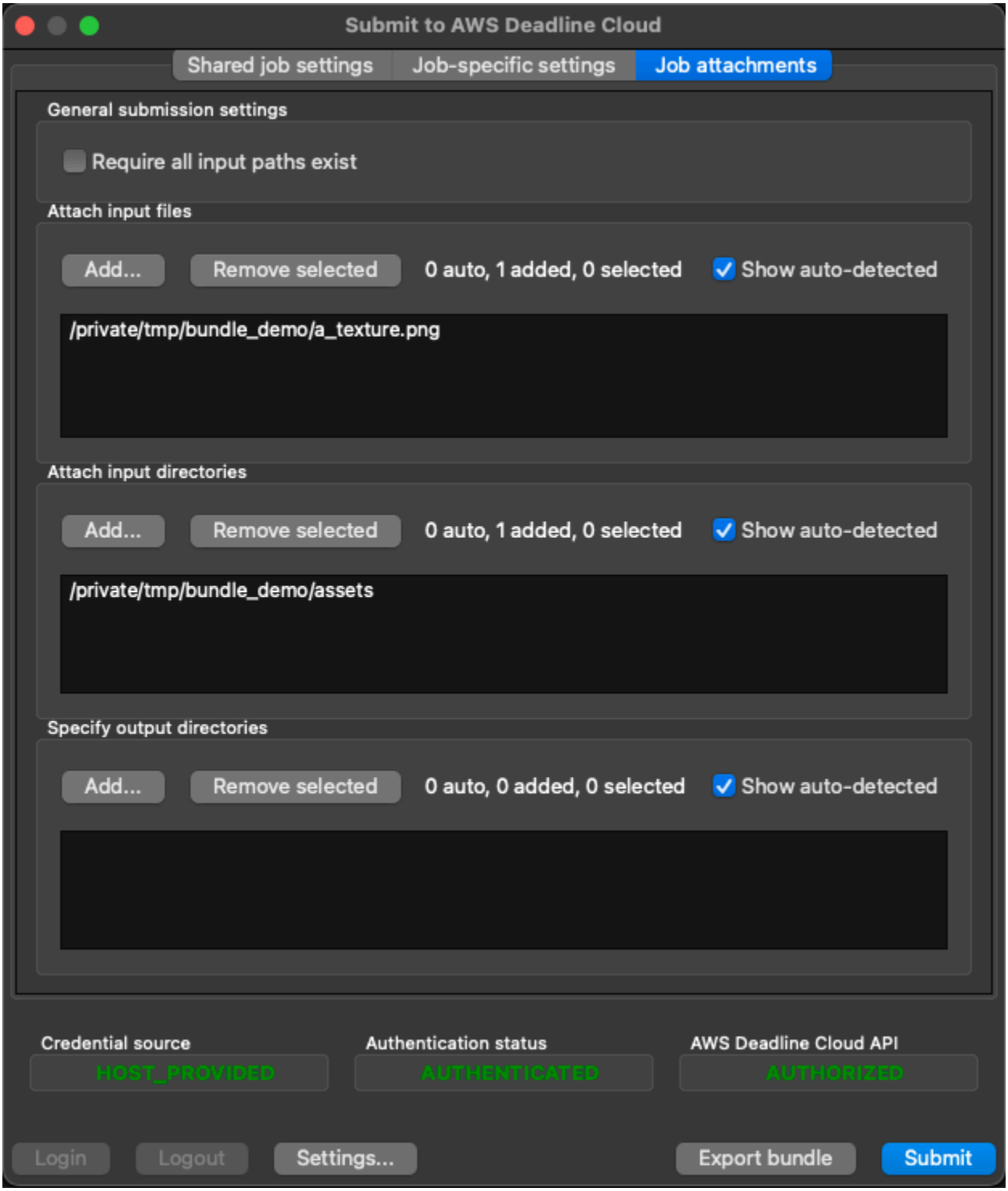
Per ulteriori informazioni sulla creazione e l'utilizzo dei profili di archiviazione, consulta Archiviazione condivisa in Deadline Cloud nella Guida per l'utente di AWS Deadline Cloud.

Example- Il file di riferimento delle risorse creato dalla GUI di Deadline Cloud

Utilizzate il comando seguente per inviare un lavoro utilizzando l'esempio blender_render.

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

Aggiungi alcuni file aggiuntivi al lavoro nella scheda Job attachments:



Dopo aver inviato il lavoro, puoi guardare il `asset_references.yaml` file nel job bundle nella directory della cronologia dei lavori per vedere le risorse nel file YAML:

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

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

Utilizzo dei file nei lavori

Molti dei lavori che invii a AWS Deadline Cloud contengono file di input e output. I file di input e le directory di output possono trovarsi su una combinazione di file system condivisi e unità locali. I lavori devono localizzare il contenuto in tali posizioni. Deadline Cloud offre due funzionalità, gli allegati dei lavori e i profili di archiviazione che interagiscono per aiutare le aziende a individuare i file di cui hanno bisogno.

Gli allegati Job offrono diversi vantaggi

- Spostamento di file tra host utilizzando Amazon S3
- Trasferisci file dalla tua postazione di lavoro agli host di lavoro e viceversa
- Disponibile per i lavori in coda in cui è abilitata la funzione
- Utilizzato principalmente con flotte gestite dai servizi, ma anche compatibile con flotte gestite dai clienti.

Utilizza i profili di storage per mappare il layout delle posizioni condivise dei file system sulla workstation e sugli host dei lavoratori. Questa mappatura aiuta i job a localizzare file e directory condivisi quando le loro posizioni differiscono tra la workstation e gli host di lavoro, ad esempio configurazioni multipiattaforma con workstation basate e worker host basati. Windows Linux La mappa del profilo di storage della configurazione del file system viene utilizzata anche dagli allegati dei lavori per identificare i file necessari per lo spostamento tra gli host tramite Amazon S3.

Se non utilizzi gli allegati di lavoro e non hai bisogno di rimappare le posizioni di file e directory tra le workstation e gli host di lavoro, non è necessario modellare le condivisioni di file con profili di storage.

Argomenti

- Esempio di infrastruttura di progetto

- Profili di archiviazione e mappatura dei percorsi

Esempio di infrastruttura di progetto

Per dimostrare l'utilizzo degli allegati di lavoro e dei profili di archiviazione, configura un ambiente di test con due progetti separati. Puoi utilizzare la console Deadline Cloud per creare le risorse di test.

1. Se non l'hai già fatto, crea una test farm. Per creare una fattoria, segui la procedura descritta in Creare una fattoria.
2. Crea due code per i lavori in ciascuno dei due progetti. Per creare code, segui la procedura riportata in Creare una coda.

   a. Crea la prima coda chiamata. **`Q1`** Usa la seguente configurazione, usa i valori predefiniti per tutti gli altri elementi.

      - Per gli allegati dei lavori, scegli Crea un nuovo bucket Amazon S3.
      - Seleziona Abilita l'associazione con flotte gestite dal cliente.
      - Per eseguire come utente, inserisci sia l'utente che **`jobuser`** il gruppo POSIX.
      - Per il ruolo del servizio di coda, create un nuovo ruolo denominato **`AssetDemoFarm-Q1-Role`**
      - Deseleziona la casella di controllo predefinita dell'ambiente di coda conda.

   b. Crea la seconda coda chiamata. **`Q2`** Usa la seguente configurazione, usa i valori predefiniti per tutti gli altri elementi.

      - Per gli allegati dei lavori, scegli Crea un nuovo bucket Amazon S3.
      - Seleziona Abilita l'associazione con flotte gestite dal cliente.
      - Per eseguire come utente, inserisci sia l'utente che **`jobuser`** il gruppo POSIX.
      - Per il ruolo del servizio di coda, create un nuovo ruolo denominato **`AssetDemoFarm-Q2-Role`**
      - Deseleziona la casella di controllo predefinita dell'ambiente di coda conda.

3. Crea un'unica flotta gestita dal cliente che esegua i lavori da entrambe le code. Per creare il parco veicoli, segui la procedura riportata in Creare un parco veicoli gestito dal cliente. Utilizza la seguente configurazione:

   - Per Nome, usa**`DemoFleet`**.

- Per il tipo di flotta scegli Customer managed
- Per il ruolo Fleet Service, crea un nuovo ruolo denominato AssetDemoFarm-Fleet-Role.
- Non associate la flotta a nessuna coda.

L'ambiente di test presuppone che vi siano tre file system condivisi tra host che utilizzano condivisioni di file di rete. In questo esempio, le posizioni hanno i seguenti nomi:

- `FSCommon`- contiene risorse lavorative di input comuni a entrambi i progetti.
- `FS1`- contiene risorse lavorative di input e output per il progetto 1.
- `FS2`- contiene risorse lavorative di input e output per il progetto 2.

L'ambiente di test presuppone inoltre che vi siano tre workstation, come segue:

- `WSAll`- Una workstation Linux basata sugli sviluppatori utilizzata dagli sviluppatori per tutti i progetti. Le posizioni dei file system condivisi sono:
  - `FSCommon`: `/shared/common`
  - `FS1`: `/shared/projects/project1`
  - `FS2`: `/shared/projects/project2`
- `WS1`- Una workstation Windows basata sul progetto 1. Le posizioni dei file system condivisi sono:
  - `FSCommon`: `S:\`
  - `FS1`: `Z:\`
  - `FS2`: non disponibile
- `WS1`- Una workstation macOS basata sul progetto 2. Le posizioni condivise del file system sono:
  - `FSCommon`: `/Volumes/common`
  - `FS1`: Non disponibile
  - `FS2`: `/Volumes/projects/project2`

Infine, definisci le posizioni dei file system condivise per i lavoratori della tua flotta. Gli esempi che seguono si riferiscono a questa configurazione come`WorkerConfig`. Le posizioni condivise sono:

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

Non è necessario configurare file system, workstation o worker condivisi che corrispondano a questa configurazione. Non è necessario che le posizioni condivise esistano per la dimostrazione.

Profili di archiviazione e mappatura dei percorsi

Utilizza i profili di storage per modellare i file system sulla workstation e sugli host dei lavoratori. Ogni profilo di storage descrive il sistema operativo e il layout del file system di una delle configurazioni di sistema. Questo argomento descrive come utilizzare i profili di archiviazione per modellare le configurazioni del file system degli host in modo che Deadline Cloud possa generare regole di mappatura dei percorsi per i tuoi lavori e come tali regole di mappatura dei percorsi vengono generate dai tuoi profili di archiviazione.

Quando invii un lavoro a Deadline Cloud, puoi fornire un ID del profilo di archiviazione opzionale per il lavoro. Questo profilo di archiviazione descrive il file system della workstation di invio. Descrive la configurazione originale del file system utilizzata dai percorsi dei file nel modello di lavoro.

È inoltre possibile associare un profilo di archiviazione a una flotta. Il profilo di storage descrive la configurazione del file system di tutti gli host di lavoro del parco macchine. Se si dispone di lavoratori con una configurazione del file system diversa, tali lavoratori devono essere assegnati a una flotta diversa nella fattoria.

Le regole di mappatura dei percorsi descrivono come i percorsi devono essere rimappati dal modo in cui sono specificati nel lavoro alla posizione effettiva del percorso su un host di lavoro. Deadline Cloud confronta la configurazione del file system descritta nel profilo di archiviazione di un lavoro con il profilo di archiviazione del parco macchine che esegue il lavoro per ricavare queste regole di mappatura dei percorsi.

Argomenti

- Modella le posizioni dei file system condivise con profili di archiviazione
- Configura i profili di archiviazione per le flotte
- Configura i profili di archiviazione per le code
- Deriva le regole di mappatura dei percorsi dai profili di archiviazione

Modella le posizioni dei file system condivise con profili di archiviazione

Un profilo di storage modella la configurazione del file system di una delle configurazioni host. Nell'infrastruttura del progetto di esempio sono disponibili quattro diverse configurazioni host. In

questo esempio si crea un profilo di archiviazione separato per ciascuna di esse. È possibile creare un profilo di archiviazione utilizzando uno dei seguenti metodi:

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

Un profilo di archiviazione è costituito da un elenco di posizioni del file system, ciascuna delle quali indica a Deadline Cloud la posizione e il tipo di posizione del file system rilevante per i lavori inviati o eseguiti su un host. Un profilo di archiviazione deve modellare solo le posizioni pertinenti per i lavori. Ad esempio, la `FSCommon` posizione condivisa si trova sulla workstation `WS1` pressoS`:\`, quindi la posizione del file system corrispondente è:

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

Utilizzate i seguenti comandi per creare il profilo di archiviazione per le configurazioni `WS1` della workstation `WS3` e la configurazione del worker `WorkerConfig` utilizzando in: `WS2` AWS CLIAWS CloudShell

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

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

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS1 \
  --os-family WINDOWS \
  --file-system-locations \
```

```
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
      {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
   ]'

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

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

> **Note**
>
> È necessario fare riferimento alle posizioni del file system nei profili di archiviazione utilizzando gli stessi valori per la `name` proprietà in tutti i profili di archiviazione della farm. Deadline Cloud confronta i nomi per determinare che le posizioni dei file system di diversi profili di archiviazione si riferiscono alla stessa posizione durante la generazione delle regole di mappatura dei percorsi.

Configura i profili di archiviazione per le flotte

È possibile configurare una flotta per includere un profilo di archiviazione che modelli le posizioni dei file system su tutti i lavoratori del parco macchine. La configurazione del file system host di tutti i lavoratori di una flotta deve corrispondere al profilo di storage della flotta. I lavoratori con configurazioni di file system diverse devono appartenere a flotte separate.

Per impostare la configurazione della flotta in modo da utilizzare il profilo `WorkerConfig` di archiviazione, utilizza in: AWS CLIAWS CloudShell

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

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

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
  "{
    \"customerManaged\": {
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

Configura i profili di archiviazione per le code

La configurazione di una coda include un elenco di nomi con distinzione tra maiuscole e minuscole delle posizioni condivise del file system a cui i lavori inviati alla coda richiedono l'accesso. Ad esempio, i lavori inviati alla coda `Q1` richiedono posizioni del file system e. `FSCommon FS1` I lavori inviati alla coda `Q2` richiedono posizioni del file system e. `FSCommon FS2`

Per impostare le configurazioni della coda in modo che richiedano queste posizioni del file system, utilizzate lo script seguente:

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

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

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

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

La configurazione di una coda include anche un elenco di profili di archiviazione consentiti che si applica ai lavori inviati e alle flotte associate a tale coda. Nell'elenco dei profili di archiviazione consentiti della coda sono consentiti solo i profili di archiviazione che definiscono le posizioni del file system per tutte le posizioni del file system richieste per la coda.

Un processo ha esito negativo se lo si invia con un profilo di archiviazione non presente nell'elenco dei profili di archiviazione consentiti per la coda. Puoi sempre inviare un lavoro senza profilo di archiviazione a una coda. Le configurazioni delle workstation sono etichettate `WSAll` ed `WS1` entrambe hanno le posizioni del file system richieste (`FSCommoneFS1`) per la coda. `Q1` Devono essere autorizzati a inviare lavori alla coda. Allo stesso modo, le configurazioni delle workstation `WS2` soddisfano `WSAll` i requisiti per la coda. `Q2` Devono essere autorizzati a inviare lavori a quella coda. Aggiorna entrambe le configurazioni di coda per consentire l'invio dei lavori con questi profili di archiviazione utilizzando lo script seguente:

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

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

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

Se si aggiunge il profilo `WS2` di archiviazione all'elenco dei profili di archiviazione consentiti per la coda`Q1`, l'operazione fallisce:

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

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

Questo perché il profilo `WS2` di archiviazione non contiene una definizione per la posizione del file system denominata richiesta dalla `FS1` coda`Q1`.

Anche l'associazione di una flotta configurata con un profilo di archiviazione non presente nell'elenco dei profili di archiviazione consentiti della coda non riesce. Esempio:

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

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

Per correggere l'errore, aggiungi il profilo di archiviazione denominato `WorkerConfig` all'elenco dei profili di archiviazione consentiti sia per la coda che per la coda`Q1`. `Q2` Quindi, associa la flotta a queste code in modo che i lavoratori del parco macchine possano eseguire i lavori da entrambe le code.

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

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

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

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

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

```
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

Deriva le regole di mappatura dei percorsi dai profili di archiviazione

Le regole di mappatura dei percorsi descrivono come devono essere rimappati i percorsi dal lavoro alla posizione effettiva del percorso su un host di lavoro. Quando un'attività è in esecuzione su un lavoratore, il profilo di archiviazione del lavoro viene confrontato con il profilo di archiviazione del parco macchine del lavoratore per ricavare le regole di mappatura dei percorsi per l'attività.

Deadline Cloud crea una regola di mappatura per ciascuna delle posizioni del file system richieste nella configurazione della coda. Ad esempio, un lavoro inviato con il profilo di `WSAll` archiviazione da mettere in coda `Q1` ha le seguenti regole di mappatura dei percorsi:

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

Deadline Cloud crea regole per le posizioni del `FS1` file system `FSComm` and, ma non per la posizione del `FS2` file system, anche se definite sia dal profilo che dal `WSAll` profilo `WorkerConfig` di archiviazione. `FS2` Questo perché l'elenco delle posizioni richieste `Q1` del file system di queue è. `["FSComm", "FS1"]`

È possibile confermare le regole di mappatura dei percorsi disponibili per i lavori inviati con un particolare profilo di archiviazione inviando un lavoro che stampi il file delle regole di mappatura dei percorsi di Open Job Description e quindi leggendo il registro della sessione dopo il completamento del lavoro:

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

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
```

```
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
            "onRun": {
              "command": "/bin/cat",
              "args": [ "{{Session.PathMappingRulesFile}}" ]
            }
          }
        }
      }
    ]
  }'
```

Se utilizzi la CLI di Deadline Cloud per inviare lavori, la relativa impostazione di `settings.storage_profile_id` configurazione imposta il profilo di archiviazione che avranno i lavori inviati con la CLI. Per inviare lavori con il profilo di `WSAll` archiviazione, imposta:

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

Per gestire un worker gestito dal cliente come se fosse in esecuzione nell'infrastruttura di esempio, segui la procedura descritta in Esegui il worker agent nella Deadline Cloud User Guide to run a worker with. AWS CloudShell Se hai già seguito queste istruzioni, elimina prima le directory `~/demoenv-logs` and`~/demoenv-persist`. Inoltre, prima di procedere, impostate i valori delle variabili `DEV_FARM_ID` e di `DEV_CMF_ID` ambiente a cui fanno riferimento le direzioni come segue:

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

Dopo l'esecuzione del processo, puoi visualizzare le regole di mappatura dei percorsi nel file di registro del processo:

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

Il registro contiene la mappatura sia per il file system che per quello `FS1` dei `FSComm` file. Riformattata per motivi di leggibilità, la voce di registro ha il seguente aspetto:

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

Puoi inviare lavori con diversi profili di archiviazione per vedere come cambiano le regole di mappatura dei percorsi.

Usa gli allegati del lavoro per condividere file

Usa gli allegati di lavoro per rendere disponibili per i tuoi lavori i file che non si trovano nelle directory condivise e per acquisire i file di output se non vengono scritti in directory condivise. Job attachments utilizza Amazon S3 per trasferire i file tra host. I file vengono archiviati in bucket S3 e non è necessario caricare un file se il suo contenuto non è cambiato.

È necessario utilizzare gli allegati dei lavori quando si eseguono lavori su flotte gestite dai servizi, poiché gli host non condividono le posizioni dei file system. Gli allegati di lavoro sono utili anche con le flotte gestite dal cliente quando i file di input o output di un lavoro sono archiviati su un file system di rete condiviso, ad esempio quando il job bundle contiene script shell o Python.

Quando invii un pacchetto di lavoro con la CLI di Deadline Cloud o un mittente di Deadline Cloud, gli allegati di lavoro utilizzano il profilo di archiviazione del lavoro e le posizioni del file system richieste della coda per identificare i file di input che non si trovano su un host di lavoro e devono essere caricati su Amazon S3 come parte dell'invio del lavoro. Questi profili di archiviazione aiutano anche Deadline Cloud a identificare i file di output nelle postazioni host dei lavoratori che devono essere caricati su Amazon S3 in modo che siano disponibili sulla tua workstation.

Gli esempi di job attachments utilizzano le configurazioni della farm, della flotta, delle code e dei profili di archiviazione di e. Esempio di infrastruttura di progetto Profili di archiviazione e mappatura dei percorsi È necessario esaminare queste sezioni prima di questa.

Negli esempi seguenti, si utilizza un job bundle di esempio come punto di partenza, quindi lo si modifica per esplorare le funzionalità di Job Attachment. I Job bundle sono il modo migliore per i tuoi lavori di utilizzare gli allegati di lavoro. Combinano un modello di lavoro Open Job Description in una directory con file aggiuntivi che elencano i file e le directory richiesti dai lavori che utilizzano il job bundle. Per ulteriori informazioni sui pacchetti di lavoro, vedere. Modelli Open Job Description (OpenJD) per Deadline Cloud

Invio di file con un lavoro

Con Deadline Cloud, puoi consentire ai flussi di lavoro di accedere ai file di input che non sono disponibili in posizioni di file system condivise sugli host dei lavoratori. I Job attachments consentono ai processi di rendering di accedere ai file che risiedono solo su un'unità di lavoro locale o su un ambiente di flotta gestito dai servizi. Quando si invia un pacchetto di lavori, è possibile includere elenchi di file di input e directory richiesti dal lavoro. Deadline Cloud identifica questi file non condivisi, li carica dal computer locale su Amazon S3 e li scarica sull'host di lavoro. Semplifica il processo di trasferimento delle risorse di input ai nodi di rendering, garantendo che tutti i file richiesti siano accessibili per l'esecuzione distribuita del lavoro.

È possibile specificare i file per i lavori direttamente nel job bundle, utilizzare i parametri nel modello di lavoro fornito utilizzando variabili di ambiente o uno script e utilizzare il file del lavoro. `assets_references` È possibile utilizzare uno di questi metodi o una combinazione di tutti e tre. È possibile specificare un profilo di archiviazione per il pacchetto per il lavoro in modo che carichi solo i file che sono stati modificati sulla workstation locale.

Questa sezione utilizza un esempio di job bundle GitHub per dimostrare in che modo Deadline Cloud identifica i file del job da caricare, come tali file sono organizzati in Amazon S3 e come vengono messi a disposizione dei worker host che elaborano i lavori.

Argomenti

- In che modo Deadline Cloud carica i file su Amazon S3
- In che modo Deadline Cloud sceglie i file da caricare
- In che modo le offerte di lavoro trovano i file di input allegati

In che modo Deadline Cloud carica i file su Amazon S3

Questo esempio mostra come Deadline Cloud carica i file dalla tua workstation o dal tuo host di lavoro su Amazon S3 in modo che possano essere condivisi. Utilizza un pacchetto di lavori di esempio GitHub e la CLI di Deadline Cloud per inviare lavori.

Inizia clonando l' GitHubarchivio di esempi di Deadline Cloud nel tuo AWS CloudShellambiente, quindi copia il `job_attachments_devguide` job bundle nella tua home directory:

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

Installa la CLI di Deadline Cloud per inviare pacchetti di lavoro:

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

Il `job_attachments_devguide` job bundle prevede un unico passaggio con un'attività che esegue uno script di shell bash la cui posizione del file system viene passata come parametro di lavoro. La definizione del parametro job è:

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

Il `IN` valore della `dataFlow` proprietà indica agli allegati del lavoro che il valore del `ScriptFile` parametro è un input per il lavoro. Il valore della `default` proprietà è una posizione relativa alla directory del job bundle, ma può anche essere un percorso assoluto. Questa definizione di parametro dichiara il `script.sh` file nella directory del job bundle come file di input necessario per l'esecuzione del processo.

Quindi, assicurati che la CLI di Deadline Cloud non abbia un profilo di archiviazione configurato, quindi invia il lavoro alla coda: `Q1`

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

```
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''

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

L'output della CLI di Deadline Cloud dopo l'esecuzione di questo comando è simile a:

```
Submitting to Queue: Q1
...
Hashing Attachments  [####################################]  100%
Hashing Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.0327 seconds at 1.19 KB/s.

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

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

Quando invii il lavoro, Deadline Cloud esegue prima l'hash del `script.sh` file e poi lo carica su Amazon S3.

Deadline Cloud considera il bucket S3 come un archivio indirizzabile ai contenuti. I file vengono caricati su oggetti S3. Il nome dell'oggetto deriva da un hash del contenuto del file. Se due file hanno contenuti identici, hanno lo stesso valore hash indipendentemente da dove si trovano i file o dal loro nome. Questa archiviazione indirizzabile ai contenuti consente a Deadline Cloud di evitare di caricare un file se è già disponibile.

Puoi usare l'AWS CLI per vedere gli oggetti che sono stati caricati su Amazon S3:

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
```

```
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

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

Due oggetti sono stati caricati su S3:

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— Il contenuto di. `script.sh` Il valore 87cb19095dd5d78fcaf56384ef0e6241 nella chiave dell'oggetto è l'hash del contenuto del file e l'estensione `xxh128` indica che il valore hash è stato calcolato come xxhash a 128 bit.
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— L'oggetto manifesto per l'invio del lavoro. I valori `<farm-id><queue-id>`, e `<guid>` sono l'identificatore della fattoria, l'identificatore di coda e un valore esadecimale casuale. Il valore `a1d221c7fd97b08175b3872a37428e8c` in questo esempio è un valore hash calcolato dalla stringa`/home/cloudshell-user/job_attachments_devguide`, la directory in cui si trova. `script.sh`

L'oggetto manifest contiene le informazioni per i file di input su un percorso root specifico caricato su S3 come parte dell'invio del lavoro. Scarica questo file manifesto ()`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`. Il suo contenuto è simile a:

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

Ciò indica che il file `script.sh` è stato caricato e l'hash del contenuto di quel file è`87cb19095dd5d78fcaf56384ef0e6241`. Questo valore hash corrisponde al valore nel nome

dell'oggetto. `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128` Viene utilizzato da Deadline Cloud per sapere quale oggetto scaricare per il contenuto di questo file.

Lo schema completo di questo file è disponibile in GitHub.

Quando si utilizza l'CreateJob operazione, è possibile impostare la posizione degli oggetti del manifesto. È possibile utilizzare l'GetJoboperazione per visualizzare la posizione:

```
{
    "attachments": {
        "file system": "COPIED",
        "manifests": [
            {
                "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
                "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/
a1d221c7fd97b08175b3872a37428e8c_input",
                "rootPath": "/home/cloudshell-user/job_attachments_devguide",
                "rootPathFormat": "posix"
            }
        ]
    },
    ...
}
```

In che modo Deadline Cloud sceglie i file da caricare

I file e le directory che Job Attachments considera per il caricamento su Amazon S3 come input per il processo sono:

- I valori di tutti i parametri `PATH` di processo di tipo definiti nel modello di lavoro del job bundle con il valore o. `dataFlow` `IN` `INOUT`
- I file e le directory elencati come input nel file di riferimenti agli asset del job bundle.

Se invii un lavoro senza profilo di archiviazione, vengono caricati tutti i file considerati per il caricamento. Se invii un lavoro con un profilo di storage, i file non vengono caricati su Amazon S3 se si trovano nelle posizioni del file system di `SHARED` tipo del profilo di storage, che sono anche posizioni del file system necessarie per la coda. Queste posizioni dovrebbero essere disponibili sugli host di lavoro che eseguono il lavoro, quindi non è necessario caricarle su S3.

In questo esempio, crei posizioni dei `SHARED` file system `WSAll` nel tuo CloudShell ambiente AWS e poi aggiungi file a tali posizioni. Utilizza il seguente comando:

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

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

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

Successivamente, aggiungi un file di riferimenti agli asset al job bundle che include tutti i file che hai creato come input per il job. Utilizza il seguente comando:

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

Quindi, configura la CLI di Deadline Cloud per inviare lavori con `WSAll` il profilo di archiviazione, quindi invia il pacchetto di lavori:

```

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

deadline config set settings.storage_profile_id $WSALL_ID

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

Deadline Cloud carica due file su Amazon S3 quando invii il lavoro. Puoi scaricare gli oggetti manifest per il lavoro da S3 per vedere i file caricati:

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

In questo esempio, c'è un singolo file manifest con i seguenti contenuti:

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

Utilizzate l'GetJob operazione per il manifesto per vedere che `rootPath` è «/».

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

Il percorso principale per un set di file di input è sempre il sottopercorso comune più lungo di tali file. Se il lavoro è stato inviato Windows invece da e ci sono file di input senza un sottopercorso comune perché si trovavano su unità diverse, viene visualizzato un percorso principale separato su ogni unità.

I percorsi in un manifesto sono sempre relativi al percorso principale del manifesto, quindi i file di input che sono stati caricati sono:

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— Il file di script nel job bundle.
- `/shared/projects/project2/file.txt`— Il file in una posizione del `SHARED` file system nel profilo `WSAll` di archiviazione che non è nell'elenco delle posizioni del file system richieste per la coda`Q1`.

I file nelle posizioni del file system `FSCommon` (`/shared/common/file.txt`) e `FS1` (`/shared/projects/project1/file.txt`) non sono presenti nell'elenco. Questo perché tali posizioni del file system si trovano `SHARED` nel profilo di `WSAll` archiviazione ed entrambe sono nell'elenco delle posizioni del file system richieste in coda`Q1`.

È possibile visualizzare le posizioni del file system prese in considerazione `SHARED` per un lavoro inviato con un particolare profilo di archiviazione con l'GetStorageProfileForQueue operazione. Per richiedere il profilo di archiviazione `WSAll` per la coda, `Q1` utilizzare il seguente comando:

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

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

In che modo le offerte di lavoro trovano i file di input allegati

Affinché un lavoro utilizzi i file che Deadline Cloud carica su Amazon S3 utilizzando gli allegati del lavoro, il tuo lavoro ha bisogno di quei file disponibili tramite il file system sugli host dei lavoratori. Quando una sessione per il tuo lavoro viene eseguita su un host di lavoro, Deadline Cloud scarica i file di input per il lavoro in una directory temporanea sull'unità locale dell'host di lavoro e aggiunge regole di mappatura dei percorsi per ciascuno dei percorsi principali del lavoro alla posizione del file system sull'unità locale.

Per questo esempio, avvia l'agente di lavoro Deadline Cloud in una CloudShell scheda AWS. Consenti a tutti i job inviati in precedenza di terminare l'esecuzione, quindi elimina i job logs dalla directory logs:

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

Lo script seguente modifica il job bundle per mostrare tutti i file nella directory di lavoro temporanea della sessione e il contenuto del file delle regole di mappatura dei percorsi, quindi invia un job con il pacchetto modificato:

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

deadline config set settings.storage_profile_id $WSALL_ID

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

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

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

```
EOF

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

È possibile visualizzare il registro dell'esecuzione del lavoro dopo che è stato eseguito dal lavoratore nel proprio ambiente: AWS CloudShell

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

Il registro mostra che la prima cosa che si verifica nella sessione è che i due file di input per il lavoro vengono scaricati sul lavoratore:

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

Il prossimo è l'output di `script.sh` run by the job:

- I file di input caricati al momento dell'invio del lavoro si trovano in una directory il cui nome inizia con «assetroot» nella directory temporanea della sessione.
- I percorsi dei file di input sono stati riposizionati rispetto alla directory «assetroot» anziché rispetto al percorso principale per l'input manifest () del lavoro. `"/"`
- Il file delle regole di mappatura dei percorsi contiene una regola aggiuntiva che si rimappa `"/"` al percorso assoluto della directory «assetroot».

Esempio:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
```

```
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }
```

> **Note**
>
> Se il lavoro inviato contiene più manifesti con percorsi root diversi, esiste una directory denominata «assetroot» diversa per ciascuno dei percorsi root.

Se è necessario fare riferimento alla posizione del file system trasferito di uno dei file di input, delle directory o delle posizioni del file system, è possibile elaborare il file delle regole di mappatura dei percorsi nel job ed eseguire la rimappatura autonomamente, oppure aggiungere un parametro PATH type job al modello di lavoro nel job bundle e passare il valore da rimappare come valore di

quel parametro. Ad esempio, l'esempio seguente modifica il job bundle in modo che abbia uno di questi parametri di job e quindi invia un job con la posizione del file system `/shared/projects/project2` come valore:

```
cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: LocationToRemap
  type: PATH
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
        - "The location of {{RawParam.LocationToRemap}} in the session is
 {{Param.LocationToRemap}}"
EOF

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

Il file di registro per l'esecuzione di questo processo contiene il relativo output:

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

Ottenere file di output da un lavoro

Questo esempio mostra come Deadline Cloud identifica i file di output generati dai tuoi lavori, decide se caricare tali file su Amazon S3 e come trasferirli sulla tua workstation.

Per questo esempio, usa il `job_attachments_devguide_output` job bundle anziché il `job_attachments_devguide` job bundle. Inizia creando una copia del pacchetto nel tuo AWS CloudShell ambiente dal tuo clone dell'archivio di esempi di Deadline Cloud: GitHub

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

La differenza importante tra questo job bundle e il `job_attachments_devguide` job bundle è l'aggiunta di un nuovo parametro di job nel modello di job:

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

La `dataFlow` proprietà del parametro ha il valore. `OUT` Deadline Cloud utilizza il valore dei parametri del `dataFlow` lavoro con un valore pari `OUT` o `INOUT` come output del lavoro. Se la posizione del file system passata come valore a questi tipi di parametri di lavoro viene rimappata a una posizione del file system locale sul lavoratore che esegue il lavoro, Deadline Cloud cercherà nuovi file nella posizione e li caricherà su Amazon S3 come output del lavoro.

Per vedere come funziona, avvia innanzitutto il worker agent di Deadline Cloud in una scheda. AWS CloudShell Lascia che tutti i lavori inviati in precedenza finiscano di essere eseguiti. Quindi elimina i registri dei lavori dalla directory dei registri:

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

Successivamente, invia un lavoro con questo pacchetto di offerte di lavoro. Dopo che il lavoratore è entrato a far parte delle tue CloudShell attività, guarda i log:

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

deadline config set settings.storage_profile_id $WSALL_ID

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

Il log mostra che un file è stato rilevato come output e caricato su Amazon S3:

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

Il registro mostra anche che Deadline Cloud ha creato un nuovo oggetto manifest nel bucket Amazon S3 configurato per l'utilizzo da parte degli allegati di lavoro in coda. Q1 Il nome dell'oggetto manifesto deriva dalla farm, dalla queue, dal job, dallo step, dal task, dal timestamp e dagli `sessionaction` identificatori dell'attività che ha generato l'output. Scarica questo file manifest per vedere dove Deadline Cloud ha inserito i file di output per questa attività:

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

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

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

Il manifesto ha il seguente aspetto:

```
{
```

```
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

Ciò dimostra che il contenuto del file di output viene salvato su Amazon S3 nello stesso modo in cui vengono salvati i file di input del lavoro. Analogamente ai file di input, il file di output viene archiviato in S3 con un nome di oggetto contenente l'hash del file e il prefisso. `DeadlineCloud/Data`

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

Puoi scaricare l'output di un lavoro sulla tua workstation utilizzando il monitor Deadline Cloud o la CLI di Deadline Cloud:

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

Il valore del parametro `OutputDir` job nel job inviato è`./output_dir`, quindi l'output viene scaricato in una directory chiamata `output_dir` all'interno della directory del job bundle. Se avete specificato un percorso assoluto o una posizione relativa diversa come valore per`OutputDir`, i file di output verranno invece scaricati in quella posizione.

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

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

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

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

Utilizzo di file da un passaggio a un passaggio dipendente

Questo esempio mostra come una fase di un processo può accedere agli output di una fase da cui dipende nello stesso processo.

Per rendere gli output di un passaggio disponibili per un altro, Deadline Cloud aggiunge azioni aggiuntive a una sessione per scaricare tali output prima di eseguire le attività nella sessione. Gli dici da quali passaggi scaricare gli output dichiarando tali passaggi come dipendenze del passaggio che deve utilizzare gli output.

Usa il `job_attachments_devguide_output` job bundle per questo esempio. Inizia creando una copia nel tuo AWS CloudShell ambiente dal tuo clone del repository di esempi di Deadline Cloud. GitHub Modificalo per aggiungere un passaggio dipendente che venga eseguito solo dopo il passaggio esistente e utilizzi l'output di quel passaggio:

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

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

Il processo creato con questo job bundle modificato viene eseguito come due sessioni separate, una per l'attività nel passaggio «Step» e la seconda per l'attività nel passaggio "DependentStep».

Per prima cosa avvia il worker agent di Deadline Cloud in una CloudShell scheda. Lascia terminare l'esecuzione di tutti i lavori inviati in precedenza, quindi elimina i log dei lavori dalla directory dei log:

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

Successivamente, invia un lavoro utilizzando il pacchetto di `job_attachments_devguide_output` lavori modificato. Attendi che finisca di essere eseguito sul lavoratore del tuo CloudShell ambiente. Guarda i log delle due sessioni:

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

deadline config set settings.storage_profile_id $WSALL_ID

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

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

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

Nel registro delle sessioni relativo all'attività indicata nella fase indicata`DependentStep`, vengono eseguite due azioni di download separate:

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

```
Total processing time of 0.03954 seconds at 5.23 KB/s.

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

La prima azione scarica il `script.sh` file utilizzato dal passaggio denominato «Step». La seconda azione scarica gli output di quel passaggio. Deadline Cloud determina quali file scaricare utilizzando il manifesto di output generato da quel passaggio come manifesto di input.

Più avanti nello stesso registro, puoi vedere l'output del passaggio denominato "DependentStep«:

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

Creare limiti di risorse per i lavori

I lavori inviati a Deadline Cloud possono dipendere da risorse condivise tra più lavori. Ad esempio, un'azienda agricola può avere più lavoratori rispetto alle licenze fluttuanti per una risorsa specifica. Oppure un file server condiviso può essere in grado di fornire dati solo a un numero limitato di lavoratori contemporaneamente. In alcuni casi, uno o più lavori possono richiedere tutte queste risorse, causando errori dovuti alla mancanza di risorse all'ingresso di nuovi lavoratori.

Per risolvere questo problema, puoi utilizzare dei limiti per queste risorse limitate. Deadline Cloud tiene conto della disponibilità di risorse limitate e utilizza tali informazioni per garantire che le risorse siano disponibili all'avvio dei nuovi dipendenti, in modo che i lavori abbiano una minore probabilità di fallire a causa della mancanza di risorse.

I limiti vengono creati per l'intera azienda agricola. I lavori inviati a una coda possono acquisire solo i limiti associati alla coda. Se si specifica un limite per un lavoro non associato alla coda, il lavoro non è compatibile e non verrà eseguito.

Per utilizzare un limite, devi

- Crea un limite
- Associa un limite e una coda
- Invia un lavoro che richiede dei limiti

> **Note**
>
> Se si esegue un processo con risorse limitate in una coda non associata a un limite, tale processo può consumare tutte le risorse. Se disponi di una risorsa limitata, assicurati che tutti i passaggi dei lavori nelle code che utilizzano la risorsa siano associati a un limite.

Per i limiti definiti in una farm, associati a una coda e specificati in un lavoro, può succedere una delle quattro cose seguenti:

- Se si crea un limite, lo si associa a una coda e si specifica il limite nel modello di un lavoro, il processo viene eseguito e utilizza solo le risorse definite nel limite.
- Se si crea un limite, lo si specifica in un modello di processo, ma non si associa il limite a una coda, il processo viene contrassegnato come incompatibile e non verrà eseguito.
- Se si crea un limite, non lo si associa a una coda e non si specifica il limite nel modello di un processo, il processo viene eseguito ma non utilizza il limite.
- Se non si utilizza affatto un limite, il processo viene eseguito.

Se si associa un limite a più code, le code condividono le risorse vincolate dal limite. Ad esempio, se si crea un limite di 100 e una coda utilizza 60 risorse, le altre code possono utilizzare solo 40 risorse. Quando una risorsa viene rilasciata, può essere utilizzata da un'operazione da qualsiasi coda.

Deadline Cloud fornisce due AWS CloudFormation metriche per aiutarti a monitorare le risorse fornite da un limite. Puoi monitorare il numero attuale di risorse in uso e il numero massimo di risorse disponibili entro il limite. Per ulteriori informazioni, consulta le metriche relative al limite delle risorse nella Deadline Cloud Developer Guide.

Applichi un limite a una fase di lavoro in un modello di lavoro. Quando si specifica la quantità, il nome del requisito di un limite nella `amounts` sezione `hostRequirements` di una fase e un limite corrispondente `amountRequirementName` viene associato alla coda del lavoro, le attività pianificate per questa fase sono vincolate dal limite della risorsa.

Se una fase richiede una risorsa limitata da un limite raggiunto, le attività di quella fase non verranno raccolte da altri lavoratori.

È possibile applicare più di un limite a una fase di lavoro. Ad esempio, se la fase utilizza due licenze software diverse, è possibile applicare un limite separato per ciascuna licenza. Se una fase richiede due limiti e viene raggiunto il limite per una delle risorse, le attività di quella fase non verranno acquisite da altri lavoratori finché le risorse non saranno disponibili.

Interruzione ed eliminazione dei limiti

Quando si interrompe o si elimina l'associazione tra una coda e un limite, un job che utilizza il limite interrompe la pianificazione delle attività relative ai passaggi che richiedono tale limite e blocca la creazione di nuove sessioni per un passaggio.

Le attività che si trovano nello stato READY rimangono pronte e le attività riprendono automaticamente quando l'associazione tra la coda e il limite diventa nuovamente attiva. Non è necessario richiedere alcun lavoro.

Quando interrompi o elimini l'associazione tra una coda e un limite, hai due scelte su come interrompere l'esecuzione delle attività:

- Interrompi e annulla le attività: i lavoratori con sessioni che hanno raggiunto il limite annullano tutte le attività.
- Interrompi e termina le attività in esecuzione: i lavoratori con sessioni che hanno raggiunto il limite completano le proprie attività.

Quando si elimina un limite utilizzando la console, i lavoratori interrompono innanzitutto l'esecuzione delle attività immediatamente o alla fine una volta completate. Quando l'associazione viene eliminata, accade quanto segue:

- Le fasi che richiedono il limite sono contrassegnate come non compatibili.
- L'intero processo contenente tali passaggi viene annullato, compresi i passaggi che non richiedono il limite.
- Il lavoro è contrassegnato come non compatibile.

Se alla coda associata al limite è associata una flotta con una capacità corrispondente all'importo richiesto (nome del limite), tale flotta continuerà a elaborare i lavori con il limite specificato.

Crea un limite

Puoi creare un limite utilizzando la console Deadline Cloud o l'CreateLimit operazione nell'API Deadline Cloud. I limiti sono definiti per una fattoria, ma associati alle code. Dopo aver creato un limite, è possibile associarlo a una o più code.

Per creare un limite

1. Dalla dashboard della console Deadline Cloud (console Deadline Cloud), seleziona la farm per cui desideri creare una coda.
2. Scegli la farm a cui aggiungere il limite, scegli la scheda Limiti, quindi scegli Crea limite.
3. Fornisci i dettagli del limite. Il nome del requisito di importo è il nome utilizzato nel modello di lavoro per identificare il limite. Deve iniziare con il prefisso **`amount.`** seguito dal nome dell'importo. Il nome del requisito relativo all'importo deve essere univoco nelle code associate al limite.
4. Se scegli Imposta un importo massimo, questo è il numero totale di risorse consentite da questo limite. Se scegli Nessun importo massimo, l'utilizzo delle risorse non è limitato. Anche quando l'utilizzo delle risorse non è limitato, viene emessa la CloudWatch metrica `CurrentCount` Amazon in modo da poter monitorare l'utilizzo. Per ulteriori informazioni, consulta le CloudWatchmetriche nella Deadline Cloud Developer Guide.
5. Se conosci già le code che devono utilizzare il limite, puoi sceglierle ora. Non è necessario associare una coda per creare un limite.
6. Scegli Crea limite.

Associa un limite e una coda

Dopo aver creato un limite, puoi associare una o più code al limite. Solo le code associate a un limite utilizzano i valori specificati nel limite.

Si crea un'associazione con una coda utilizzando la console Deadline Cloud o l'CreateQueueLimitAssociation operazione nell'API Deadline Cloud.

Per associare una coda a un limite

1. Dalla dashboard della console Deadline Cloud (console Deadline Cloud), seleziona la farm in cui desideri associare un limite a una coda.
2. Scegli la scheda Limiti, scegli il limite a cui associare una coda, quindi scegli Modifica limite.

3. Nella sezione Associa code, scegli le code da associare al limite.
4. Scegli Save changes (Salva modifiche).

Invia un lavoro che richiede dei limiti

Puoi applicare un limite specificandolo come requisito dell'host per il lavoro o la fase del lavoro. Se non si specifica un limite in una fase e tale fase utilizza una risorsa associata, l'utilizzo della fase non viene conteggiato nel limite quando i lavori sono pianificati.

Alcuni mittenti di Deadline Cloud ti consentono di impostare un requisito di host. Puoi specificare il nome del requisito relativo all'importo del limite nel mittente per applicare il limite.

Se il mittente non supporta l'aggiunta dei requisiti dell'host, puoi anche applicare un limite modificando il modello di lavoro relativo al lavoro.

Per applicare un limite a una fase di lavoro nel pacchetto di lavoro

1. Apri il modello di lavoro per il lavoro utilizzando un editor di testo. Il modello di lavoro si trova nella directory dei pacchetti di lavoro relativa al lavoro. Per ulteriori informazioni, consulta Job bundles nella Deadline Cloud Developer Guide.
2. Trova la definizione della fase a cui applicare il limite.
3. Aggiungi quanto segue alla definizione del passo. *amount.name* Sostituiscilo con il nome del requisito relativo all'importo del limite. Per un utilizzo tipico, è necessario impostare il `min` valore su 1.

   YAML

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

   JSON

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

```
        }
    }
}
```

È possibile aggiungere più limiti a una fase di lavoro come segue. Sostituisci *amount.name_1* e *amount.name_2* inserisci i nomi dei requisiti relativi agli importi indicati nei tuoi limiti.

YAML

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

JSON

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

4. Salva le modifiche al modello di lavoro.

Come inviare un'offerta di lavoro a Deadline Cloud

Esistono molti modi diversi per inviare offerte di lavoro a AWS Deadline Cloud. Questa sezione descrive alcuni dei modi in cui puoi inviare lavori utilizzando gli strumenti forniti da Deadline Cloud o creando strumenti personalizzati per i tuoi carichi di lavoro.

- Da un terminale, per quando stai sviluppando per la prima volta un pacchetto di lavori o quando gli utenti che inviano un lavoro si sentono a proprio agio utilizzando la riga di comando
- Da uno script: per personalizzare e automatizzare i carichi di lavoro
- Da un'applicazione: per quando il lavoro dell'utente è in un'applicazione o quando il contesto di un'applicazione è importante.

I seguenti esempi utilizzano la libreria `deadline` Python e lo strumento da `deadline` riga di comando. Entrambi sono disponibili PyPie ospitati su. GitHub

Argomenti

- Invia un lavoro a Deadline Cloud da un terminale
- Invia un lavoro a Deadline Cloud utilizzando uno script
- Invia un'offerta di lavoro all'interno di una candidatura

Invia un lavoro a Deadline Cloud da un terminale

Utilizzando solo un job bundle e la CLI di Deadline Cloud, tu o i tuoi utenti più tecnici potete iniziare rapidamente a scrivere pacchetti di lavoro per testare l'invio di un lavoro. Usa il seguente comando per inviare un job bundle:

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

Se invii un job bundle con parametri che non hanno valori predefiniti nel bundle, puoi specificarli con l'opzione/. `-p` `--parameter`

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

Per un elenco completo delle opzioni disponibili, esegui il comando help:

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

Invia un lavoro a Deadline Cloud utilizzando una GUI

La CLI di Deadline Cloud è inoltre dotata di un'interfaccia utente grafica che consente agli utenti di visualizzare i parametri che devono fornire prima di inviare un lavoro. Se i tuoi utenti preferiscono non

interagire con la riga di comando, puoi scrivere una scorciatoia sul desktop che apra una finestra di dialogo per inviare un pacchetto di lavori specifico:

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

Utilizza l'`--browse`opzione can in modo che l'utente possa selezionare un pacchetto di lavori:

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

Per un elenco completo delle opzioni disponibili, esegui il comando help:

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

Invia un lavoro a Deadline Cloud utilizzando uno script

Per automatizzare l'invio di lavori a Deadline Cloud, puoi creare script utilizzando strumenti come bash, Powershell e file batch.

È possibile aggiungere funzionalità come la compilazione dei parametri del lavoro da variabili di ambiente o altre applicazioni. Puoi anche inviare più lavori di seguito o creare uno script per la creazione di un pacchetto di lavori da inviare.

Invia un lavoro usando Python

Deadline Cloud dispone anche di una libreria Python open source per interagire con il servizio. Il codice sorgente è disponibile su. GitHub

La libreria è disponibile su pypi tramite pip ()`pip install deadline`. È la stessa libreria utilizzata dallo strumento CLI Deadline Cloud:

```
from deadline.client import api

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

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

Per creare una finestra di dialogo come il `deadline bundle gui-submit` comando, puoi usare `show_job_bundle_submitter` la funzione di. `deadline.client.ui.job_bundle_submitter`

L'esempio seguente avvia un'applicazione Qt e mostra il job bundle submitter:

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

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

Per creare la tua finestra di dialogo puoi usare la `SubmitJobToDeadlineDialog` classe in. `deadline.client.ui.dialogs.submit_job_to_deadline_dialog` Puoi trasmettere valori, incorporare la tua scheda specifica del lavoro e determinare come il job bundle viene creato (o passato).

Invia un'offerta di lavoro all'interno di una candidatura

Per facilitare l'invio dei lavori da parte degli utenti, è possibile utilizzare i runtime di script o i sistemi di plug-in forniti da un'applicazione. Gli utenti dispongono di un'interfaccia familiare ed è possibile creare potenti strumenti che assistono gli utenti nell'invio di un carico di lavoro.

Incorpora pacchetti di lavoro in un'applicazione

Questo esempio dimostra l'invio di pacchetti di lavoro che rendi disponibili nell'applicazione.

Per consentire a un utente di accedere a questi pacchetti di lavoro, crea uno script incorporato in una voce di menu che avvia la CLI di Deadline Cloud.

Lo script seguente consente a un utente di selezionare il job bundle:

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

Per utilizzare invece un job bundle specifico in una voce di menu, utilizzate quanto segue:

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

Si apre una finestra di dialogo in cui l'utente può modificare i parametri, gli input e gli output del lavoro e quindi inviare il lavoro. È possibile avere diverse voci di menu per diversi pacchetti di lavoro che un utente può inviare in una candidatura.

Se il lavoro che invii con un pacchetto di offerte di lavoro contiene parametri e riferimenti alle risorse simili tra gli invii, puoi inserire i valori predefiniti nel pacchetto di lavoro sottostante.

Ottieni informazioni da una candidatura

Per estrarre informazioni da un'applicazione in modo che gli utenti non debbano aggiungerle manualmente all'invio, puoi integrare Deadline Cloud con l'applicazione in modo che gli utenti possano inviare lavori utilizzando un'interfaccia familiare senza dover uscire dall'applicazione o utilizzare strumenti da riga di comando.

Se la tua applicazione ha un runtime di scripting che supporta Python e pyside/pyqt, puoi utilizzare i componenti della GUI dalla libreria client Deadline Cloud per creare un'interfaccia utente. Per un esempio, consulta l'integrazione di Deadline Cloud for Maya su. GitHub

La libreria client Deadline Cloud fornisce operazioni che eseguono le seguenti operazioni per aiutarti a fornire un'esperienza utente solida e integrata:

- Recupera i parametri di ambiente della coda, i parametri di lavoro e i riferimenti agli asset dalle variabili di ambiente e chiamando l'SDK dell'applicazione.
- Imposta i parametri nel job bundle. Per evitare di modificare il pacchetto originale, è necessario creare una copia del pacchetto e inviare la copia.

Se si utilizza il `deadline bundle gui-submit` comando per inviare il job bundle, è necessario digitare programmaticamente `asset_references.yaml` i file `parameter_values.yaml` and per passare le informazioni dall'applicazione. Per ulteriori informazioni su questi file, vedere. Modelli Open Job Description (OpenJD) per Deadline Cloud

Se hai bisogno di controlli più complessi di quelli offerti da OpenJD, hai bisogno di astrarre il lavoro dall'utente o vuoi fare in modo che l'integrazione corrisponda allo stile visivo dell'applicazione, puoi scrivere la tua finestra di dialogo che richiami la libreria client di Deadline Cloud per inviare il lavoro.

Pianifica lavori in Deadline Cloud

Dopo aver creato un lavoro, AWS Deadline Cloud ne pianifica l'elaborazione su una o più flotte associate a una coda. La flotta che elabora una particolare attività viene scelta in base alla configurazione di pianificazione, alle funzionalità configurate per la flotta e ai requisiti dell'host di una fase specifica.

Le sezioni seguenti forniscono dettagli sul processo di pianificazione di un lavoro.

Configurazioni di pianificazione

Puoi configurare il modo in cui Deadline Cloud pianifica i lavori in una coda impostando una configurazione di pianificazione sulla coda. La configurazione della pianificazione controlla il modo in cui i lavoratori vengono distribuiti tra i lavori.

Puoi impostare la configurazione di pianificazione utilizzando la console Deadline Cloud o chiamando o. CreateQueueUpdateQueue APIs

Sono disponibili tre configurazioni di pianificazione:

- Priority, first-in-first-out (`priorityFifo`): pianifica per primo il lavoro inviato per primo e con la priorità più alta (impostazione predefinita).
- Priorità, bilanciata (`priorityBalanced`): distribuisce i lavoratori in modo uniforme tra le mansioni con la massima priorità.
- Ponderato, bilanciato (`weightedBalanced`): utilizza una formula ponderata per determinare la distribuzione dei lavoratori tra le mansioni.

In tutte le configurazioni di pianificazione, le attività in corso vengono completate prima che venga presa una nuova decisione di pianificazione. Se si modifica la configurazione di pianificazione mentre le attività sono in esecuzione, la modifica si applica solo alla successiva assegnazione dei lavoratori. Le attività in esecuzione non vengono interrotte o riassegnate.

Priorità, first-in-first-out

Priority, first-in-first-out (`priorityFifo`) è la configurazione di pianificazione predefinita per le nuove code. Deadline Cloud assegna innanzitutto ai lavoratori il lavoro con la massima priorità. Quando più lavori condividono la stessa priorità, il lavoro più vecchio (inviato per primo) riceve per primo tutti i lavoratori disponibili.

Usa la priorità FIFO quando desideri un ordine rigoroso dei lavori. Questa configurazione è appropriata quando i lavori devono essere completati uno alla volta nell'ordine in cui sono stati inviati, ad esempio nelle fasi sequenziali della pipeline o nell'elaborazione in batch in cui ogni lavoro deve terminare prima dell'inizio di quello successivo.

Questa configurazione non ha parametri aggiuntivi.

Priorità, equilibrata

Priority, balanced (`priorityBalanced`) distribuisce i lavoratori in modo uniforme tra tutti i posti di lavoro al livello di priorità più elevato. Quando esiste un solo lavoro con la massima priorità, Deadline Cloud assegna a tutti i lavoratori quel lavoro. Quando più lavori condividono la massima priorità, i lavoratori vengono suddivisi equamente tra loro. Se i lavoratori non possono essere suddivisi equamente, i lavoratori aggiuntivi vengono distribuiti tra i lavori con la priorità più alta.

Usa la priorità bilanciata quando più artisti o utenti inviano lavori con la stessa priorità e ogni utente ha bisogno di un feedback immediato. Questa configurazione garantisce che nessun lavoro monopolizzi tutti i lavoratori disponibili, in modo che a tutti gli utenti vengano assegnati lavoratori subito dopo l'invio.

Se un posto di lavoro ha meno mansioni rimanenti rispetto alla quota di lavoratori, i lavoratori in eccedenza vengono ridistribuiti ad altri lavori con lo stesso livello di priorità. Se tutti i posti di lavoro con la massima priorità sono interamente assegnati, i lavoratori in eccedenza vengono assegnati a cascata ai posti di lavoro con il livello di priorità successivo più elevato.

Questa configurazione ha il seguente parametro:

`renderingTaskBuffer`

Controlla la viscosità dell'operatore. Un lavoratore passa dal lavoro corrente a un altro lavoro con la stessa priorità solo se la differenza nella resa delle attività supera il valore. `renderingTaskBuffer` Un valore più elevato consente ai lavoratori di continuare a lavorare più a lungo, riducendo il cambio di contesto. Il valore predefinito è `1`.

Ponderato, equilibrato

Weighted, balanced (`weightedBalanced`) utilizza una formula per calcolare un peso per ogni lavoro. Deadline Cloud assegna innanzitutto ai lavoratori il lavoro con il peso maggiore. Se più lavori hanno lo stesso peso, i lavoratori vengono distribuiti tra di essi.

Usa la bilancia ponderata quando hai bisogno di un controllo preciso sulla distribuzione dei lavoratori tra le mansioni con priorità, tassi di errore e tempi di invio diversi. Questa configurazione è appropriata per ambienti di render farm complessi in cui si desidera ottimizzare l'equilibrio tra priorità del lavoro, età del lavoro, gestione degli errori e persistenza dei lavoratori.

Il peso per ogni lavoro viene calcolato come segue:

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

Il `renderingTaskBuffer` componente viene applicato solo se il lavoratore sta attualmente lavorando sul posto di lavoro. Di solito `renderingTaskWeight` è impostato su un valore negativo in modo che i lavori con lavoratori assegnati ricevano un peso inferiore, portando gli altri lavori in prima fila. Inoltre, di solito `errorWeight` è negativo, quindi i lavori con errori vengono ridotti in ordine di priorità. È possibile utilizzare le sostituzioni di pianificazione per i lavori con priorità minima e massima.

Questa configurazione ha i seguenti parametri:

`priorityWeight`

Il peso applicato alla priorità di un lavoro. Un valore positivo indica che i lavori con priorità più alta vengono pianificati per primi. Il valore predefinito è `100.0`. Intervallo: `0` fino a. `10000`

`errorWeight`

Il peso applicato al conteggio degli errori di un lavoro. Un valore negativo indica che i lavori senza errori vengono pianificati per primi. Il valore predefinito è `-10.0`. Intervallo: `-10000` fino a`10000`.

`submissionTimeWeight`

Il peso applicato al tempo di invio di un lavoro (in secondi). Un valore positivo indica che i lavori inviati in precedenza vengono programmati per primi. Il valore predefinito è `3.0`. Intervallo: `0` fino a`10000`.

`renderingTaskWeight`

Il peso applicato al numero di attività attualmente renderizzate per un lavoro. Un valore negativo indica che i prossimi lavori con meno lavoratori sono programmati. Il valore predefinito è `-100.0`. Intervallo: `-10000` fino a`10000`.

`renderingTaskBuffer`

Il numero di attività di rendering prima che il peso dell'attività di rendering abbia effetto. Un valore positivo consente ai lavoratori di continuare a svolgere il loro lavoro attuale. Il valore predefinito è 1. Intervallo: `0` fino a`1000`.

`maxPriorityOverride`

Opzionale. Se impostato su`alwaysScheduleFirst`, i lavori con la priorità massima (100) vengono sempre programmati prima degli altri lavori, indipendentemente dalla formula ponderata. Quando più lavori hanno la massima priorità, i pareggi vengono risolti utilizzando la formula ponderata standard. Quando l'override è assente, i lavori con priorità massima utilizzano la formula ponderata standard senza alcun trattamento speciale.

`minPriorityOverride`

Opzionale. Se impostato su`alwaysScheduleLast`, i lavori con la priorità minima (0) vengono sempre programmati dopo gli altri lavori, indipendentemente dalla formula ponderata. Quando più lavori hanno la priorità minima, i pareggi vengono risolti utilizzando la formula ponderata standard. Quando l'esclusione è assente, i lavori con priorità minima utilizzano la formula ponderata standard senza alcun trattamento speciale.

Determina la compatibilità della flotta

Dopo aver creato un lavoro, Deadline Cloud verifica i requisiti dell'host per ogni fase del lavoro rispetto alle capacità delle flotte associate alla coda a cui è stato inviato il lavoro. Se una flotta soddisfa i requisiti dell'host, il lavoro viene assegnato allo stato. `READY`

Se una fase del lavoro presenta requisiti che non possono essere soddisfatti da una flotta associata alla coda, lo stato della fase viene impostato `NOT_COMPATIBLE` su. Inoltre, gli altri passaggi del processo vengono annullati.

Le capacità di una flotta sono impostate a livello di flotta. Anche se un lavoratore di una flotta soddisfa i requisiti del lavoro, non gli verranno assegnati i compiti previsti dal lavoro se la flotta non soddisfa i requisiti del lavoro.

Il seguente modello di lavoro presenta una fase che specifica i requisiti dell'host per la fase:

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
        - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
    # Capabilities starting with "amount." are amount capabilities. If they start with
 "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
 usage.
    - name: amount.worker.vcpu
      min: 4
      max: 8
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Questo lavoro può essere programmato per una flotta con le seguenti funzionalità:

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

Questo lavoro non può essere programmato per una flotta con nessuna delle seguenti funzionalità:

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

```
    The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

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

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
    The osFamily doesn't match.
```

Ridimensionamento della flotta

Quando un lavoro viene assegnato a una flotta compatibile gestita dai servizi, la flotta viene ridimensionata automaticamente. Il numero di lavoratori della flotta cambia in base al numero di attività disponibili per l'esecuzione della flotta.

Quando un lavoro viene assegnato a una flotta gestita dal cliente, è possibile che i lavoratori esistano già o possano essere creati utilizzando la scalabilità automatica basata su eventi. Per ulteriori informazioni, consulta Use EventBridge to handle auto scaling events nella Amazon EC2 Auto Scaling User Guide.

Sessioni

Le attività di un job sono suddivise in una o più sessioni. I lavoratori eseguono le sessioni per configurare l'ambiente, eseguire le attività e quindi demolire l'ambiente. Ogni sessione è composta da una o più azioni che un lavoratore deve intraprendere.

Quando un lavoratore completa le azioni della sezione, al lavoratore possono essere inviate azioni di sessione aggiuntive. Il lavoratore riutilizza gli ambienti e gli allegati di lavoro esistenti nella sessione per completare le attività in modo più efficiente.

Sui lavoratori della flotta gestiti dal servizio, le directory delle sessioni vengono eliminate al termine della sessione, ma le altre directory vengono conservate tra le sessioni. Questo comportamento

consente di implementare strategie di memorizzazione nella cache per i dati che possono essere riutilizzati in più sessioni. Per memorizzare nella cache i dati tra le sessioni, memorizzali nella home directory dell'utente che esegue il processo. Ad esempio, i pacchetti conda vengono memorizzati nella cache nella home directory dell'utente del lavoro all'indirizzo `C:\Users\job-user\.conda-pkgs` on Windows workers e `/home/job-user/.conda-pkgs` on Linux workers. Questi dati rimangono disponibili fino alla chiusura del lavoratore.

Gli allegati dei lavori vengono creati dal mittente che utilizzi come parte del pacchetto di lavori CLI di Deadline Cloud. Puoi anche creare allegati di lavoro utilizzando l'opzione relativa al comando. `--attachments` `create-job` AWS CLI Gli ambienti sono definiti in due punti: ambienti di coda collegati a una coda specifica e ambienti di lavoro e fasi definiti nel modello di lavoro.

Esistono quattro tipi di azioni di sessione:

- `syncInputJobAttachments`— Scarica gli allegati del lavoro di input per il lavoratore.
- `envEnter`— Esegue le on`Enter` azioni per un ambiente.
- `taskRun`— Esegue le onRun azioni relative a un'attività.
- `envExit`— Esegue le on`Exit` azioni per un ambiente.

Il seguente modello di lavoro ha un ambiente a fasi. Ha una on`Enter` definizione per configurare l'ambiente delle fasi, una onRun definizione che definisce l'attività da eseguire e una on`Exit` definizione per eliminare l'ambiente delle fasi. Le sessioni create per questo lavoro includeranno un'`envEnter`azione, una o più `task`Run azioni e quindi un'`envExit`azione.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
```

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

Pipeline delle azioni della sessione

La pipeline delle azioni di sessione consente a uno scheduler di preassegnare più azioni di sessione a un lavoratore. Il lavoratore può quindi eseguire queste azioni in sequenza, riducendo o eliminando i tempi di inattività tra le attività.

Per creare un'assegnazione iniziale, lo scheduler crea una sessione con un'attività, il lavoratore completa l'attività e quindi lo scheduler analizza la durata dell'attività per determinare le assegnazioni future.

Affinché lo scheduler sia efficace, esistono delle regole sulla durata delle attività. Per le attività di durata inferiore a un minuto, lo scheduler utilizza un modello di crescita power-of-2. Ad esempio, per un'attività di 1 secondo, lo scheduler assegna 2 nuove attività, quindi 4, quindi 8. Per le attività di durata superiore a un minuto, lo scheduler assegna solo una nuova attività e la pipelining rimane disabilitata.

Per calcolare le dimensioni della pipeline, lo scheduler esegue le seguenti operazioni:

- Utilizza la durata media delle attività completate
- Mira a mantenere il lavoratore occupato per un minuto
- Considera solo le attività all'interno della stessa sessione
- Non condivide i dati sulla durata tra i lavoratori

Grazie alla pipeline delle azioni di sessione, i lavoratori iniziano immediatamente nuove attività e non ci sono tempi di attesa tra le richieste dello scheduler. Fornisce inoltre una maggiore efficienza dei lavoratori e una migliore distribuzione delle attività per i processi a lunga durata.

Inoltre, se è disponibile un nuovo lavoro con priorità più alta, il lavoratore terminerà tutto il lavoro precedentemente assegnato prima della fine della sessione corrente e prima che venga assegnata una nuova sessione da un lavoro con priorità più alta.

Dipendenze tra fasi

Deadline Cloud supporta la definizione delle dipendenze tra i passaggi in modo che un passaggio attenda il completamento di un altro passaggio prima di iniziare. Puoi definire più di una dipendenza per un passaggio. Un passaggio con una dipendenza non è pianificato fino al completamento di tutte le relative dipendenze.

Se il modello di lavoro definisce una dipendenza circolare, il lavoro viene rifiutato e lo stato del processo viene impostato su. `CREATE_FAILED`

Il seguente modello di lavoro crea un lavoro con due passaggi. `StepB`dipende da`StepA`. `StepB`viene eseguito solo dopo essere stato `StepA` completato con successo.

Dopo la creazione del lavoro, `StepA` si trova nello `READY` stato e `StepB` si trova nello `PENDING` stato. Al `StepA` termine, `StepB` passa allo `READY` stato. Se `StepA` fallisce o se `StepA` viene annullato, `StepB` passa allo `CANCELED` stato.

È possibile impostare una dipendenza su più passaggi. Ad esempio, `StepC` dipende da entrambi `StepA` e`StepB`, `StepC` non inizierà fino al termine degli altri due passaggi.

Le dipendenze dei passaggi hanno le seguenti restrizioni:

- Dipendenze per fase: una fase può dipendere da un massimo di 128 altre fasi.
- Consumatori per fase: un massimo di altri 32 passaggi possono dipendere da un singolo passaggio.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task A Done!
- name: B
  dependencies:
  - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
```

```
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task B Done!
```

Modifica un lavoro in Deadline Cloud

Puoi utilizzare i seguenti `update` comandi AWS Command Line Interface (AWS CLI) per modificare la configurazione di un lavoro o per impostare lo stato di destinazione di un lavoro, un passaggio o un'attività:

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

Nei seguenti esempi di `update` comandi, sostituiteli *`user input placeholder`* con le vostre informazioni.

Example— Richiedere un lavoro

Tutte le attività del lavoro passano `READY` allo stato, a meno che non vi siano dipendenze tra fasi. I passaggi con dipendenze passano a uno o all'altro `READY` o `PENDING` man mano che vengono ripristinati.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status PENDING
```

Example— Annullare un lavoro

Tutte le attività del lavoro che non hanno lo stato `SUCCEEDED` o `FAILED` sono contrassegnate`CANCELED`.

```
aws deadline update-job \
```

```
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example— Contrassegna un lavoro come non riuscito

Tutte le attività del lavoro che hanno lo stato SUCCEEDED rimangono invariate. Tutte le altre attività sono contrassegnateFAILED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example— Contrassegna un lavoro riuscito

Tutte le attività lavorative vengono trasferite allo SUCCEEDED stato.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example— Sospendere un lavoro

Le attività del lavoro nello SUCCEEDED FAILED stato o non cambiano. CANCELED Tutte le altre attività sono contrassegnateSUSPENDED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUSPENDED
```

Example— Modificare la priorità di un lavoro

Aggiorna la priorità di un lavoro in coda per modificare l'ordine in cui è pianificato. I lavori con priorità più alta vengono generalmente pianificati per primi.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example— Modificare il numero di attività non riuscite consentite

Aggiorna il numero massimo di attività non riuscite che il lavoro può avere prima che le attività rimanenti vengano annullate.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example— Modifica il numero di tentativi consentiti per le nuove attività

Aggiorna il numero massimo di tentativi per un'attività prima che l'operazione abbia esito negativo. Un'attività che ha raggiunto il numero massimo di tentativi non può essere richiesta finché questo valore non viene aumentato.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example— Archivia un lavoro

Aggiorna lo stato del ciclo di vita del lavoro su. `ARCHIVED` I lavori archiviati non possono essere pianificati o modificati. È possibile archiviare solo un lavoro che si trova nello `SUSPENDED` stato `FAILEDCANCELED`,`SUCCEEDED`, o.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example— Modificare il nome di un lavoro

Aggiorna il nome visualizzato di un lavoro. Il nome del lavoro può contenere fino a 128 caratteri.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example— Modificare la descrizione di un lavoro

Aggiorna la descrizione di un lavoro. La descrizione può contenere fino a 2048 caratteri. Per rimuovere la descrizione esistente, passa una stringa vuota.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example— Richiedi un passaggio

Tutte le attività della fase passano `READY` allo stato, a meno che non vi siano dipendenze tra fasi. Le attività in fasi con dipendenze passano a uno `READY` o all'altro e `PENDING` l'attività viene ripristinata.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example— Annullare un passaggio

Tutte le attività del passaggio che non hanno lo stato `SUCCEEDED` o `FAILED` sono contrassegnate`CANCELED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
```

```
--step-id stepID \
--target-task-run-status CANCELED
```

Example— Contrassegna un passaggio come fallito

Tutte le attività del passaggio che hanno lo stato `SUCCEEDED` rimangono invariate. Tutte le altre attività sono contrassegnate`FAILED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example— Contrassegna un passaggio riuscito

Tutte le attività del passaggio sono contrassegnate`SUCCEEDED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example— Sospendere un passaggio

Le attività del passaggio nello `SUCCEEDED` `FAILED` stato o non vengono modificate. `CANCELED` Tutte le altre attività sono contrassegnate`SUSPENDED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example— Modificare lo stato di un'attività

Quando si utilizza il comando CLI `update-task` Deadline Cloud, l'attività passa allo stato specificato.

```
aws deadline update-task \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--task-id taskID \
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

Crea e utilizza flotte gestite dai clienti di Deadline Cloud

Quando crei una flotta gestita dal cliente (CMF), hai il pieno controllo sulla pipeline di elaborazione. Siete voi a definire l'ambiente di rete e software per ogni lavoratore. Deadline Cloud funge da archivio e pianificatore per i tuoi lavori.

Un lavoratore può essere un'istanza Amazon Elastic Compute Cloud (Amazon EC2), un lavoratore in una struttura in co-location o un lavoratore locale. Ogni lavoratore deve eseguire l'agente di lavoro Deadline Cloud. Tutti i lavoratori devono avere accesso all'endpoint del servizio Deadline Cloud.

I seguenti argomenti mostrano come creare un CMF di base utilizzando istanze Amazon EC2.

Argomenti

- Crea una flotta gestita dai clienti
- Configurazione e configurazione dell'host di lavoro
- Gestisci l'accesso ai segreti degli utenti del Windows lavoro
- Installa e configura il software necessario per i lavori
- Configurazione delle credenziali AWS
- Flusso di dati host dei lavoratori per flotte gestite dai clienti
- Verifica la configurazione del tuo host di lavoro
- Crea un Amazon Machine Image
- Crea un'infrastruttura per il parco veicoli con un gruppo Amazon EC2 Auto Scaling

Crea una flotta gestita dai clienti

Per creare una flotta gestita dal cliente (CMF), completa i seguenti passaggi.

Deadline Cloud console

Per utilizzare la console Deadline Cloud per creare una flotta gestita dal cliente

1. Apri la console Deadline Cloud.
2. Seleziona Fattorie. Viene visualizzato un elenco di fattorie disponibili.
3. Seleziona il nome della fattoria in cui vuoi lavorare.
4. Seleziona la scheda Flotte, quindi scegli Crea flotta.

5. Inserisci un nome per la tua flotta.
6. (Facoltativo) Inserisci una descrizione per la tua flotta.
7. Seleziona Gestito dal cliente per il tipo di flotta.
8. Seleziona l'accesso al servizio della tua flotta.
    a. Ti consigliamo di utilizzare l'opzione Crea e utilizza un nuovo ruolo di servizio per ogni flotta per un controllo più granulare delle autorizzazioni. Questa opzione è selezionata per impostazione predefinita.
    b. Puoi anche utilizzare un ruolo di servizio esistente selezionando Scegli un ruolo di servizio.
9. Controlla le tue selezioni, quindi scegli Avanti.
10. Seleziona un sistema operativo per la tua flotta. Tutti i dipendenti della flotta devono disporre di un sistema operativo comune.
11. Seleziona l'architettura della CPU host.
12. Seleziona le funzionalità hardware di memoria e vCPU minime e massime per soddisfare le esigenze di carico di lavoro delle tue flotte.
13. Seleziona un tipo di Auto Scaling. Per ulteriori informazioni, consultate Utilizzare EventBridge per gestire gli eventi di Auto Scaling.
    - Nessuna scalabilità: stai creando una flotta locale e desideri rinunciare a Deadline Cloud Auto Scaling.
    - Consigli di scalabilità: stai creando una flotta Amazon Elastic Compute Cloud (Amazon EC2).
14. (Facoltativo) Seleziona la freccia per espandere la sezione Aggiungi funzionalità.
15. (Facoltativo) Seleziona la casella di controllo Aggiungi funzionalità GPU - Facoltativo, quindi inserisci il valore minimo e massimo GPUs e la memoria.
16. Controlla le tue selezioni, quindi scegli Avanti.
17. (Facoltativo) Definisci le funzionalità personalizzate dei lavoratori, quindi scegli Avanti.
18. Utilizzando il menu a discesa, seleziona una o più code da associare alla flotta.

> **Note**
>
> Ti consigliamo di associare un parco veicoli solo a code che si trovano tutte all'interno dello stesso limite di trust. Questa raccomandazione garantisce un forte limite di sicurezza tra l'esecuzione di lavori sullo stesso lavoratore.

19. Controlla le associazioni delle code, quindi scegli Avanti.
20. (Facoltativo) Per l'ambiente di coda Conda predefinito, creeremo un ambiente per la coda che installerà i pacchetti conda richiesti dai job.

> **Note**
>
> L'ambiente di coda conda viene utilizzato per installare i pacchetti conda richiesti dai job. In genere, è necessario deselezionare l'ambiente di coda conda sulle code associate a CMFs perché CMFs non verranno installati i comandi conda richiesti per impostazione predefinita.

21. (Facoltativo) Aggiungi tag al tuo CMF. Per ulteriori informazioni, consulta Taggare le AWS risorse.
22. Controlla la configurazione del parco veicoli e apporta eventuali modifiche, quindi scegli Crea flotta.
23. Seleziona la scheda Flotte, quindi annota l'ID della flotta.

AWS CLI

Da utilizzare per AWS CLI creare una flotta gestita dal cliente

1. Apri un terminale.
2. Crea `fleet-trust-policy.json` in un nuovo editor.
   a. Aggiungi la seguente policy IAM, sostituendo il *`ITALICIZED`* testo con l'ID AWS dell'account e l'ID della farm Deadline Cloud.

      JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:deadline:*:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

b. Salvare le modifiche.

3. Creare il `fleet-policy.json`.

   a. Aggiungi la seguente politica IAM.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:AssumeFleetRoleForWorker",
                "deadline:UpdateWorker",
                "deadline:DeleteWorker",
                "deadline:UpdateWorkerSchedule",
                "deadline:BatchGetJobEntity",
                "deadline:AssumeQueueRoleForWorker"
            ],
            "Resource": "arn:aws:deadline:*:111122223333:*",
```

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

b. Salvare le modifiche.

4. Aggiungi un ruolo IAM da utilizzare per i lavoratori della tua flotta.

```
aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
 FleetWorkerPolicy --policy-document file://fleet-policy.json
```

5. Creare il `create-fleet-request.json`.

a. Aggiungi la seguente policy IAM, sostituendo il testo IN CORSIVO con i valori del tuo CMF.

> **Note**
>
> Puoi trovarli nel. *ROLE_ARN* `create-cmf-fleet.json`
> Per il*OS_FAMILY*, devi scegliere uno dei`linux`, `macos` o`windows`.

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
}
```

b. Salvare le modifiche.

6. Crea la tua flotta.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

Configurazione e configurazione dell'host di lavoro

Un worker host si riferisce a una macchina host che esegue un worker Deadline Cloud. Questa sezione spiega come configurare l'host di lavoro e configurarlo per esigenze specifiche. Ogni worker host esegue un programma chiamato worker agent. L'agente di lavoro è responsabile di:

- Gestione del ciclo di vita del lavoratore.
- Sincronizzazione del lavoro assegnato, dello stato di avanzamento e dei risultati.
- Monitoraggio del lavoro in corso.
- Inoltro dei log a destinazioni configurate.

Ti consigliamo di utilizzare l'agente di lavoro Deadline Cloud fornito. Il worker agent è open source e incoraggiamo la richiesta di funzionalità, ma puoi anche svilupparlo e personalizzarlo in base alle tue esigenze.

Per completare le attività descritte nelle seguenti sezioni, è necessario quanto segue:

Linux

- Un'istanza Amazon Elastic Compute Cloud (Amazon EC2) Linux basata su Amazon Elastic Compute Cloud (Amazon EC2). Consigliamo Amazon Linux 2023.
- `sudo`privilegi
- Python 3.9 o versioni successive

Windows

- Un'istanza Amazon Elastic Compute Cloud (Amazon EC2) Windows basata su Amazon Elastic Compute Cloud (Amazon EC2). Consigliamo. Windows Server 2022

- Accesso dell'amministratore all'host del lavoratore

- Python 3.9 o superiore installato per tutti gli utenti

Creare e configurare un ambiente virtuale Python

Puoi creare un ambiente virtuale Python su Linux se hai installato Python 3.9 o versione successiva e lo hai inserito nel tuo. PATH

> **Note**
>
> SìWindows, i file dell'agente devono essere installati nella directory globale dei pacchetti del sito di Python. Gli ambienti virtuali Python non sono attualmente supportati.

Per creare e attivare un ambiente virtuale Python

1. Apri un terminale come `root` utente (o usa`sudo`/`su`).
2. Crea e attiva un ambiente virtuale Python.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Installa l'agente di lavoro Deadline Cloud

Dopo aver configurato Python e creato un ambiente virtualeLinux, installa i pacchetti Python dell'agente di lavoro di Deadline Cloud.

Per installare i pacchetti Python dell'agente di lavoro

Linux

1. Apri un terminale come `root` utente (o usa`sudo`/`su`).
2. Scarica e installa i pacchetti Deadline Cloud worker agent da PyPI:

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. Apri un prompt dei comandi o un terminale dell'amministratore. PowerShell

2. Scarica e installa i pacchetti Deadline Cloud worker agent da PyPI:

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

Quando il tuo Windows host di lavoro richiede nomi di percorso lunghi (più di 250 caratteri), devi abilitare i nomi di percorso lunghi come segue:

Per abilitare percorsi lunghi per gli host Windows di lavoro

1. Assicurati che la chiave di registro a percorso lungo sia abilitata. Per ulteriori informazioni, vedere Impostazione del registro per abilitare i percorsi di registro sul sito Web di Microsoft.
2. Installa l'WindowsSDK per le app desktop C++ x86. Per ulteriori informazioni, consulta WindowsSDK nel Dev Center. Windows
3. Apri la posizione di installazione di Python nell'ambiente in cui è installato l'agente di lavoro. Il valore predefinito è `C:\Program Files\Python311`. Esiste un file eseguibile denominato`pythonservice.exe`.
4. Crea un nuovo file chiamato `pythonservice.exe.manifest` nella stessa posizione. Aggiungi quanto segue:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
 version="1.0.0.0"/>
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. Apri un prompt dei comandi ed esegui il comando seguente nella posizione del file manifesto che hai creato:

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

Verrà visualizzato un output simile al seguente:

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

Il lavoratore è ora in grado di accedere a percorsi lunghi. Per eseguire la pulizia, rimuovi il `pythonservice.exe.manifest` file e disinstalla l'SDK.

Configura l'agente di lavoro Deadline Cloud

Puoi configurare le impostazioni dell'agente Deadline Cloud Worker in tre modi. Ti consigliamo di utilizzare la configurazione del sistema operativo eseguendo lo `install-deadline-worker` strumento.

L'agente di lavoro non supporta l'esecuzione come utente di dominio su Windows. Per eseguire un processo come utente di dominio, è possibile specificare un account utente di dominio quando si configura un utente in coda per l'esecuzione dei lavori. Per ulteriori informazioni, consulta il passaggio 7 delle code di Deadline Cloud nella Guida per l'utente di AWS Deadline Cloud.

Argomenti della riga di comando: puoi specificare gli argomenti quando esegui l'agente di lavoro Deadline Cloud dalla riga di comando. Alcune impostazioni di configurazione non sono disponibili tramite gli argomenti della riga di comando. Per visualizzare tutti gli argomenti disponibili nella riga di comando, digitare`deadline-worker-agent --help`.

Variabili di ambiente: puoi configurare l'agente di lavoro di Deadline Cloud impostando la variabile di ambiente che inizia con`DEADLINE_WORKER_`. Ad esempio, per visualizzare tutti gli argomenti disponibili della riga di comando, puoi utilizzare `export DEADLINE_WORKER_VERBOSE=true` per impostare l'output del worker agent su verboso. Per ulteriori esempi e informazioni, vedere `/etc/amazon/deadline/worker.toml.example` on Linux or `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` onWindows.

File di configurazione: quando si installa il worker agent, viene creato un file di configurazione che si trova in `/etc/amazon/deadline/worker.toml` on Linux o `C:\ProgramData\Amazon\Deadline\Config\worker.toml` onWindows. Il worker agent carica questo file di configurazione all'avvio. È possibile utilizzare il file di configurazione di esempio (`/etc/amazon/deadline/worker.toml.example`LinuxC:\ProgramData\Amazon\Deadline\Config\worker.toml.exampleacceso o attivatoWindows) per personalizzare il file di configurazione del worker agent predefinito in base alle proprie esigenze specifiche.

Infine, ti consigliamo di abilitare lo spegnimento automatico per l'agente di lavoro dopo che il software è stato distribuito e ha funzionato come previsto. Ciò consente alla flotta di lavoratori di espandersi quando necessario e di spegnersi al termine di un lavoro. La scalabilità automatica aiuta a garantire l'utilizzo solo delle risorse necessarie. Per consentire la chiusura di un'istanza avviata dal gruppo auto scaling, è necessario aggiungerla `shutdown_on_stop=true` al file di `worker.toml` configurazione.

Per abilitare lo spegnimento automatico

Come utente: **`root`**

- Installa l'agente di lavoro con i parametri**`--allow-shutdown`**.

  Linux

  Inserisci:

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

  Windows

  Inserisci:

  ```
  install-deadline-worker ^
    --farm-id FARM_ID ^
    --fleet-id FLEET_ID ^
    --region REGION ^
    --allow-shutdown
  ```

Crea utenti e gruppi di lavoro

Questa sezione descrive la relazione utente e di gruppo richiesta tra l'utente agente e quella `jobRunAsUser` definita nelle code.

Il worker agent di Deadline Cloud deve funzionare come utente dedicato specifico dell'agente sull'host. È necessario configurare la `jobRunAsUser` proprietà delle code di Deadline Cloud in modo

che i lavoratori eseguano i lavori in coda come utenti e gruppi specifici del sistema operativo. Questa configurazione significa che puoi controllare le autorizzazioni condivise del file system dei tuoi lavori. Fornisce inoltre un importante limite di sicurezza tra i lavori e l'utente worker agent.

Linuxutenti e gruppi di lavoro

Per configurare un utente Worker Agent locale e `jobRunAsUser` assicurarsi di soddisfare i seguenti requisiti. Se si utilizza un Linux Pluggable Authentication Module (PAM) come Active Directory o LDAP, la procedura potrebbe essere diversa.

L'utente del worker agent e il `jobRunAsUser` gruppo condiviso vengono impostati al momento dell'installazione del worker agent. Le impostazioni predefinite sono `deadline-worker-agent` e`deadline-job-users`, ma è possibile modificarle quando si installa il worker agent.

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

I comandi devono essere eseguiti come utente root.

- Ciascuno `jobRunAsUser` dovrebbe avere un gruppo primario corrispondente. La creazione di un utente con il `adduser` comando di solito crea un gruppo primario corrispondente.

  ```
  adduser -r -m jobRunAsUser
  ```

- Il gruppo principale di `jobRunAsUser` è un gruppo secondario per l'utente del worker agent. Il gruppo condiviso consente all'agente di lavoro di rendere disponibili i file al lavoro mentre è in esecuzione.

  ```
  usermod -a -G jobRunAsUser deadline-worker-agent
  ```

- `jobRunAsUser`Deve essere un membro del gruppo di lavoro condiviso.

  ```
  usermod -a -G deadline-job-users jobRunAsUser
  ```

- Non `jobRunAsUser` deve appartenere al gruppo principale dell'utente del worker agent. I file sensibili scritti dal worker agent sono di proprietà del gruppo principale dell'agente. Se a `jobRunAsUser` fa parte di questo gruppo, i file del worker agent possono essere accessibili ai job in esecuzione sul worker.

- L'impostazione predefinita Regione AWS deve corrispondere alla regione dell'azienda agricola a cui appartiene il lavoratore. Questo dovrebbe essere applicato a tutti gli `jobRunAsUser` account del lavoratore.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

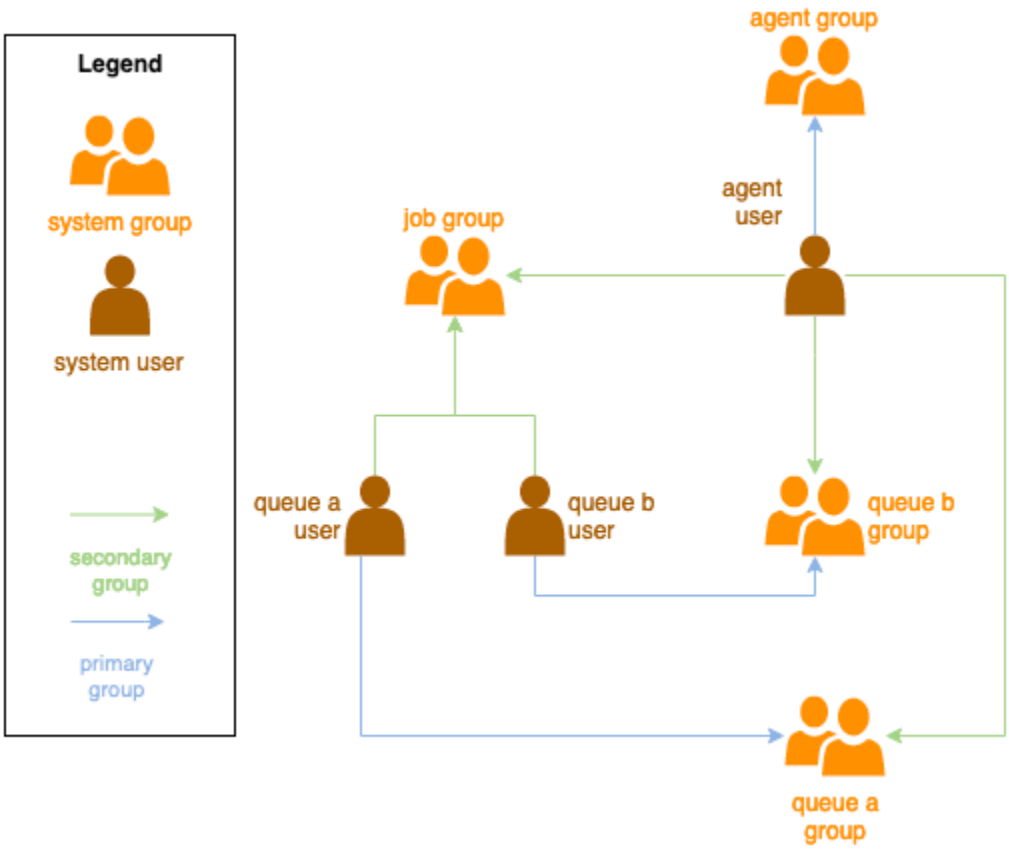
- L'utente del worker agent deve essere in grado di eseguire `sudo` comandi come`jobRunAsUser`. Esegui il comando seguente per aprire un editor e creare una nuova regola sudoers:

```
visudo -f /etc/sudoers.d/deadline-worker-job-user
```

  Aggiungi quanto segue al file:

```
# Allows the Deadline Cloud worker agent OS user to run commands
 # as the queue OS user without requiring a password.
deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
```

Il diagramma seguente illustra la relazione tra l'utente agente e `jobRunAsUser` gli utenti e i gruppi per le code associate alla flotta.



Utenti Windows

Per utilizzare un Windows utente come`jobRunAsUser`, deve soddisfare i seguenti requisiti:

- Tutti gli `jobRunAsUser` utenti della coda devono esistere.
- Le loro password devono corrispondere al valore del segreto specificato nel campo della coda. `JobRunAsUser` Per istruzioni, consulta il passaggio 7 delle code di Deadline Cloud nella Guida per l'utente di AWS Deadline Cloud.
- L'utente-agente deve essere in grado di accedere come tali utenti.

Protezione dell'host di lavoro

Quando configuri il tuo worker host, segui le migliori pratiche di sicurezza per proteggere le informazioni sensibili e mantenere controlli di accesso adeguati.

Configurazione delle autorizzazioni per le cartelle di registro

L'agente di lavoro scrive file di registro che possono contenere informazioni riservate provenienti dagli script di configurazione dell'host e dall'esecuzione del lavoro. Il `install-deadline-worker` comando crea la directory di log con autorizzazioni sicure. Se è necessario creare la directory manualmente prima dell'installazione, utilizzare le seguenti procedure per abbinare le autorizzazioni utilizzate dalle flotte gestite dai servizi:

Linux

Per configurare le autorizzazioni della directory di registro su Linux

1. Crea la directory dei log:

```
sudo mkdir -p /var/log/amazon/deadline
```

2. Imposta il proprietario e il gruppo come utente worker agent:

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. Imposta le autorizzazioni su 750:

```
sudo chmod -R 750 /var/log/amazon/deadline
```

Queste autorizzazioni assicurano che solo l'utente e il gruppo worker agent possano accedere ai file di registro, impedendo agli utenti del lavoro e ad altri utenti non autorizzati di leggere informazioni potenzialmente sensibili.

Windows

Per configurare le autorizzazioni della directory di registro su Windows

1. Aprire un PowerShell terminale per amministratori.
2. Crea la directory dei log:

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. Configura con restrizioni ACLs per consentire solo l'utente del worker agent e gli amministratori:

```
$acl = Get-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs"
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
$agentRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("deadline-worker",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$adminRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("Administrators",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$acl.AddAccessRule($agentRule)
$acl.AddAccessRule($adminRule)
Set-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs" $acl
```

Questi comandi limitano l'accesso alla directory dei log solo all'utente del worker agent e al gruppo Administrators, impedendo agli utenti del job e ad altri utenti non autorizzati di leggere informazioni potenzialmente riservate.

Gestisci l'accesso ai segreti degli utenti del Windows lavoro

Quando si configura una coda con un Windows`jobRunAsUser`, è necessario specificare un segreto di AWS Secrets Manager. Il valore di questo segreto dovrebbe essere un oggetto del modulo con codifica JSON:

```
{
    "password": "JOB_USER_PASSWORD"
}
```

Affinché Workers esegua i job secondo la configurazione della coda`jobRunAsUser`, il ruolo IAM della flotta deve disporre delle autorizzazioni necessarie per ottenere il valore del segreto. Se il segreto è crittografato utilizzando una chiave KMS gestita dal cliente, anche il ruolo IAM della flotta deve disporre delle autorizzazioni per la decrittografia utilizzando la chiave KMS.

Si consiglia vivamente di seguire il principio del privilegio minimo per questi segreti. Ciò significa che l'accesso per recuperare il valore segreto di → → di una coda dovrebbe essere: `jobRunAsUser` `windows` `passwordArn`

- concesso a un ruolo di flotta quando viene creata un'associazione queue-fleet tra la flotta e la coda

- revocato da un ruolo di flotta quando viene eliminata un'associazione queue-fleet tra la flotta e la coda

Inoltre, il segreto di AWS Secrets Manager contenente la `jobRunAsUser` password deve essere eliminato quando non viene più utilizzato.

Concedi l'accesso a una password segreta

Le flotte Deadline Cloud richiedono l'accesso alla `jobRunAsUser` password memorizzata nella password segreta della coda quando coda e flotta sono associate. Ti consigliamo di utilizzare la politica delle risorse di AWS Secrets Manager per concedere l'accesso ai ruoli della flotta. Se ti attieni rigorosamente a queste linee guida, è più facile determinare quali ruoli della flotta hanno accesso al segreto.

Per concedere l'accesso al segreto

1. Apri la console AWS Secret Manager al segreto.
2. Nella sezione Autorizzazioni delle risorse, aggiungi una dichiarazione politica nel modulo:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

Revoca l'accesso a una password segreta

Quando una flotta non richiede più l'accesso a una coda, rimuovi l'accesso alla password segreta per la coda. `jobRunAsUser` Ti consigliamo di utilizzare la politica delle risorse di AWS Secrets Manager per concedere l'accesso ai ruoli della flotta. Se ti attieni rigorosamente a queste linee guida, è più facile determinare quali ruoli della flotta hanno accesso al segreto.

Per revocare l'accesso al segreto

1. Apri la console AWS Secret Manager al segreto.
2. Nella sezione Autorizzazioni delle risorse, rimuovi la dichiarazione politica del modulo:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action" : [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

Installa e configura il software necessario per i lavori

Dopo aver configurato l'agente di lavoro di Deadline Cloud, puoi preparare l'host di lavoro con qualsiasi software necessario per eseguire i lavori.

Quando invii un lavoro a una coda con un lavoro associato`jobRunAsUser`, il lavoro viene eseguito come tale utente. Quando un lavoro viene inviato con comandi che non sono un percorso assoluto, quel comando deve essere trovato nella cartella PATH di quell'utente.

In Linux, è possibile specificare PATH per un utente in uno dei seguenti modi:

- loro ~/.bashrc o ~/.bash_profile
- file di configurazione del sistema come /etc/profile.d/* e /etc/profile
- script di avvio della shell:/etc/bashrc.

In Windows, è possibile specificare PATH per un utente in uno dei seguenti modi:

- le loro variabili di ambiente specifiche dell'utente
- le variabili di ambiente a livello di sistema

Installa gli adattatori per strumenti di creazione di contenuti digitali

Deadline Cloud fornisce OpenJobDescription adattatori per l'utilizzo delle più diffuse applicazioni DCC (Digital Content Creation). Per utilizzare questi adattatori in una flotta gestita dal cliente, è necessario installare il software DCC e gli adattatori per applicazioni. Quindi, assicuratevi che i programmi eseguibili del software siano disponibili nel percorso di ricerca del sistema (ad esempio, nella variabile di ambiente). PATH

Per installare adattatori DCC su una flotta gestita dal cliente

1. Apri il terminale a.
    a. Su Linux, apri un terminale come root utente (o usasudo/su)
    b. AttivatoWindows, apri il prompt dei comandi o il PowerShell terminale dell'amministratore.
2. Installa i pacchetti di adattatori Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

Configurazione delle credenziali AWS

La fase iniziale del ciclo di vita del lavoratore è il bootstrap. In questa fase, il software Worker Agent crea un lavoratore nella vostra flotta e ottiene AWS le credenziali del ruolo del parco macchine per ulteriori operazioni.

AWS credentials for Amazon EC2

Per creare un ruolo IAM per Amazon EC2 con le autorizzazioni degli host worker di Deadline Cloud

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, scegli Ruoli nel riquadro di navigazione, quindi scegli Crea ruolo.
3. Seleziona AWS servizio.
4. Seleziona EC2 come servizio o caso d'uso, quindi seleziona Avanti.
5. Per concedere le autorizzazioni necessarie, allega la politica `AWSDeadlineCloud-WorkerHost` AWS gestita.

On-premises AWS credentials

I tuoi dipendenti locali utilizzano le credenziali per accedere a Deadline Cloud. Per un accesso più sicuro, ti consigliamo di utilizzare IAM Roles Anywhere per autenticare i tuoi lavoratori. Per ulteriori informazioni, consulta IAM Roles Anywhere.

Per i test, puoi utilizzare le chiavi di accesso utente IAM per le AWS credenziali. Ti consigliamo di impostare una scadenza per l'utente IAM includendo una policy in linea restrittiva.

> **Important**
>
> Presta attenzione ai seguenti avvertimenti:
>
> - NON utilizzare le credenziali root del tuo account per accedere alle risorse. AWS Queste credenziali forniscono un accesso illimitato all'account e sono difficili da revocare.
> - NON inserire chiavi di accesso letterali o informazioni sulle credenziali nei file dell'applicazione. In caso contrario, rischi di esporre accidentalmente le credenziali se, per esempio, carichi il progetto in repository pubblici.
> - NON includere file che contengono credenziali nell'area del progetto.
> - Proteggi le tue chiavi di accesso. Non fornire le chiavi di accesso a parti non autorizzate, neppure per contribuire a trovare gli identificatori di account. In questo modo, potresti concedere a qualcuno l'accesso permanente al tuo account.

- Tieni presente che tutte le credenziali archiviate nel file delle AWS credenziali condivise vengono archiviate in testo semplice.

Per maggiori dettagli, consulta le migliori pratiche per la gestione delle chiavi di AWS accesso nella Guida AWS generale.

Creare un utente IAM

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, seleziona Utenti, quindi seleziona Crea utente.
3. Assegna un nome all'utente. Deseleziona la casella di controllo Fornisci l'accesso utente a Console di gestione AWS, quindi scegli Avanti.
4. Scegli Allega direttamente le politiche.
5. Dall'elenco delle politiche di autorizzazione, scegli la AWSDeadlineCloud-WorkerHostpolitica, quindi scegli Avanti.
6. Controlla i dettagli dell'utente, quindi scegli Crea utente.

Limitare l’accesso dell’utente a una finestra a tempo limitato

Le chiavi di accesso create per un utente IAM sono credenziali a lungo termine. Per garantire che queste credenziali scadano nel caso vengano utilizzate impropriamente, puoi assegnarvi un limite di tempo creando una policy in linea che specifichi una data dopo la quale le chiavi non saranno più valide.

1. Apri l’utente IAM appena creato. Nella scheda Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Crea politica in linea.
2. Nell’editor JSON, specifica le seguenti autorizzazioni. Per utilizzare questa politica, sostituisci il valore del `aws:CurrentTime` timestamp nella politica di esempio con la tua ora e data.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Creare una chiave di accesso

1. Nella pagina dei dettagli dell'utente, seleziona la scheda Credenziali di sicurezza. Nella sezione Chiavi di accesso, scegliere Crea chiave di accesso.
2. Indica che desideri utilizzare la chiave per Altro, quindi scegli Avanti, quindi scegli Crea chiave di accesso.
3. Nella pagina Recupera chiavi di accesso, scegli Mostra per rivelare il valore della chiave di accesso segreta dell'utente. Puoi copiare le credenziali o scaricare un file .csv.

Memorizza le chiavi di accesso dell'utente

- Memorizza le chiavi di accesso dell'utente nel file delle AWS credenziali dell'utente agente sul sistema host di lavoro:
  - SìLinux, il file si trova in `~/.aws/credentials`
  - WindowsAttivo, il file si trova in `%USERPROFILE\.aws\credentials`

  Sostituisci i seguenti tasti:

  ```
  [default]
  aws_access_key_id=ACCESS_KEY_ID
  aws_secret_access_key=SECRET_ACCESS_KEY
  ```

> **Important**
>
> Quando non hai più bisogno di questo utente IAM, ti consigliamo di rimuoverlo per allinearlo alle best practice AWS di sicurezza. Ti consigliamo di richiedere agli utenti umani di utilizzare credenziali temporanee AWS IAM Identity Centerdurante l'accesso. AWS

Flusso di dati host dei lavoratori per flotte gestite dai clienti

Questo argomento descrive le connessioni di rete che gli host dei worker di AWS Deadline Cloud (Deadline Cloud) effettuano durante il funzionamento, inclusi gli endpoint contattati, i protocolli utilizzati e i dati trasmessi. Queste informazioni si applicano ai lavoratori della flotta gestita dai clienti (CMF), inclusi sia le istanze Amazon Elastic Compute Cloud (Amazon EC2) che i lavoratori locali. Utilizza queste informazioni per configurare le regole del firewall, creare endpoint VPC, eseguire controlli di sicurezza o pianificare le politiche di rete per i tuoi host di lavoro. Per informazioni sulle reti di flotte gestite dai servizi, vedere. Inter-network privacy del traffico

Tutte le comunicazioni con i lavoratori sono solo in uscita. Gli host di lavoro avviano tutte le connessioni: non è necessario consentire alcuna connessione in entrata. Tutte le connessioni utilizzano HTTPS (TLS 1.2 o versione successiva) sulla porta 443.

Questo argomento include le seguenti sezioni:

- the section called “Endpoint e protocolli”
- the section called “Operazioni API utilizzate dai lavoratori”
- the section called “Altri dati trasmessi”
- the section called “Opzioni di connettività privata”

Endpoint e protocolli

La tabella seguente elenca gli endpoint AWS di servizio a cui gli host dei lavoratori si connettono durante il funzionamento. Per l'elenco completo degli endpoint regionali per ogni servizio, consulta gli endpoint e le quote del servizio nella Guida generale.AWS

Riferimento all'endpoint Worker Host

AWS servizio	Endpoint	Porta/ Protocollo	Scopo	Richiesto
Deadline Cloud (pianificazione)	`scheduling.deadline. [Region].amazonaws.com`	443/ HTTPS	Registrazione dei lavoratori, sondaggi sulle attività, aggiornamenti dello stato, scambio di credenziali, recupero dell'entità lavorativa. Per informazioni, consulta the section called “Operazioni API utilizzate dai lavoratori”.	Sempre
CloudWatch Registri Amazon (CloudWatch registri)	`logs.[Region].amazonaws.com`	443/ HTTPS	Agente di lavoro e consegna del registro delle sessioni.	Sempre
Amazon Simple Storage Service (Amazon S3)	`s3.[Region].amazonaws.com`	443/ HTTPS	Caricamento e download degli allegati Job.	Se si utilizzano allegati di lavoro

Se i tuoi lavori utilizzano altri AWS servizi, potresti dover consentire anche le connessioni in uscita a tali endpoint di servizio.

Operazioni API utilizzate dai lavoratori

Tutte le seguenti operazioni API utilizzano l'`scheduling.deadline.[Region].amazonaws.com`endpoint. Per gli schemi completi di richiesta e risposta di ciascuna operazione, consulta il Deadline Cloud API Reference.

Fase Bootstrap

All'avvio di un worker host, l'agente lavoratore si registra con la flotta. Le credenziali di bootstrap richiedono le autorizzazioni previste nella policy `AWSDeadlineCloud-WorkerHost` AWS gestita o autorizzazioni personalizzate equivalenti. La fase di bootstrap utilizza le seguenti operazioni API:

- `CreateWorker`— Registra il lavoratore nella flotta. Invia il nome host e gli indirizzi IP. Riceve un ID lavoratore.
- `AssumeFleetRoleForWorker`— Ottiene le credenziali del ruolo della flotta. Riceve AWS le credenziali temporanee utilizzate dall'agente di lavoro per le operazioni successive.

Fase operativa

Dopo il bootstrap, l'agente di lavoro effettua sondaggi per le sessioni di lavoro e di elaborazione. Il ruolo Fleet richiede le autorizzazioni previste nella policy `AWSDeadlineCloud-FleetWorker` AWS gestita o autorizzazioni personalizzate equivalenti e utilizza le seguenti operazioni API:

- `UpdateWorker`— Aggiorna lo stato del lavoratore, ad esempio `STOPPED` durante lo spegnimento.
- `UpdateWorkerSchedule`— Sondaggi per incarichi di lavoro. Invia aggiornamenti sullo stato delle azioni della sessione, tra cui lo stato di completamento, la percentuale di avanzamento, il messaggio di avanzamento e gli hash del manifesto di output. Riceve le sessioni assegnate (ID del lavoro, ID della coda, azioni di sessione, configurazione del registro), le richieste di annullamento, lo stato del lavoratore desiderato e l'intervallo di aggiornamento.
- `BatchGetJobEntity`— Recupera i dettagli del lavoro assegnato. Invia gli identificatori delle entità lavorative. Riceve dettagli sul lavoro, dettagli sull'ambiente e dettagli sugli allegati del lavoro.
- `AssumeFleetRoleForWorker`— Aggiorna periodicamente le credenziali del ruolo del parco veicoli.
- `AssumeQueueRoleForWorker`— Ottiene le credenziali del ruolo di coda limitate a una coda specifica. Il lavoratore utilizza queste credenziali per accedere agli allegati del lavoro in Amazon S3.

Altri dati trasmessi

Oltre alle operazioni dell'API di pianificazione di Deadline Cloud, gli host di lavoro trasmettono i seguenti dati ad altri AWS servizi:

Dati di registro

L'agente di lavoro invia i registri degli agenti di lavoro e i registri delle sessioni (stdout e stderr dai processi di lavoro) ai CloudWatch registri utilizzando l'operazione API. `PutLogEvents`

Allegati Job

I lavoratori trasferiscono i file di input e output tramite Amazon S3 utilizzando operazioni `GetObject` `PutObject` API. Il lavoratore utilizza le credenziali del ruolo di coda ottenute tramite `AssumeQueueRoleForWorker` questo accesso.

Telemetria (opzionale)

L'agente di lavoro invia metriche operative come rapporti sugli arresti anomali. Puoi disattivare la raccolta di dati di telemetria. Per ulteriori informazioni, consulta Disattivazione.

Opzioni di connettività privata

Puoi utilizzarlo AWS PrivateLink per mantenere il traffico tra gli host dei lavoratori CMF e Deadline Cloud all'interno del tuo VPC, senza attraversare la rete Internet pubblica. Per i lavoratori locali, è possibile combinare AWS PrivateLink con AWS Direct Connect ()Direct Connect o una connessione VPN. Per ulteriori informazioni, consulta Accesso AWS Deadline Cloud utilizzando un endpoint di interfaccia (AWS PrivateLink).

Verifica la configurazione del tuo host di lavoro

Dopo aver installato l'agente di lavoro, installato il software necessario per elaborare i lavori e configurato AWS le credenziali per l'agente di lavoro, è necessario verificare che l'installazione sia in grado di elaborare i lavori prima di creare un AMI per la tua flotta. Dovresti testare quanto segue:

- L'agente di lavoro Deadline Cloud è configurato correttamente per essere eseguito come servizio di sistema.
- Che il lavoratore interroghi la coda associata per il lavoro.
- Che il lavoratore elabori con successo i lavori inviati alla coda associata al parco macchine.

Dopo aver testato la configurazione e aver elaborato correttamente i lavori rappresentativi, è possibile utilizzare il lavoratore configurato per creare un AMI per EC2 i lavoratori di Amazon o come modello per i lavoratori locali.

> **Note**
>
> Se state testando la configurazione worker host di una flotta con scalabilità automatica, potreste avere difficoltà a testare il vostro lavoratore nelle seguenti situazioni:
>
> - Se non c'è lavoro in coda, Deadline Cloud arresta il worker agent poco dopo l'inizio del lavoratore.
> - Se l'agente di lavoro è configurato per spegnere l'host al momento dell'arresto, spegne la macchina se non c'è lavoro in coda.
>
> Per evitare questi problemi, utilizza una flotta di staging che non sia scalabile automaticamente per configurare e testare i tuoi lavoratori. Dopo aver testato il worker host, assicurati di impostare l'ID corretto della flotta prima di preparare un AMI.

Per testare la configurazione del tuo host di lavoro

1. Esegui l'agente di lavoro avviando il servizio del sistema operativo.

   Linux

   Da una shell root esegui il seguente comando:

   ```
   systemctl start deadline-worker
   ```

   Windows

   Dal prompt dei comandi dell'amministratore o PowerShell terminale, immettete il seguente comando:

   ```
   sc.exe start DeadlineWorker
   ```

2. Monitora il lavoratore per assicurarti che parta ed esegua i sondaggi per sapere se è al lavoro.

   Linux

   Da una shell root esegui il seguente comando:

   ```
   systemctl status deadline-worker
   ```

Il comando dovrebbe restituire una risposta del tipo:

```
Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
```

Se la risposta non è così, ispeziona il file di registro usando il seguente comando:

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

Dal prompt dei comandi dell'amministratore o PowerShell terminale, immettete il seguente comando:

```
sc.exe query DeadlineWorker
```

Il comando dovrebbe restituire una risposta del tipo:

```
STATE   : 4 RUNNING
```

Se la risposta non lo contieneRUNNING, ispeziona il file di registro del lavoratore. Apri e amministra PowerShell richiedi ed esegui il seguente comando:

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. Invia i lavori alla coda associata alla tua flotta. I lavori devono essere rappresentativi dei lavori svolti dalla flotta.
4. Monitora lo stato di avanzamento del lavoro utilizzando il monitor Deadline Cloud o la CLI. Se un lavoro fallisce, controlla i registri della sessione e dei lavoratori.
5. Aggiorna la configurazione dell'host di lavoro secondo necessità fino al completamento corretto dei lavori.
6. Quando i lavori di test hanno esito positivo, puoi fermare il lavoratore:

   Linux

   Da una shell root esegui il seguente comando:

```
systemctl stop deadline-worker
```

Windows

Dal prompt dei comandi dell'amministratore o PowerShell terminale, immettete il seguente comando:

```
sc.exe stop DeadlineWorker
```

Crea un Amazon Machine Image

Per creare un Amazon Machine Image (AMI) per utilizzarlo in una flotta Amazon Elastic Compute Cloud (Amazon EC2) gestita dai clienti (CMF), completa le attività in questa sezione. È necessario creare un' EC2 istanza Amazon prima di procedere. Per ulteriori informazioni, consulta Launch your istance nella Amazon EC2 User Guide for Linux Instances.

> **Important**
>
> Creare un AMI crea un'istantanea dei volumi collegati all' EC2 istanza Amazon. Qualsiasi software installato sull'istanza persiste, così come le istanze, che vengono riutilizzate quando si avviano istanze da AMI. Consigliamo di adottare una strategia di patch e di aggiornare regolarmente le nuove AMI con un software aggiornato prima di iscriverti alla tua flotta.

Preparare l' EC2 istanza Amazon

Prima di creare un AMI, è necessario eliminare lo stato del lavoratore. Lo stato del lavoratore persiste tra un avvio e l'altro del worker agent. Se questo stato persiste nel AMI, quindi tutte le istanze avviate da esso condivideranno lo stesso stato.

Ti consigliamo inoltre di eliminare tutti i file di registro esistenti. I file di log possono rimanere su un' EC2istanza Amazon durante la preparazione dell'AMI. L'eliminazione di questi file riduce al minimo la confusione durante la diagnosi di possibili problemi nelle flotte di lavoratori che utilizzano l'AMI.

Dovresti anche abilitare il servizio di sistema worker agent in modo che il worker agent di Deadline Cloud EC2 si avvii all'avvio di Amazon.

Infine, ti consigliamo di abilitare lo spegnimento automatico dell'agente di lavoro. Ciò consente alla flotta di lavoratori di ampliarsi quando necessario e di interrompersi al termine del lavoro di rendering. Questa scalabilità automatica aiuta a garantire l'utilizzo delle risorse solo se necessario.

Per preparare l' EC2 istanza Amazon

1. Apri la EC2 console Amazon.
2. Avvia un' EC2 istanza Amazon. Per ulteriori informazioni, consulta Launch your istance.
3. Configura l'host per la connessione al tuo provider di identità (IdP), quindi monta qualsiasi file system condiviso di cui ha bisogno.
4. Segui i tutorial per, quindi, e. Installa l'agente di lavoro Deadline Cloud Configura l'agente di lavoro Crea utenti e gruppi di lavoro
5. Se stai preparando un AMI basato su Amazon Linux 2023 per eseguire software compatibile con la piattaforma di riferimento VFX, è necessario aggiornare diversi requisiti. Per informazioni, consulta la compatibilità della piattaforma di riferimento VFX nella Guida per l'utente di AWS Deadline Cloud.
6. Apri un terminale.
   a. Su Linux, apri un terminale come `root` utente (o usa`sudo`/) `su`
   b. Abilitato Windows, apri il prompt dei comandi o il PowerShell terminale dell'amministratore.
7. Assicurati che il servizio di lavoro non sia in esecuzione e configurato per l'avvio all'avvio:
   a. Su Linux, esegui

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. Abilitato Windows, esegui

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. Eliminare lo stato del lavoratore.
   a. Su Linux, esegui

```
rm -rf /var/lib/deadline/*
```

   b. Abilitato Windows, esegui

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Eliminare i file di registro.

   a. Su Linux, esegui

   ```
   rm -rf /var/log/amazon/deadline/*
   ```

   b. Abilitato Windows, esegui

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
   ```

10. Abilitato Windows, si consiglia di eseguire l'applicazione Amazon EC2 Launch Settings disponibile nel menu Start per completare la preparazione finale dell'host e lo spegnimento dell'istanza.

> **Note**
>
> DEVI scegliere Shutdown senza Sysprep e non scegliere mai Shutdown with Sysprep. L'arresto con Sysprep renderà inutilizzabili tutti gli utenti locali. Per ulteriori informazioni, consulta la sezione Prima di iniziare dell'argomento Creare un'AMI personalizzata della Guida dell'utente per le istanze di Windows.

Costruisci il AMI

Per costruire il AMI

1. Apri la EC2 console Amazon.
2. Seleziona Istanze nel riquadro di navigazione, quindi seleziona la tua istanza.
3. Scegli lo stato dell'istanza, quindi Arresta l'istanza.
4. Dopo che l'istanza è stata interrotta, scegli Azioni.
5. Scegli Immagine e modelli, quindi Crea immagine.
6. Inserisci un nome per l'immagine.
7. (Facoltativo) Inserisci una descrizione per l'immagine.
8. Scegliere Create Image (Crea immagine).

Crea un'infrastruttura per il parco veicoli con un gruppo Amazon EC2 Auto Scaling

Questa sezione spiega come creare una flotta Amazon EC2 Auto Scaling.

Utilizza il modello CloudFormation YAML riportato di seguito per creare un gruppo Amazon EC2 Auto Scaling (Auto Scaling), un Amazon Virtual Private Cloud (Amazon VPC) con due sottoreti, un profilo di istanza e un ruolo di accesso all'istanza. Questi sono necessari per avviare l'istanza utilizzando Auto Scaling nelle sottoreti.

È necessario rivedere e aggiornare l'elenco dei tipi di istanze per adattarlo alle proprie esigenze di rendering.

Per una spiegazione completa delle risorse e dei parametri utilizzati nel modello CloudFormation YAML, consulta il riferimento al tipo di risorsa Deadline Cloud nella Guida per l'AWS CloudFormation utente.

Per creare una flotta Amazon EC2 Auto Scaling

1. Usa l'esempio seguente per creare un CloudFormation modello che definisca i parametri `FarmIDFleetID`, e`AMIId`. Salvate il modello in un `.YAML` file sul computer locale.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
```

```
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
    DependsOn:
      - deadlineIGW
      - deadlineVPCGatewayAttachment
  deadlinePublicSubnet0:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.16.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - a
```

```
  deadlineSubnetRouteTableAssociation0:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
  deadlinePublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.20.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - c
  deadlineSubnetRouteTableAssociation1:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet1
  deadlineInstanceAccessAccessRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
        - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
```

```
      - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
        MetadataOptions:
          HttpTokens: required
          HttpEndpoint: enabled

  deadlineAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      AutoScalingGroupName: !Join
        - ''
        - - deadline-ASG-autoscalable-
          - !Ref FleetId
      MinSize: 0
      MaxSize: 10
      VPCZoneIdentifier:
        - !Ref deadlinePublicSubnet0
        - !Ref deadlinePublicSubnet1
      NewInstancesProtectedFromScaleIn: true
      MixedInstancesPolicy:
        InstancesDistribution:
          OnDemandBaseCapacity: 0
          OnDemandPercentageAboveBaseCapacity: 0
          SpotAllocationStrategy: capacity-optimized
          OnDemandAllocationStrategy: lowest-price
```

```
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref deadlineLaunchTemplate
            Version: !GetAtt
              - deadlineLaunchTemplate
              - LatestVersionNumber
          Overrides:
            - InstanceType: m5.large
            - InstanceType: m5d.large
            - InstanceType: m5a.large
            - InstanceType: m5ad.large
            - InstanceType: m5n.large
            - InstanceType: m5dn.large
            - InstanceType: m4.large
            - InstanceType: m3.large
            - InstanceType: r5.large
            - InstanceType: r5d.large
            - InstanceType: r5a.large
            - InstanceType: r5ad.large
            - InstanceType: r5n.large
            - InstanceType: r5dn.large
            - InstanceType: r4.large
      MetricsCollection:
        - Granularity: 1Minute
          Metrics:
            - GroupMinSize
            - GroupMaxSize
            - GroupDesiredCapacity
            - GroupInServiceInstances
            - GroupTotalInstances
            - GroupInServiceCapacity
            - GroupTotalCapacity
```

2. Apri la CloudFormation console in https://console.aws.amazon.com/cloudformation.

   Usa la CloudFormation console per creare uno stack utilizzando le istruzioni per caricare il file modello che hai creato. Per ulteriori informazioni, consulta Creazione di uno stack sulla CloudFormation console nella Guida per l'AWS CloudFormation utente.

Note

- Le credenziali del ruolo IAM collegate all'istanza Amazon EC2 del lavoratore sono disponibili per tutti i processi in esecuzione su quel lavoratore, compresi i job. Il lavoratore deve avere i privilegi minimi per operare: e `deadline:CreateWorker` `deadline:AssumeFleetRoleForWorker.`
- L'agente di lavoro ottiene le credenziali per il ruolo di coda e le configura per l'utilizzo da parte dell'esecuzione dei job. Il ruolo del profilo dell'istanza Amazon EC2 non dovrebbe includere le autorizzazioni necessarie per i tuoi lavori.

Ridimensiona automaticamente la tua flotta Amazon EC2 con la funzionalità di raccomandazione di scalabilità Deadline Cloud

Deadline Cloud sfrutta un gruppo Amazon EC2 Auto Scaling (Auto Scaling) per scalare automaticamente la flotta gestita dai clienti (CMF) di Amazon EC2. Devi configurare la modalità flotta e implementare l'infrastruttura richiesta nel tuo account per far sì che la tua flotta si ridimensioni automaticamente. L'infrastruttura che hai implementato funzionerà per tutte le flotte, quindi dovrai configurarla una sola volta.

Il flusso di lavoro di base è: configuri la modalità flotta per la scalabilità automatica, quindi Deadline Cloud invierà un EventBridge evento per quella flotta ogni volta che la dimensione della flotta consigliata cambia (un evento contiene l'ID della flotta, la dimensione della flotta consigliata e altri metadati). Avrai una EventBridge regola per filtrare gli eventi rilevanti e avrai una Lambda per consumarli. Lambda si integrerà con Amazon EC2 Auto `AutoScalingGroup` Scaling per scalare automaticamente la flotta Amazon EC2.

Imposta la modalità flotta su **`EVENT_BASED_AUTO_SCALING`**

Configura la modalità flotta su`EVENT_BASED_AUTO_SCALING`. Puoi utilizzare la console per eseguire questa operazione oppure utilizzare la AWS CLI per chiamare direttamente l'`UpdateFleet`API `CreateFleet` or. Dopo aver configurato la modalità, Deadline Cloud inizia a inviare EventBridge eventi ogni volta che la dimensione della flotta consigliata cambia.

- `UpdateFleet`Comando di esempio:

```
aws deadline update-fleet \
```

```
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- `CreateFleet`Comando di esempio:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

Di seguito è riportato un esempio di `configuration.json` utilizzo dei comandi CLI precedenti (`--configuration file://configuration.json`.

- Per abilitare Auto Scaling sulla tua flotta, devi impostare la modalità su. `EVENT_BASED_AUTO_SCALING`
- `workerCapabilities`Sono i valori predefiniti assegnati al CMF al momento della sua creazione. È possibile modificare questi valori se è necessario aumentare le risorse disponibili per il CMF.

Dopo aver configurato la modalità flotta, Deadline Cloud inizia a emettere eventi di raccomandazione sulle dimensioni della flotta per quella flotta.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

Implementa lo stack Auto Scaling utilizzando il modello CloudFormation

Puoi impostare una EventBridge regola per filtrare gli eventi, una Lambda per consumare gli eventi e controllare l'Auto Scaling e una coda SQS per archiviare gli eventi non elaborati. Usa il seguente CloudFormation modello per distribuire tutto in uno stack. Dopo aver distribuito correttamente le risorse, puoi inviare un lavoro e la flotta aumenterà automaticamente.

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
              "source": "aws.deadline",
              "account": "111122223333",
              "time": "2017-12-22T18:43:48Z",
              "region": "us-west-1",
              "resources": [],
              "detail": {
                  "farmId": "farm-12345678900000000000000000000000",
                  "fleetId": "fleet-12345678900000000000000000000000",
                  "oldFleetSize": 1,
                  "newFleetSize": 5,
              }
          }
          """

          import json
          import boto3
          import logging

          logger = logging.getLogger()
```

```
          logger.setLevel(logging.INFO)

          auto_scaling_client = boto3.client("autoscaling")

          def lambda_handler(event, context):
              logger.info(event)
              event_detail = event["detail"]
              fleet_id = event_detail["fleetId"]
              desired_capacity = event_detail["newFleetSize"]

              asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
              auto_scaling_client.set_desired_capacity(
                  AutoScalingGroupName=asg_name,
                  DesiredCapacity=desired_capacity,
                  HonorCooldown=False,
              )

              return {
                  'statusCode': 200,
                  'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
              }
      Handler: index.lambda_handler
      Role: !GetAtt
        - AutoScalingLambdaServiceRole
        - Arn
      Runtime: python3.11
    DependsOn:
      - AutoScalingLambdaServiceRoleDefaultPolicy
      - AutoScalingLambdaServiceRole
  AutoScalingEventRule:
    Type: 'AWS::Events::Rule'
    Properties:
      EventPattern:
        source:
          - aws.deadline
        detail-type:
          - Fleet Size Recommendation Change
      State: ENABLED
      Targets:
        - Arn: !GetAtt
            - AutoScalingLambda
            - Arn
          DeadLetterConfig:
```

```
          Arn: !GetAtt
            - UnprocessedAutoScalingEventQueue
            - Arn
        Id: Target0
        RetryPolicy:
          MaximumRetryAttempts: 15
AutoScalingEventRuleTargetPermission:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !GetAtt
      - AutoScalingLambda
      - Arn
    Principal: events.amazonaws.com
    SourceArn: !GetAtt
      - AutoScalingEventRule
      - Arn
AutoScalingLambdaServiceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: 'sts:AssumeRole'
          Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
      Version: 2012-10-17
    ManagedPolicyArns:
      - !Join
        - ''
        - - 'arn:'
          - !Ref 'AWS::Partition'
          - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
AutoScalingLambdaServiceRoleDefaultPolicy:
  Type: 'AWS::IAM::Policy'
  Properties:
    PolicyDocument:
      Statement:
        - Action: 'autoscaling:SetDesiredCapacity'
          Effect: Allow
          Resource: '*'
      Version: 2012-10-17
    PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
    Roles:
```

```
      - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

Esegui un controllo dello stato della flotta

Dopo aver creato la tua flotta, dovresti creare un controllo sanitario personalizzato per assicurarti che la flotta rimanga integra e priva di istanze bloccate per evitare costi inutili. Vedi Implementazione di un controllo dello stato della flotta Deadline Cloud su. GitHub Un controllo dello stato di salute può ridurre il rischio che una modifica accidentale della configurazione di reteAmazon Machine Image, del modello di lancio o della configurazione di rete venga eseguita inosservata.

Configura e utilizza le flotte gestite dai servizi Deadline Cloud

Una flotta gestita dai servizi (SMF) è una raccolta di lavoratori gestita da Deadline Cloud. Un SMF elimina la necessità di gestire la scalabilità della flotta per le richieste di elaborazione o di ridurre le dimensioni della flotta dopo il completamento delle attività.

Quando un SMF è associato a una coda utilizzando l'ambiente di coda conda predefinito, Deadline Cloud configura i lavoratori della flotta con il pacchetto software appropriato. Per le applicazioni dei partner supportate, consulta Default conda queue Environment nella Deadline Cloud User Guide.AWS

Nella maggior parte dei casi, non è necessario modificare un SMF per elaborare i carichi di lavoro. Tuttavia, alcune situazioni potrebbero richiedere di apportare modifiche alle flotte.

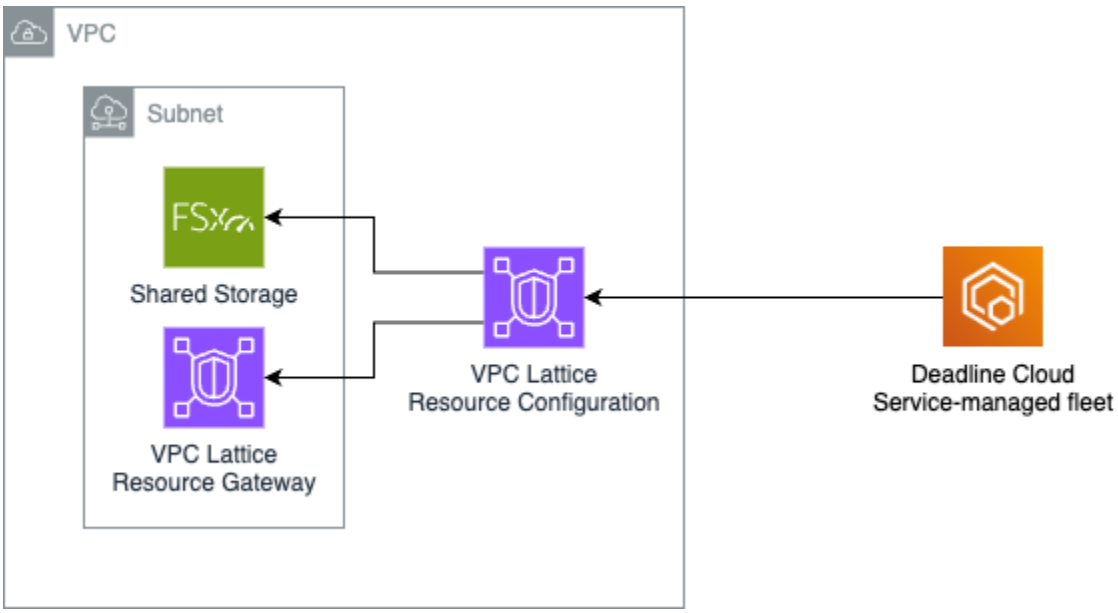
> **Note**
>
> Per installare software personalizzato sui lavoratori che utilizzano script di configurazione dell'host, vedere. Esegui script di configurazione host con privilegi di amministratore

Argomenti

- Connetti le risorse VPC alla tua SMF con gli endpoint di risorse VPC
- Usa gli allegati di lavoro con flotte gestite dai servizi

Connetti le risorse VPC alla tua SMF con gli endpoint di risorse VPC

Con gli endpoint di risorse Amazon VPC per le flotte gestite dai servizi Deadline Cloud (SMF), puoi connettere le tue risorse VPC come file system di rete (NFS), server di licenza e database con i tuoi lavoratori di Deadline Cloud. Questa funzionalità ti consente di sfruttare la piattaforma completamente gestita di Deadline Cloud integrandola con la tua infrastruttura esistente all'interno di un VPC.



> **Tip**
>
> Per un CloudFormation modello di riferimento che configura un FSx cluster Amazon e lo connette a una flotta gestita dai servizi, consulta smf_vpc_fsx nell'archivio degli esempi di Deadline Cloud su. GitHub

Come funzionano gli endpoint di risorse VPC

Gli endpoint di risorse VPC utilizzano VPC Lattice per creare una connessione sicura tra i dipendenti SMF di Deadline Cloud e le risorse nel tuo VPC. La connessione è unidirezionale, il che significa che i lavoratori possono stabilire una connessione alle risorse nel tuo VPC e trasferire dati avanti e indietro, ma le risorse nel tuo VPC non possono stabilire una connessione con un lavoratore.

Quando colleghi una risorsa VPC a una flotta gestita dal servizio Deadline Cloud, i tuoi dipendenti possono accedere alle risorse nel tuo VPC utilizzando un nome di dominio privato. Inoltre, il traffico fluisce dai lavoratori alle risorse VPC tramite VPC Lattice e le risorse nel VPC vedono il traffico proveniente dal gateway di risorse VPC Lattice.

Per saperne di più, consulta la guida per l'utente di VPC Lattice.

Prerequisiti

Prima di connettere le risorse VPC alla tua flotta gestita dai servizi Deadline Cloud, assicurati di disporre di quanto segue:

- Un AWS account con un VPC contenente le risorse che desideri connettere.
- Autorizzazioni IAM per creare e gestire risorse VPC Lattice.
- Una farm Deadline Cloud con almeno una flotta gestita dai servizi.
- Risorse VPC che desideri rendere accessibili (NFSFSx, server di licenza, ecc.).

Configura un endpoint di risorse VPC

Per configurare un endpoint di risorse VPC, devi creare risorse in VPC Lattice e AWS RAMquindi connettere tali risorse alla tua flotta in Deadline Cloud. Per configurare un endpoint di risorse VPC per SMF, completa i seguenti passaggi.

1. Per creare un gateway di risorse in VPC Lattice, consulta Creare un gateway di risorse nella guida utente di VPC Lattice.
2. Per creare una configurazione delle risorse in VPC Lattice, consulta Creare una configurazione delle risorse nella guida utente di VPC Lattice.
3. Per condividere la risorsa con la tua flotta di Deadline Cloud, crea una condivisione di risorse in. AWS RAM Per istruzioni, consulta Creazione di una condivisione di risorse.

   Durante la creazione di una condivisione di risorse, per Principal, seleziona Service Principal dal menu a discesa, quindi inserisci. **`fleets.deadline.amazonaws.com`**
4. Per connettere la configurazione delle risorse alla tua flotta di Deadline Cloud, completa i seguenti passaggi.

   a. Se non l'hai già fatto, apri la console Deadline Cloud.
   b. Nel riquadro di navigazione, scegli Fattorie, quindi seleziona la tua fattoria.
   c. Scegli la scheda Flotte, quindi seleziona la tua flotta.
   d. Scegliere la scheda Configurations (Configurazioni).
   e. In Endpoint di risorse VPC, scegli Modifica.
   f. Seleziona la configurazione delle risorse che hai creato, quindi scegli Salva modifiche.

Accesso alle risorse VPC

Dopo aver collegato la risorsa VPC alla flotta, i lavoratori possono accedervi utilizzando un nome di dominio privato nel seguente formato: `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Questo dominio è privato e accessibile solo dai lavoratori (non da Internet o dalla postazione di lavoro).

Per montare o configurare l'accesso alla risorsa VPC sui tuoi lavoratori, usa uno script di configurazione dell'host. Gli script di configurazione dell'host vengono eseguiti con privilegi di amministratore all'avvio dei worker e consentono di montare file system, configurare le impostazioni di rete o eseguire altre attività di configurazione.

Autenticazione e sicurezza

Per le risorse che richiedono l'autenticazione, archivia le credenziali in modo sicuro in AWS Secrets Manager, accedi ai segreti dagli script di configurazione dell'host o dagli script di lavoro e implementa le autorizzazioni del file system appropriate per controllare l'accesso. Considerate le implicazioni in termini di sicurezza quando condividete le risorse tra più flotte. Ad esempio, se due flotte sono collegate allo stesso storage condiviso, i job eseguiti su una flotta potrebbero essere in grado di accedere alle risorse create dall'altra flotta.

Considerazioni tecniche

Quando utilizzi gli endpoint di risorse VPC, considera quanto segue:

- Le connessioni possono essere avviate solo dai lavoratori alle risorse VPC, non dalle risorse VPC ai lavoratori.
- Una volta stabilita, una connessione persiste fino a quando non viene ripristinata, anche se la configurazione delle risorse viene disconnessa.
- La connessione VPC Lattice gestisce automaticamente le connessioni tra le zone di disponibilità senza costi aggiuntivi. Il gateway di risorse deve condividere una zona di disponibilità con la risorsa VPC, quindi consigliamo di configurare il gateway di risorse in modo che si estenda su tutte le zone di disponibilità.
- Il traffico che attraversa l'endpoint di risorse VPC utilizza Network Address Translation (NAT), che non è compatibile con tutti i casi d'uso. Ad esempio, Microsoft Active Directory (AD) non può connettersi tramite NAT.

Per ulteriori informazioni sulle quote VPC Lattice, consulta Quotas for VPC Lattice.

Risoluzione dei problemi

Se riscontri problemi con gli endpoint di risorse VPC, controlla quanto segue.

- Se ricevete un messaggio di errore come «mount.nfs: accesso negato dal server durante il montaggio», potrebbe essere necessario aggiornare la configurazione client del volume NFS.
- Verifica la configurazione delle risorse eseguendo test da un'istanza Amazon EC2 o AWS CloudShell nel tuo VPC.
- Metti alla prova la tua connessione Deadline Cloud con semplici lavori CLI. Per ulteriori informazioni, consulta gli esempi di Deadline Cloud su. GitHub
- Controlla le impostazioni del gruppo di sicurezza del Resource Gateway se riscontri errori di connessione.
- Abilita i log di accesso VPC per monitorare le connessioni.

Usa gli allegati di lavoro con flotte gestite dai servizi

Gli allegati Job trasferiscono file tra la tua workstation e i lavoratori di Deadline Cloud utilizzando Amazon Simple Storage Service (Amazon S3). Puoi utilizzare gli allegati dei lavori da soli o insieme allo storage condiviso per allegare dati ausiliari a lavori che non sono condivisi con altri lavori, come script di lavoro, file di configurazione o risorse di progetto archiviate localmente.

Per informazioni su come funzionano gli allegati di lavoro, consulta Job attachments nella Deadline Cloud User Guide. Per i dettagli sulla specificazione dei file di input e output nei job bundle, consulta. Usa gli allegati del lavoro per condividere file

Scegliete una modalità di filesystem

Quando invii un lavoro con allegati, puoi scegliere in che modo gli operatori caricano i file da Amazon S3 impostando la proprietà: `fileSystem`

- COPIED (impostazione predefinita): scarica tutti i file sul disco locale prima dell'inizio delle attività. Ideale quando ogni attività richiede la maggior parte dei file di input.
- VIRTUAL: monta un file system virtuale che scarica file su richiesta. Ideale quando le attività richiedono solo un sottoinsieme di file di input. Disponibile solo su Linux SMF worker.

> **Important**
>
> Il caching in modalità VIRTUALE può aumentare il consumo di memoria e non è ottimizzato per tutti i carichi di lavoro. Si consiglia di testare il carico di lavoro prima di eseguire i lavori di produzione.

Per informazioni dettagliate sulla configurazione della modalità filesystem, consulta Virtual file system nella Guida per l'utente di Deadline Cloud.

Ottimizza le prestazioni di trasferimento

La velocità di sincronizzazione dei file da Amazon S3 ai worker SMF dipende dalla configurazione del volume Amazon Elastic Block Store (Amazon EBS) del tuo parco macchine. Per impostazione predefinita, i lavoratori SMF utilizzano volumi Amazon EBS gp3 con impostazioni prestazionali di base. Per carichi di lavoro con file di input di grandi dimensioni o molti file di piccole dimensioni, puoi migliorare la velocità di trasferimento aumentando il throughput di Amazon EBS e le impostazioni IOPS. Puoi aggiornare queste impostazioni usando (). AWS Command Line Interface AWS CLI

Throughput (MiB/s)

La velocità con cui i dati possono essere letti o scritti sul volume. L'impostazione predefinita è 125 MiB/s, maximum is 1,000 MiB/s per i volumi gp3. Aumento per trasferimenti sequenziali di file di grandi dimensioni.

IOPS

Operazioni di input/output al secondo. L'impostazione predefinita è 3.000 IOPS, il massimo è 16.000 IOPS per i volumi gp3. Aumenta quando trasferisci molti file di piccole dimensioni.

> **Note**
>
> L'aumento del throughput e degli IOPS di Amazon EBS aumenta il costo della flotta. Per informazioni sui prezzi, consulta i prezzi di Deadline Cloud.

Per aggiornare le impostazioni di Amazon EBS su una flotta esistente utilizzando AWS CLI

- Esegui il comando seguente:

```
aws deadline update-fleet \
  --farm-id farm-0123456789abcdef0 \
  --fleet-id fleet-0123456789abcdef0 \
  --configuration '{
    "serviceManagedEc2": {
      "instanceCapabilities": {
        "vCpuCount": {"min": 4},
        "memoryMiB": {"min": 8192},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64",
        "rootEbsVolume": {
          "sizeGiB": 250,
          "iops": 6000,
          "throughputMiB": 500
        }
      },
      "instanceMarketOptions": {"type": "spot"}
    }
  }'
```

Scarica i risultati del lavoro

Una volta completato il lavoro, scarica i file di output utilizzando la CLI di Deadline Cloud o il monitor Deadline Cloud (AWS Deadline Cloud monitor):

```
deadline job download-output --job-id job-0123456789abcdef0
```

Per il download automatico degli output al termine dei lavori, consulta Download automatici nella Guida per l'utente di Deadline Cloud.

Implementa e configura software personalizzato sui lavoratori

AWS Deadline Cloud offre diversi metodi per distribuire e configurare software, plugin e strumenti personalizzati per i tuoi dipendenti. Il metodo scelto dipende dalle tue esigenze, ad esempio se hai bisogno dei privilegi di amministratore, con che frequenza il software cambia e se il software deve essere disponibile per tutti i lavori o solo per lavori specifici.

Scegli un metodo di distribuzione

Utilizza la tabella seguente per scegliere il metodo di distribuzione giusto per il tuo caso d'uso.

Criteri	Ambiente di coda	Script di configurazione dell'host	Pacchetto conda personalizzato
Sono richiesti i privilegi di amministratore	No	Sì	No
Quando viene eseguito	Inizio della sessione	Avvio del lavoratore	Inizio della sessione
Scope	Per coda o lavoro	Tutti i lavoratori della flotta	Per coda o lavoro
Può essere controllato inviando un lavoro	Sì	No	Sì
Complessità della configurazione	Bassa	Media	Elevata
Ideale per	Plugin semplici, script, variabili di ambiente	Driver di sistema, Docker, supporti di archiviazione	Applicazioni complesse con dipendenze

Guida rapida alla decisione:

- Hai bisogno dei privilegi di amministratore o root? Usa uno script di configurazione dell'host.

- Plugin o script semplice senza diritti di amministratore? Usa un ambiente di coda.
- Applicazione complessa con esigenze di controllo della versione? Crea un pacchetto conda personalizzato.

Configurazione dei lavori utilizzando ambienti di coda

AWS Deadline Cloud utilizza ambienti di coda per configurare il software sui dipendenti. Un ambiente consente di eseguire attività che richiedono molto tempo, come la configurazione e lo smontaggio, una volta per tutte le attività di una sessione. Definisce le azioni da eseguire su un lavoratore all'avvio o all'interruzione di una sessione. È possibile configurare un ambiente per una coda, i lavori che vengono eseguiti in coda e i singoli passaggi per un lavoro.

Gli ambienti vengono definiti come ambienti di coda o ambienti di lavoro. Crea ambienti di coda con la console Deadline Cloud o con Deadline: CreateQueueEnvironment gestisci e definisci gli ambienti di lavoro nei modelli di lavoro dei lavori che invii. Seguono la specifica Open Job Description (OpenJD) per gli ambienti. Per i dettagli, vedere le specifichehttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment <Environment>di OpenJD su. GitHub

Oltre a una `name` e`description`, ogni ambiente contiene due campi che definiscono l'ambiente sull'host. Questi sono:

- `script`— L'azione intrapresa quando questo ambiente viene eseguito su un lavoratore.
- `variables`— Un insieme di name/value coppie di variabili d'ambiente che vengono impostate quando si entra nell'ambiente.

È necessario impostare almeno una delle `script` o`variables`.

È possibile definire più di un ambiente nel modello di lavoro. Ogni ambiente viene applicato nell'ordine in cui è elencato nel modello. È possibile utilizzarlo per gestire la complessità degli ambienti.

L'ambiente di coda predefinito per Deadline Cloud utilizza il gestore di pacchetti conda per caricare il software nell'ambiente, ma puoi usare altri gestori di pacchetti. L'ambiente predefinito definisce due parametri per specificare il software da caricare. Queste variabili sono impostate dai mittenti forniti da Deadline Cloud, sebbene sia possibile impostarle nei propri script e applicazioni che utilizzano l'ambiente predefinito. Questi sono:

- `CondaPackages`— Un elenco separato da spazi di pacchetti conda che corrispondono alle specifiche da installare per il lavoro. Ad esempio, il mittente di Blender `blender=3.6` aggiungerebbe fotogrammi di rendering in Blender 3.6.
- `CondaChannels`— Un elenco separato da spazi di canali conda da cui installare i pacchetti. Per le flotte gestite dai servizi, i pacchetti vengono installati dal canale. `deadline-cloud` Puoi aggiungere altri canali.

Controlla l'ambiente di lavoro con gli ambienti di coda OpenJD

È possibile definire ambienti personalizzati per i lavori di rendering utilizzando ambienti di coda. Un ambiente di coda è un modello che controlla le variabili di ambiente, le mappature dei file e altre impostazioni per i lavori in esecuzione in una coda specifica. Consente di personalizzare l'ambiente di esecuzione per i lavori inviati a una coda in base ai requisiti dei carichi di lavoro. AWS Deadline Cloud offre tre livelli annidati in cui è possibile applicare ambienti Open Job Description (OpenJD): queue, job e step. Definendo gli ambienti di coda, è possibile garantire prestazioni coerenti e ottimizzate per diversi tipi di lavori, ottimizzare l'allocazione delle risorse e semplificare la gestione delle code.

L'ambiente di coda è un modello che si collega a una coda del AWS proprio account dalla console di gestione o utilizzando il AWS . AWS CLIÈ possibile creare un ambiente per una coda oppure creare più ambienti di coda applicati per creare l'ambiente di esecuzione. Questo approccio consente di creare e testare un ambiente in fasi per garantire che funzioni correttamente per i propri job.

Gli ambienti Job e Step sono definiti nel modello di job utilizzato per creare un job nella coda. La sintassi di OpenJD è la stessa in queste diverse forme di ambienti. In questa sezione li mostreremo all'interno dei modelli di lavoro.

Argomenti

- Imposta le variabili di ambiente in un ambiente di coda
- Imposta il percorso in un ambiente di coda
- Esegui un processo demone in background dall'ambiente di coda

Imposta le variabili di ambiente in un ambiente di coda

Molte applicazioni e framework utilizzano variabili di ambiente per controllare le impostazioni delle funzionalità, i livelli di registrazione e la configurazione dello schermo. È possibile utilizzare gli

ambienti Open Job Description (OpenJD) per impostare le variabili di ambiente che ogni comando di task all'interno del relativo ambito eredita.

Ambito della variabile di ambiente

AWS Deadline Cloud applica le variabili di ambiente dagli ambienti di coda collegati a una coda. All'interno di un modello di lavoro, puoi anche definire le variabili di ambiente a livello di processo e fase utilizzando gli ambienti OpenJD. Le variabili definite in un ambito più ristretto sostituiscono le variabili con lo stesso nome in un ambito più ampio.

- Ambiente di coda: un modello da allegare a una coda in Deadline Cloud. Le variabili si applicano a tutti i lavori inviati alla coda. È possibile impostare variabili con una `variables` mappa per valori fissi o utilizzare script per valori dinamici.
- Ambiente di lavoro: definito `jobEnvironments` in un modello di lavoro. Le variabili si applicano a tutte le fasi e le attività del lavoro. Una variabile a livello di processo sostituisce una variabile a livello di coda con lo stesso nome.
- Ambiente a fasi: definito in un modello di lavoro. `stepEnvironments` Le variabili si applicano solo alle attività di quella fase. Una variabile a livello di fase sostituisce una variabile a livello di processo o di coda con lo stesso nome.

Impostazione delle variabili in un ambiente di coda

È possibile impostare le variabili di ambiente in un ambiente di coda utilizzando una `variables` mappa per valori fissi o utilizzando un'`scriptonEnter`azione per valori dinamici.

Il seguente modello di ambiente di coda utilizza una `variables` mappa su cui impostare la `QT_QPA_PLATFORM` variabile`offscreen`, che consente alle applicazioni che utilizzano Qt Framework di funzionare su host di lavoro senza un display interattivo.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Per valori dinamici, come la modifica PATH o l'attivazione di ambienti virtuali, usa uno script che stampa le righe nel formato `openjd_env: VAR=value` su stdout. Il `openjd_env:` prefisso è obbligatorio. L'utilizzo di `echo``export`, o altri meccanismi di shell senza il prefisso non propaga le variabili ai job e alle attività.

Il seguente modello di ambiente di coda imposta la `QT_QPA_PLATFORM` variabile utilizzando uno script.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

Per collegare un ambiente di coda alla tua coda, usa la console Deadline Cloud o il. AWS CLI Per ulteriori informazioni, consulta Creare un ambiente di coda nella Guida per l'utente di AWS Deadline Cloud. Il AWS CLI comando seguente crea un ambiente di coda da un file modello.

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

Per esempi più complessi, come la creazione e l'attivazione di ambienti virtuali conda, consulta gli esempi di ambiente di coda di Deadline Cloud su. GitHub

Impostazione delle variabili in un modello di lavoro

In un modello di lavoro, aggiungi una `variables` mappa a una `stepEnvironments` voce `jobEnvironments` or. Ogni voce è una coppia chiave-valore in cui la chiave è il nome della variabile e il valore è il valore della variabile.

Il seguente modello di lavoro imposta la variabile di `QT_QPA_PLATFORM` ambiente su`offscreen`, che consente alle applicazioni che utilizzano Qt Framework di funzionare su host di lavoro senza un display interattivo.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

È possibile impostare più variabili in un'unica definizione di ambiente.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
    QT_QPA_PLATFORM: offscreen
```

È possibile fare riferimento ai parametri del lavoro in valori variabili utilizzando la `{{Param.ParameterName}}` sintassi.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Per sovrascrivere una variabile a livello di job per un passaggio specifico, definite una `stepEnvironments` voce con lo stesso nome di variabile. L'esempio seguente definisce `JOB_PROJECT_ID` a livello di processo il valore`project-12`, quindi sostituisce il valore a livello di fase con. `step-project-12` Le attività della fase utilizzano il valore a livello di fase.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    JOB_PROJECT_ID: project-12
```

```
steps:
- name: MyStep
  stepEnvironments:
  - name: StepEnv
    variables:
      JOB_PROJECT_ID: step-project-12
```

Prova: esegui l'esempio della variabile di ambiente

L'archivio di esempi di Deadline Cloud include un job bundle che dimostra l'impostazione e la visualizzazione delle variabili di ambiente. Il modello di lavoro di esempio definisce le variabili sia a livello di processo che a livello di fase, quindi esegue un'attività che stampa il risultato unito. Utilizzare la procedura seguente per eseguire il campione e controllare i risultati.

Prerequisiti

1. Se non disponi di una farm Deadline Cloud con una coda e una flotta Linux associata, segui l'esperienza di onboarding guidata nella console Deadline Cloud per crearne una con le impostazioni predefinite.
2. Se non disponi della CLI di Deadline Cloud AWS e del monitor Deadline Cloud sulla tua workstation, segui i passaggi in Configurare i mittenti di Deadline Cloud.
3. Utilizzalo per clonare l'archivio di esempi `git` di Deadline Cloud. GitHub

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

Esecuzione dell'esempio

1. Utilizza la CLI di Deadline Cloud per inviare `job_env_vars` il campione.

```
deadline bundle submit job_env_vars
```

2. Nel monitor Deadline Cloud, seleziona il nuovo lavoro per monitorarne l'avanzamento. Dopo che la Linux flotta associata alla coda ha un lavoratore disponibile, il lavoro viene completato in pochi secondi. Seleziona l'attività, quindi scegli Visualizza registri nel menu in alto a destra del pannello delle attività.

Confronto delle azioni della sessione con le relative definizioni

La visualizzazione del registro mostra tre azioni di sessione. Apri il file job_env_vars/template.yaml in un editor di testo per confrontare ogni azione con la sua definizione nel modello di lavoro.

1. JobEnvSeleziona l'azione Avvia sessione. L'output del registro mostra le variabili di ambiente a livello di job che vengono impostate.

```
Setting: JOB_VERBOSITY=MEDIUM
Setting: JOB_EXAMPLE_PARAM=An example parameter value
Setting: JOB_PROJECT_ID=project-12
Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/path
Setting: QT_QPA_PLATFORM=offscreen
```

   Le righe seguenti del modello di lavoro definiscono questo ambiente.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. Seleziona l'azione Avvia StepEnv sessione. L'output del registro mostra le variabili a livello di fase, incluse quelle sostituite. `JOB_PROJECT_ID`

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   Le righe seguenti del modello di lavoro definiscono questo ambiente.

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Seleziona l'azione Task run session. L'output del registro mostra le variabili di ambiente unite disponibili per l'attività. Si noti che `JOB_PROJECT_ID` utilizza il valore a livello di fase. `step-project-12`

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

Imposta il percorso in un ambiente di coda

Usa gli ambienti OpenJD per fornire nuovi comandi in un ambiente. Per prima cosa create una directory contenente i file di script, quindi aggiungete quella directory alle variabili di `PATH` ambiente in modo che gli eseguibili dello script possano eseguirli senza dover specificare ogni volta il percorso della directory. L'elenco delle variabili in una definizione di ambiente non fornisce un modo per modificare la variabile, quindi è possibile farlo eseguendo invece uno script. Dopo che lo script ha impostato le cose e modificato la`PATH`, esporta la variabile nel runtime di OpenJD con il comando. `echo "openjd_env: PATH=$PATH"`

Prerequisiti

Esegui i passaggi seguenti per eseguire il pacchetto di job di esempio con le variabili di ambiente dal repository github degli esempi di Deadline Cloud.

1. Se non disponi di una Deadline Cloud farm con una coda e una flotta Linux associata, segui l'esperienza di onboarding guidata nella console Deadline Cloud per crearne una con le impostazioni predefinite.
2. Se non disponi della CLI di Deadline Cloud e del monitor Deadline Cloud sulla tua workstation, segui i passaggi in Configurare i mittenti di Deadline Cloud dalla guida per l'utente.
3. Utilizzalo per clonare l'archivio di esempi `git` di Deadline Cloud. GitHub

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Esegui l'esempio di percorso

1. Utilizza la CLI di Deadline Cloud per inviare `job_env_with_new_command` il campione.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. Nel monitor Deadline Cloud, vedrai il nuovo lavoro e potrai monitorarne l'avanzamento. Una volta che la Linux flotta associata alla coda ha un lavoratore disponibile per eseguire l'attività del lavoro, il lavoro viene completato in pochi secondi. Seleziona l'attività, quindi scegli l'opzione Visualizza registri nel menu in alto a destra del pannello delle attività.

   Sulla destra ci sono due azioni di sessione, Launch RandomSleepCommand e Task run. Il visualizzatore di log al centro della finestra corrisponde all'azione della sessione selezionata sulla destra.

Confronta le azioni della sessione con le relative definizioni

In questa sezione si utilizza il monitor Deadline Cloud per confrontare le azioni della sessione con il punto in cui sono definite nel modello di lavoro. Continua dalla sezione precedente.

Apri il file job_env_with_new_command/template.yaml in un editor di testo. Confrontate le azioni della sessione con quelle definite nel modello di lavoro.

1. Seleziona l'azione Avvia RandomSleepCommand sessione nel monitor Deadline Cloud. Vedrai l'output del registro come segue.

```
 2024/07/16 17:25:32-07:00
 2024/07/16 17:25:32-07:00 ================================================
 2024/07/16 17:25:32-07:00 --------- Entering Environment: RandomSleepCommand
 2024/07/16 17:25:32-07:00 ================================================
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Setup
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
 2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
```

```
 2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Running action
 2024/07/16 17:25:32-07:00 ------------------------------------------------
 2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
 2024/07/16 17:25:32-07:00 Command started as pid: 2205
 2024/07/16 17:25:32-07:00 Output:
 2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
 No newer logs at this moment.
```

Le seguenti righe del modello di lavoro hanno specificato questa azione.

```
 jobEnvironments:
 - name: RandomSleepCommand
   description: Adds a command 'random-sleep' to the environment.
   script:
     actions:
       onEnter:
         command: bash
         args:
         - "{{Env.File.Enter}}"
     embeddedFiles:
     - name: Enter
       type: TEXT
       data: |
         #!/bin/env bash
         set -euo pipefail

         # Make a bin directory inside the session's working directory for providing
 new commands
         mkdir -p '{{Session.WorkingDirectory}}/bin'

         # If this bin directory is not already in the PATH, then add it
         if ! [[ ":$PATH:" == *':{{Session.WorkingDirectory}}/bin:'* ]]; then
           export "PATH={{Session.WorkingDirectory}}/bin:$PATH"
```

```
          # This message to Open Job Description exports the new PATH value to the
 environment
          echo "openjd_env: PATH=$PATH"
        fi

        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...
```

2. Seleziona l'azione Avvia StepEnv sessione nel monitor Deadline Cloud. L'output del log viene visualizzato come segue.

```
 2024/07/16 17:25:33-07:00
 2024/07/16 17:25:33-07:00 ================================================
 2024/07/16 17:25:33-07:00 --------- Running Task
 2024/07/16 17:25:33-07:00 ================================================
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Setup
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
 2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Running action
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
 2024/07/16 17:25:33-07:00 Command started as pid: 2256
 2024/07/16 17:25:33-07:00 Output:
 2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
 2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
 2024/07/16 17:26:00-07:00 ------------------------------------------------
 2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
 2024/07/16 17:26:00-07:00 ------------------------------------------------
```

3. Questa azione è stata specificata nelle righe seguenti del modello di lavoro.

```
steps:
- name: EnvWithCommand
  script:
    actions:
      onRun:
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      data: |
        set -xeuo pipefail

        # Run the script installed into PATH by the job environment
        random-sleep 12.5 27.5
  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Esegui un processo demone in background dall'ambiente di coda

In molti casi d'uso del rendering, il caricamento dei dati dell'applicazione e della scena può richiedere molto tempo. Se un job li ricarica per ogni frame, impiegherà la maggior parte del tempo a sovraccaricare. Spesso è possibile caricare l'applicazione una sola volta come processo demone in background, fare in modo che carichi i dati della scena e quindi inviarle i comandi tramite la comunicazione tra processi (IPC) per eseguire i rendering.

Molte delle integrazioni open source di Deadline Cloud utilizzano questo modello. Il progetto Open Job Description fornisce una libreria di runtime di adattatori con modelli IPC robusti su tutti i sistemi operativi supportati.

Per dimostrare questo modello, esiste un pacchetto di job di esempio autonomo che utilizza Python e codice bash per implementare un demone in background e l'IPC per le attività per comunicare con esso. Il demone è implementato in Python e ascolta un SIGUSR1 segnale POSIX per sapere quando

elaborare un'attività. I dettagli dell'attività vengono passati al demone in un file JSON specifico e i risultati dell'esecuzione dell'operazione vengono restituiti come un altro file JSON.

Prerequisiti

Esegui i passaggi seguenti per eseguire il pacchetto di job di esempio con un processo daemon dal repository github degli esempi di Deadline Cloud.

1. Se non disponi di una farm Deadline Cloud con una coda e una flotta Linux associata, segui l'esperienza di onboarding guidata nella console Deadline Cloud per crearne una con le impostazioni predefinite.
2. Se non disponi della CLI di Deadline Cloud e del monitor Deadline Cloud sulla tua workstation, segui i passaggi in Configurare i mittenti di Deadline Cloud dalla guida per l'utente.
3. Utilizzalo per clonare l'archivio di esempi `git` di Deadline Cloud. GitHub

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Esegui l'esempio del demone

1. Utilizza la CLI di Deadline Cloud per inviare `job_env_daemon_process` il campione.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. Nell'applicazione di monitoraggio Deadline Cloud, vedrai il nuovo lavoro e potrai monitorarne l'avanzamento. Una volta che il Linux parco macchine associato alla coda ha un lavoratore disponibile per eseguire l'operazione, questa viene completata in circa un minuto. Con una delle attività selezionate, scegli l'opzione Visualizza registri nel menu in alto a destra del pannello delle attività.

   Sulla destra ci sono due azioni di sessione, Launch DaemonProcess e Task run. Il visualizzatore di log al centro della finestra corrisponde all'azione della sessione selezionata sulla destra.

Seleziona l'opzione Visualizza i registri per tutte le attività. La sequenza temporale mostra il resto delle attività eseguite come parte della sessione e l'`Shut down DaemonProcess`azione uscita dall'ambiente.

Visualizza i registri dei demoni

1. In questa sezione si utilizza il monitor Deadline Cloud per confrontare le azioni della sessione con il punto in cui sono definite nel modello di lavoro. Continua dalla sezione precedente.

   Apri il file job_env_daemon_process/template.yaml in un editor di testo. Confrontate le azioni della sessione con quelle definite nel modello di lavoro.

2. Seleziona l'azione della `Launch DaemonProcess` sessione nel monitor di Deadline Cloud. Vedrai l'output del registro come segue.

```
 2024/07/17 16:27:20-07:00
 2024/07/17 16:27:20-07:00 ================================================
 2024/07/17 16:27:20-07:00 --------- Entering Environment: DaemonProcess
 2024/07/17 16:27:20-07:00 ================================================
 2024/07/17 16:27:20-07:00 ------------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Setup
 2024/07/17 16:27:20-07:00 ------------------------------------------------
 2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
```

```
 2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 ------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Running action
 2024/07/17 16:27:20-07:00 ------------------------------------------
 2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
 2024/07/17 16:27:20-07:00 Command started as pid: 2187
 2024/07/17 16:27:20-07:00 Output:
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

Le seguenti righe del modello di lavoro hanno specificato questa azione.

```
  stepEnvironments:
  - name: DaemonProcess
    description: Runs a daemon process for the step's tasks to share.
    script:
      actions:
        onEnter:
          command: bash
          args:
          - "{{Env.File.Enter}}"
        onExit:
          command: bash
          args:
          - "{{Env.File.Exit}}"
      embeddedFiles:
      - name: Enter
        filename: enter-daemon-process-env.sh
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail
```

```
          DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
          echo "openjd_env: DAEMON_LOG=$DAEMON_LOG"
          nohup python {{Env.File.DaemonScript}} > $DAEMON_LOG 2>&1 &
          echo "openjd_env: DAEMON_PID=$!"
          echo "openjd_env:
 DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

          echo 0 > 'daemon_log_cursor.txt'
    ...
```

3. Seleziona una delle azioni Task run: N session nel monitor Deadline Cloud. Vedrai l'output del registro come segue.

```
2024/07/17 16:27:22-07:00
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 --------- Running Task
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 Parameter values:
 2024/07/17 16:27:22-07:00 Frame(INT) = 2
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Setup
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
 2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Running action
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
 2024/07/17 16:27:22-07:00 Command started as pid: 2301
 2024/07/17 16:27:22-07:00 Output:
 2024/07/17 16:27:23-07:00 Daemon PID is 2223
 2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Previous output from daemon
 2024/07/17 16:27:23-07:00 ===
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 Sending command to daemon
 2024/07/17 16:27:23-07:00 Received task result:
```

```
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00   "result": "SUCCESS",
2024/07/17 16:27:23-07:00   "processedTaskCount": 1,
2024/07/17 16:27:23-07:00   "randomValue": 0.2578537967668988,
2024/07/17 16:27:23-07:00   "failureRate": 0.1
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 === Daemon log from running the task
2024/07/17 16:27:23-07:00 Loading the task details file
2024/07/17 16:27:23-07:00 Received task details:
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00  "pid": 2329,
2024/07/17 16:27:23-07:00  "frame": 2
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00 Processing frame number 2
2024/07/17 16:27:23-07:00 Writing result
2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 ----------------------------------------------
2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
2024/07/17 16:27:23-07:00 ----------------------------------------------
```

Le seguenti righe del modello di lavoro sono ciò che specifica questa azione. ```passi:

```
steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
        - '{{Task.File.Run}}'
```

```
    embeddedFiles:
    - name: Run
      filename: run-task.sh
      type: TEXT
      data: |
        # This bash script sends a task to the background daemon process,
        # then waits for it to respond with the output result.

        set -euo pipefail

        source "$DAEMON_BASH_HELPER_SCRIPT"

        echo "Daemon PID is $DAEMON_PID"
        echo "Daemon log file is $DAEMON_LOG"

        print_daemon_log "Previous output from daemon"

        send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
        wait_for_daemon_task_result

        echo Received task result:
        echo "$TASK_RESULT" | jq .

        print_daemon_log "Daemon log from running the task"

  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Fornisci candidature per i tuoi lavori

È possibile utilizzare un ambiente di coda per caricare le applicazioni per elaborare i lavori. Quando crei una flotta gestita dai servizi utilizzando la console Deadline Cloud, hai la possibilità di creare un ambiente di coda che utilizza il gestore di pacchetti conda per caricare le applicazioni.

Se desideri utilizzare un gestore di pacchetti diverso, puoi creare un ambiente di coda per quel gestore. Per un esempio di utilizzo di Rez, vediUsa un gestore di pacchetti diverso.

Deadline Cloud fornisce un canale conda per caricare una selezione di applicazioni di rendering nel tuo ambiente. Supportano i mittenti che Deadline Cloud fornisce per le applicazioni di creazione di contenuti digitali.

Puoi anche caricare software per conda-forge da utilizzare nei tuoi lavori. Gli esempi seguenti mostrano modelli di lavoro che utilizzano l'ambiente di coda fornito da Deadline Cloud per caricare le applicazioni prima di eseguire il lavoro.

Argomenti

- Ottenere un'applicazione da un canale conda
- Usa un gestore di pacchetti diverso

Ottenere un'applicazione da un canale conda

Puoi creare un ambiente di coda personalizzato per i tuoi lavoratori di Deadline Cloud che installino il software che preferisci. Questo esempio di ambiente di coda ha lo stesso comportamento dell'ambiente utilizzato dalla console per le flotte gestite dai servizi. Esegue direttamente conda per creare l'ambiente.

L'ambiente crea un nuovo ambiente virtuale conda per ogni sessione di Deadline Cloud eseguita su un lavoratore, quindi elimina l'ambiente al termine.

Conda memorizza nella cache i pacchetti scaricati in modo che non debbano essere scaricati nuovamente, ma ogni sessione deve collegare tutti i pacchetti all'ambiente.

L'ambiente definisce tre script che vengono eseguiti quando Deadline Cloud avvia una sessione su un lavoratore. Il primo script viene eseguito quando viene chiamata l'`onEnter`azione. Chiama gli altri due per impostare le variabili di ambiente. Al termine dell'esecuzione dello script, l'ambiente conda è disponibile con tutte le variabili di ambiente specificate impostate.

Per la versione più recente dell'esempio, vedete conda_queue_env_console_equivalent.yaml nel repository on. deadline-cloud-samples GitHub

Se desideri utilizzare un'applicazione che non è disponibile nel canale conda, puoi creare un canale conda in Amazon S3 e quindi creare i tuoi pacchetti per quell'applicazione. Per ulteriori informazioni, consulta Crea un canale conda usando S3.

Ottieni librerie open source da conda-forge

Questa sezione descrive come utilizzare le librerie open source del canale. `conda-forge` L'esempio seguente è un modello di lavoro che utilizza il pacchetto `polars` Python.

Il job imposta i `CondaChannels` parametri `CondaPackages` and definiti nell'ambiente di coda che indicano a Deadline Cloud dove trovare il pacchetto.

La sezione del modello di lavoro che imposta i parametri è:

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
 to handle this.
  type: STRING
  default: polars
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
 Environment to handle this.
  type: STRING
  default: conda-forge
```

Per la versione più recente del modello di lavoro di esempio completo, vedete stage_1_self_contained_template/template.yaml. Per la versione più recente dell'ambiente di coda che carica i pacchetti conda, vedete conda_queue_env_console_equivalent.yaml nel repository on. deadline-cloud-samples GitHub

Accedi Blender dal canale deadline-cloud

L'esempio seguente mostra un modello di lavoro che Blender proviene dal canale conda. `deadline-cloud` Questo canale supporta i mittenti forniti da Deadline Cloud per il software di creazione di contenuti digitali, sebbene sia possibile utilizzare lo stesso canale per caricare software per uso personale.

Per un elenco del software fornito dal `deadline-cloud` canale, consulta Ambiente di coda predefinito nella Guida per l'utente di AWS Deadline Cloud.

Questo lavoro imposta il `CondaPackages` parametro definito nell'ambiente di coda per indicare a Deadline Cloud di Blender caricarsi nell'ambiente.

La sezione del modello di lavoro che imposta il parametro è:

```
- name: CondaPackages
```

```
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
channel.
```

Per la versione più recente del modello di lavoro di esempio completo, vedi blender_render/template.yaml. Per la versione più recente dell'ambiente di coda che carica i pacchetti conda, vedi conda_queue_env_console_equivalent.yaml nel repository on. deadline-cloud-samplesGitHub

Usa un gestore di pacchetti diverso

Il gestore di pacchetti predefinito per Deadline Cloud è conda. Se è necessario utilizzare un gestore di pacchetti diverso, ad esempioRez, è possibile creare un ambiente di coda personalizzato che contenga script che utilizzano invece il gestore di pacchetti.

Questo esempio di ambiente di coda fornisce lo stesso comportamento dell'ambiente utilizzato dalla console per le flotte gestite dai servizi. Sostituisce il gestore di pacchetti conda con. Rez

L'ambiente definisce tre script che vengono eseguiti quando Deadline Cloud avvia una sessione su un lavoratore. Il primo script viene eseguito quando viene chiamata l'`onEnter`azione. Chiama gli altri due per impostare le variabili di ambiente. Al termine dell'esecuzione dello script, l'Rezambiente è disponibile con tutte le variabili di ambiente specificate impostate.

L'esempio presuppone che si disponga di una flotta gestita dal cliente che utilizza un file system condiviso per i pacchetti Rez.

Per la versione più recente dell'esempio, vedi rez_queue_env.yaml nel repository su. deadline-cloud-samplesGitHub

Crea un canale conda usando S3

Se i tuoi lavori devono eseguire applicazioni non disponibili sui `conda-forge`canali `deadline-cloud`or, puoi ospitare un canale conda personalizzato per fornire i tuoi pacchetti. Quando crei una coda nella console AWS Deadline Cloud (Deadline Cloud), la console aggiunge un ambiente di coda conda per impostazione predefinita. Per rendere i pacchetti disponibili per i lavori, aggiungi il canale personalizzato all'ambiente di coda.

Un canale conda è un contenuto statico ospitato che puoi ospitare in vari modi, ad esempio su un file system o in un bucket Amazon Simple Storage Service (Amazon S3). Se la tua farm Deadline Cloud utilizza un file system condiviso per le risorse, puoi utilizzare qualsiasi percorso su di esso come nome del canale. Puoi ospitare il canale in un bucket Amazon S3 per un accesso più ampio utilizzando le autorizzazioni AWS Identity and Access Management (IAM).

Puoi creare e testare pacchetti localmente, quindi pubblicarli su un canale. La creazione di pacchetti localmente è un modo semplice per iniziare a iterare sulle ricette di compilazione dei pacchetti senza configurare l'infrastruttura. Puoi anche utilizzare una coda di creazione di pacchetti Deadline Cloud per creare pacchetti e pubblicarli su un canale. Una coda per la creazione di pacchetti semplifica la manutenzione dei pacchetti per più sistemi operativi e configurazioni di acceleratori. Puoi aggiornare le versioni e inviare set completi di build di pacchetti da qualsiasi luogo.

Puoi configurare i canali per il tuo studio e la tua Deadline Cloud farm in diversi modi. Puoi avere un canale Amazon S3 e configurare tutte le workstation e gli host della farm per utilizzarlo. Puoi anche avere più di un canale e configurare il mirroring con AWS DataSync (). DataSync Ad esempio, la coda di creazione dei pacchetti Deadline Cloud può essere pubblicata su un canale Amazon S3 di cui viene eseguito il mirroring in locale per workstation e host di farm locali.

Argomenti

- Compila e testa pacchetti localmente
- Pubblica pacchetti su un canale conda Amazon S3
- Configura le autorizzazioni per la coda di produzione per pacchetti conda personalizzati
- Aggiungi un canale conda a un ambiente di coda
- Crea un pacchetto conda per un'applicazione o un plug-in
- Crea una ricetta di costruzione di conda per Blender
- Crea una ricetta di costruzione di conda per Autodesk Maya
- Crea una ricetta di costruzione conda per l'adattatore Maya
- Crea una ricetta di compilazione conda per il plugin Autodesk Maya to Arnold (MtoA)
- Automatizza la creazione di pacchetti con Deadline Cloud

Compila e testa pacchetti localmente

Prima di pubblicare pacchetti su Amazon S3 o configurare CI/CD l'automazione nella tua farm Deadline Cloud, puoi creare e testare pacchetti conda sulla tua workstation utilizzando un canale di

file system locale. Questo approccio consente di iterare rapidamente a livello locale sulle ricette e verificare i pacchetti.

Il `rattler-build publish` comando crea una ricetta, copia il pacchetto risultante su un canale e indicizza il canale in un unico passaggio. Quando si sceglie come destinazione una directory del filesystem locale, `rattler-build` crea e inizializza automaticamente il canale se la directory non esiste.

Le seguenti istruzioni utilizzano la ricetta di esempio Blender 4.5 contenuta nell'archivio degli esempi di Deadline Cloud. GitHub Puoi sostituire una ricetta diversa dall'archivio degli esempi o utilizzare la tua ricetta.

Prerequisiti

Prima di iniziare, installa i seguenti strumenti sulla tua workstation:

- pixi — Un gestore di pacchetti utilizzato per installare `rattler-build` e testare i pacchetti. Installa pixi da pixi.sh.
- rattler-build — Lo strumento per la creazione di pacchetti utilizzato dalle ricette conda di Deadline Cloud. Dopo aver installato pixi, esegui il seguente comando per l'installazione. `rattler-build`

```
pixi global install rattler-build
```

- git — Necessario per clonare il repository degli esempi. OnWindows, git for fornisce Windows anche una `bash` shell, richiesta da alcune ricette di Windows esempio.

Creazione e pubblicazione di un pacchetto su un canale locale

In questa procedura, cloni il repository di esempi di Deadline Cloud e lo usi `rattler-build publish` per creare e pubblicare il pacchetto su un canale di filesystem locale.

> **Note**
>
> Le applicazioni di grandi dimensioni possono richiedere decine di GB di spazio libero su disco per l'archivio di origine, i file estratti e l'output della build. Assicurati di utilizzare un disco con spazio disponibile sufficiente per l'output di compilazione del pacchetto.

Per creare e pubblicare un pacchetto su un canale locale

1. Clona il repository di esempi di Deadline Cloud.

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

2. Passare alla directory `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Esegui il comando seguente per creare la ricetta Blender 4.5 e pubblicare il pacchetto in una directory di canale locale.

   Su Linux emacOS, esegui il seguente comando.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

   Su Windows (cmd), esegui il comando seguente.

```
rattler-build publish blender-4.5/recipe/recipe.yaml ^
    --to file://%USERPROFILE%/my-conda-channel ^
    --build-number=+1
```

   Il `rattler-build publish` comando esegue le seguenti azioni:

   - Crea il pacchetto a partire dalla ricetta.
   - Crea la directory dei canali se la directory non esiste.
   - Copia il file del pacchetto sul canale.
   - Indicizza il canale in modo che i gestori di pacchetti possano trovare il pacchetto.

   Se la ricetta del pacchetto dipende dai pacchetti di un canale particolare, come conda-forge, aggiungila al comando. `-c conda-forge`

Informazioni sui numeri di build

L'`--build-number=+1`opzione seleziona automaticamente il numero di build successivo in base a ciò che già esiste nel canale di destinazione. La migliore pratica è quella di non sovrascrivere mai un pacchetto in un canale. Crea sempre con un nuovo numero di build se altrimenti il pacchetto avrebbe lo stesso nome di file. Using `--build-number=+1` consente di raggiungere questo obiettivo quando si passa a un canale di produzione o a un canale di staging che rispecchia la produzione.
Se desideri controllare direttamente il numero di build, puoi impostarlo con un valore specifico come. `--build-number=7` Se ometti l'opzione, `rattler-build` utilizza il numero di build definito nel `recipe.yaml` file.

Per ulteriori informazioni in meritо`rattler-build publish`, consulta la documentazione di rattler-build publish.

Build di debug

Se una compilazione fallisce, `rattler-build` conserva la directory di compilazione in modo da poter indagare. Esegui il comando seguente per aprire una shell interattiva nell'ambiente di compilazione con tutte le variabili di ambiente impostate com'erano durante la compilazione.

```
rattler-build debug shell
```

Dalla shell di debug, puoi modificare i file, eseguire singoli comandi di compilazione e aggiungere dipendenze per isolare il problema. Per ulteriori informazioni, consulta Debugging build nella documentazione di rattler-build.

Test del pacchetto

Dopo aver creato e pubblicato il pacchetto, create un progetto pixi temporaneo. Utilizzate il progetto per installare il pacchetto dal canale locale e verificare che funzioni correttamente.

Per testare il pacchetto

1. Crea una directory di test temporanea e inizializza un progetto pixi con il canale locale.

   Su Linux emacOS, esegui i seguenti comandi.

   ```
   mkdir package-test-env
   ```

```
cd package-test-env
pixi init --channel file://$HOME/my-conda-channel
```

Su Windows (cmd), esegui i seguenti comandi.

```
mkdir package-test-env
cd package-test-env
pixi init --channel file://%USERPROFILE%/my-conda-channel
```

2. Aggiungi il pacchetto al progetto.

```
pixi add blender=4.5
```

3. Verifica che il pacchetto funzioni correttamente.

```
pixi run blender --version
```

Il `pixi run`comando attiva l'ambiente conda per la directory del progetto ed esegue il comando specificato al suo interno. L'ambiente persiste nella directory del progetto, quindi è possibile utilizzare lo stesso `pixi run` comando da altri terminali.

Quando sei soddisfatto del pacchetto, puoi pubblicarlo su un canale conda di Amazon S3 in modo che gli operatori di Deadline Cloud possano installare il pacchetto. Vedi Pubblicare pacchetti su un canale conda S3.

Rimozione dei pacchetti dal canale

Evitate di rimuovere i pacchetti dai canali utilizzati per la produzione, perché i lockfile fanno riferimento a pacchetti specifici tramite hash. La rimozione di un pacchetto impedisce di ricreare ambienti da quei file di blocco. Per i canali di sviluppo e test, è possibile rimuovere un pacchetto specifico eliminando il `.conda` file dalla directory del canale e quindi reindicizzando il canale. Innanzitutto, installa. `rattler-index`

```
pixi global install rattler-index
```

Quindi elimina il file del pacchetto e reindicizza il canale.

Su Linux emacOS, esegui i seguenti comandi.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

Su Windows (cmd), esegui i seguenti comandi.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

I file del pacchetto vengono archiviati in sottodirectory specifiche della piattaforma come, o. `linux-64` `win-64` `osx-arm64` Elenca il contenuto di queste sottodirectory per trovare il nome esatto del pacchetto che desideri rimuovere.

Pulizia

Dopo il test, puoi rimuovere il progetto di test e il canale locale.

Per ripulire le risorse di test

1. Rimuovi la directory del progetto di test.

   Su Linux emacOS, esegui il seguente comando.

   ```
   rm -rf package-test-env
   ```

   Su Windows (cmd), esegui il comando seguente.

   ```
   rmdir /s /q package-test-env
   ```

2. Rimuovi la directory del canale conda locale.

   Su Linux emacOS, esegui il seguente comando.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Su Windows (cmd), esegui il comando seguente.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Facoltativo) Rimuove la directory `rattler-build` di output che contiene il file del pacchetto creato.

Su Linux emacOS, esegui il comando seguente.

```
rm -rf deadline-cloud-samples/conda_recipes/output
```

Su Windows (cmd), esegui il comando seguente.

```
rmdir /s /q deadline-cloud-samples\conda_recipes\output
```

Pubblica pacchetti su un canale conda Amazon S3

Puoi pubblicare pacchetti conda su un bucket Amazon Simple Storage Service (Amazon S3) in modo che gli operatori di AWS Deadline Cloud (Deadline Cloud) possano installarli per eseguire lavori. Il `rattler-build publish` comando funziona con Amazon S3 allo stesso modo di un canale di filesystem locale. Il comando può creare una ricetta e pubblicare il risultato oppure pubblicare un file di pacchetto già creato. In entrambi i casi, il comando carica il pacchetto nel bucket e indicizza il canale in un unico passaggio.

Il `rattler-build publish` comando si autentica AWS utilizzando la catena di credenziali standard, quindi utilizza la configurazione come qualsiasi altro strumento. AWS AWS Per ulteriori informazioni sulla configurazione delle credenziali, consultate Configurazione e impostazioni dei file di credenziali nella Guida per l'utente ().AWS Command Line Interface AWS CLI

Prerequisiti

Prima di pubblicare pacchetti su Amazon S3, completa i seguenti prerequisiti:

- pixi e rattler-build: installa pixi da pixi.sh, quindi installa. `rattler-build`

  ```
  pixi global install rattler-build
  ```

- git — Necessario per clonare il repository degli esempi. OnWindows, git for fornisce Windows anche una `bash` shell, richiesta da alcune ricette di Windows esempio.
- Bucket Amazon S3: un bucket Amazon S3 da utilizzare come canale conda. Puoi utilizzare il bucket Job Attachments della tua Deadline Cloud farm o creare un bucket separato.
- AWS credenziali: configura le credenziali sulla tua workstation utilizzando il comando o il comando. `aws configure` `aws login` Per ulteriori informazioni, consulta Configurazione della AWS CLI nella Guida per l'utente di AWS Command Line Interface .

- Autorizzazioni IAM — (Facoltativo) Per ridurre l'ambito delle autorizzazioni a disposizione delle tue credenziali, puoi utilizzare una policy AWS Identity and Access Management (IAM) che concede solo le seguenti autorizzazioni sul bucket Amazon S3 e sul prefisso di canale che utilizzi (ad esempio,): `/Conda/*`
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:DeleteObject`
  - `s3:ListBucket`
  - `s3:GetBucketLocation`

Pubblicazione di un pacchetto su un canale Amazon S3

Usalo `rattler-build publish` con un `s3://` target per pubblicare un pacchetto sul tuo canale Amazon S3 conda. Se il canale non esiste nel bucket, `rattler-build` inizializza automaticamente il canale. Prima di iniziare, assicuratevi di aver completato i prerequisiti.

L'esempio seguente pubblica la ricetta di esempio Blender 4.5 dal repository di esempi di Deadline Cloud su. GitHub È possibile sostituire una ricetta diversa dall'archivio degli esempi o utilizzare una ricetta personalizzata.

> **Note**
>
> Le applicazioni di grandi dimensioni possono richiedere decine di GB di spazio libero su disco per l'archivio di origine, i file estratti e l'output di compilazione. Assicurati di utilizzare un disco con spazio disponibile sufficiente per l'output di compilazione del pacchetto.

Per pubblicare un pacchetto su un canale Amazon S3

1. Clona l'archivio di esempi di Deadline Cloud.

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

2. Passare alla directory `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Eseguire il seguente comando seguente. Sostituiscilo *amzn-s3-demo-bucket* con il nome del tuo bucket.

```
rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
Conda/Default --build-number=+1
```

   Il `/Conda/Default` prefisso organizza il canale all'interno del bucket. È possibile utilizzare un prefisso diverso, ma il prefisso deve essere coerente in tutti i comandi e le configurazioni di coda che fanno riferimento al canale.

> **Informazioni sui numeri di build**
>
> L'`--build-number=+1`opzione seleziona automaticamente il numero di build successivo in base a ciò che già esiste nel canale di destinazione. La migliore pratica è quella di non sovrascrivere mai un pacchetto in un canale. Crea sempre con un nuovo numero di build se altrimenti il pacchetto avrebbe lo stesso nome di file. Using `--build-number=+1` consente di raggiungere questo obiettivo quando si passa a un canale di produzione o a un canale di staging che rispecchia la produzione.
> Se desideri controllare direttamente il numero di build, puoi impostarlo con un valore specifico come. `--build-number=7` Se ometti l'opzione, `rattler-build` utilizza il numero di build definito nel `recipe.yaml` file.

Se la ricetta del pacchetto dipende dai pacchetti di un canale particolare, come conda-forge, `-c conda-forge` aggiungila al comando.

Puoi anche pubblicare un file di pacchetto che hai già creato, ad esempio un `.conda` file da una build locale. *amzn-s3-demo-bucket*Sostituiscilo con il nome del tuo bucket.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Inizializzazione o reindicizzazione di un canale

Quando si utilizza `rattler-build publish` per pubblicare un pacchetto, il comando inizializza automaticamente il canale se il canale non esiste già. Nella maggior parte dei casi, non è necessario inizializzare o reindicizzare il canale manualmente.

Potrebbe essere necessario inizializzare o reindicizzare manualmente un canale nelle seguenti situazioni:

- Desideri creare un canale vuoto prima di pubblicare qualsiasi pacchetto, ad esempio per verificare che l'ambiente di coda di Deadline Cloud possa connettersi al canale.
- Hai caricato o eliminato `.conda` file direttamente con gli strumenti di Amazon S3 anziché utilizzarli `rattler-build publish` e l'indice dei canali non è aggiornato.

Inizializzazione di un canale vuoto

Per inizializzare un canale vuoto, create un `repodata.json` file e caricatelo `noarch` nella sottodirectory del prefisso del canale. Sostituiscilo *`amzn-s3-demo-bucket`* con il nome del tuo bucket.

```
echo '{"info":{"subdir":"noarch"},"packages":{},"packages.conda":{},"removed":
[],"repodata_version":1}' > empty_channel_repodata.json
aws s3api put-object --body empty_channel_repodata.json --key Conda/Default/noarch/
repodata.json --bucket amzn-s3-demo-bucket
```

Il `/Conda/Default` prefisso deve corrispondere al prefisso del canale utilizzato dall'ambiente di coda. Dopo aver inizializzato il canale, è possibile pubblicare pacchetti sul canale utilizzando. `rattler-build publish`

Reindicizzazione di un canale

Se l'indice del canale non è aggiornato, utilizzalo `rattler-index` per ricostruire l'indice dai file del pacchetto nel canale. Innanzitutto, install`rattler-index`.

```
pixi global install rattler-index
```

Quindi reindicizza il canale. *`amzn-s3-demo-bucket`*Sostituiscilo con il nome del tuo bucket.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

Test del pacchetto

Dopo aver pubblicato il pacchetto, create un progetto pixi temporaneo per verificare che il pacchetto funzioni correttamente. Il progetto installa il pacchetto dal canale Amazon S3.

Per testare il pacchetto

1. Crea una directory di test temporanea e inizializza un progetto pixi con il canale Amazon S3. Sostituiscilo *amzn-s3-demo-bucket* con il nome del tuo bucket.

```
mkdir package-test-env
cd package-test-env
pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
```

2. Aggiungi il pacchetto al progetto.

```
pixi add blender=4.5
```

3. Verifica che il pacchetto funzioni correttamente.

```
pixi run blender --version
```

   Il `pixi run`comando attiva l'ambiente conda per la directory del progetto ed esegue il comando specificato al suo interno. L'ambiente persiste nella directory del progetto, quindi è possibile utilizzare lo stesso `pixi run` comando da altri terminali.

Rimozione dei pacchetti dal canale

Evitate di rimuovere i pacchetti dai canali utilizzati per la produzione, perché i lockfile fanno riferimento a pacchetti specifici tramite hash. La rimozione di un pacchetto impedisce di ricreare ambienti da quei file di blocco. Per i canali di sviluppo e test, è possibile rimuovere un pacchetto specifico eliminando il `.conda` file dal bucket e quindi reindicizzando il canale.

Eliminate il file del pacchetto e quindi reindicizzate il canale. *amzn-s3-demo-bucket*Sostituiscilo con il nome del tuo bucket.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

Dopo aver eliminato il file, reindicizza il canale per aggiornare i metadati del canale. Per istruzioni, consultate Reindicizzazione di un canale.

I file del pacchetto vengono archiviati in sottodirectory specifiche della piattaforma come, o. `linux-64` `win-64` `osx-arm64` Per elencare i pacchetti in una sottodirectory, esegui il comando seguente.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

Pulizia

Dopo il test, rimuovete la directory del progetto di test.

Per ripulire le risorse di test

- Rimuovi la directory del progetto di test.

  Su Linux emacOS, esegui il seguente comando.

  ```
  rm -rf package-test-env
  ```

  Su Windows (cmd), esegui il comando seguente.

  ```
  rmdir /s /q package-test-env
  ```

Compilazioni di debug

Se una compilazione fallisce, `rattler-build` conserva la directory di compilazione in modo da poter indagare. Esegui il comando seguente per aprire una shell interattiva nell'ambiente di compilazione con tutte le variabili di ambiente impostate com'erano durante la compilazione.

```
rattler-build debug shell
```

Dalla shell di debug, puoi modificare i file, eseguire singoli comandi di compilazione e aggiungere dipendenze per isolare il problema. Per ulteriori informazioni, consulta Debugging build nella documentazione di rattler-build.

Creazione di pacchetti per altre piattaforme

Il `rattler-build publish` comando crea pacchetti per il sistema operativo della workstation su cui viene eseguito il comando. Se la tua flotta Deadline Cloud utilizza un sistema operativo diverso dalla tua workstation o se il tuo pacchetto ha altri requisiti di host, hai le seguenti opzioni:

- Esegui `rattler-build publish` su un host che corrisponde al sistema operativo di destinazione. Ad esempio, usa un'istanza Amazon Elastic Compute Cloud (Amazon EC2) in Linux esecuzione per creare pacchetti per una flotta. Linux
- Usa una coda di creazione di pacchetti Deadline Cloud per automatizzare le build sulla piattaforma di destinazione. Vedi Creare una coda per la creazione di pacchetti.
- (Avanzato) Usa la compilazione incrociata per creare pacchetti per una piattaforma diversa dalla tua workstation. Per ulteriori informazioni, consulta la compilazione incrociata nella documentazione di rattler-build.

Fasi successive

Dopo aver pubblicato i pacchetti sul tuo canale Amazon S3 conda, configura le code di Deadline Cloud per utilizzare il canale:

- Configura le autorizzazioni della coda di produzione per pacchetti conda personalizzati: concedi alle tue code di produzione l'accesso in sola lettura al canale conda di Amazon S3.
- Aggiungi un canale conda a un ambiente di coda: configura l'ambiente di coda per installare i pacchetti dal canale conda di Amazon S3.

Configura le autorizzazioni per la coda di produzione per pacchetti conda personalizzati

La tua coda di produzione richiede autorizzazioni di sola lettura per il `/Conda` prefisso nel bucket S3 della coda. Apri la pagina AWS Identity and Access Management (IAM) per il ruolo associato alla coda di produzione e modifica la policy con quanto segue:

1. Apri la console Deadline Cloud e vai alla pagina dei dettagli della coda per la coda di compilazione del pacchetto.
2. Scegli il ruolo del servizio di coda, quindi scegli Modifica coda.
3. Scorri fino alla sezione Queue service role, quindi scegli Visualizza questo ruolo nella console IAM.
4. Dall'elenco delle politiche di autorizzazione, scegli quella AmazonDeadlineCloudQueuePolicyper la tua coda.
5. Dalla scheda Autorizzazioni, scegli Modifica.

6. Aggiungi una nuova sezione al ruolo del servizio di coda come segue. Sostituisci *amzn-s3-demo-bucket* e *111122223333* con il tuo bucket e il tuo account.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadOnly",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

Aggiungi un canale conda a un ambiente di coda

Per utilizzare il canale S3 conda, devi aggiungere la posizione del `s3://amzn-s3-demo-bucket/Conda/Default` canale al `CondaChannels` parametro dei lavori che invii a Deadline Cloud. I mittenti forniti con Deadline Cloud forniscono campi per specificare canali e pacchetti conda personalizzati.

Puoi evitare di modificare ogni lavoro modificando l'ambiente di coda conda per la tua coda di produzione. Attenersi alla seguente procedura:

1. Apri la console Deadline Cloud e vai alla pagina dei dettagli della coda per la coda di produzione.
2. Scegli la scheda Ambienti.
3. Seleziona l'ambiente di coda Conda, quindi scegli Modifica.
4. Scegli l'editor JSON, quindi nello script, trova la definizione del parametro per. `CondaChannels`
5. Modifica la riga `default: "deadline-cloud"` in modo che inizi con il canale conda S3 appena creato:

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Le flotte gestite dai servizi consentono una priorità flessibile dei canali per conda per impostazione predefinita. Per una richiesta di lavoro `blender=4.5` se Blender 4.5 è presente sia nel nuovo canale che nel canale, il pacchetto verrà estratto dal `deadline-cloud` canale che si trova per primo nell'elenco dei canali. Se una versione del pacchetto specificata non viene trovata nel primo canale, i canali successivi verranno controllati in ordine per verificare la versione del pacchetto.

Per le flotte gestite dai clienti, puoi abilitare l'uso dei pacchetti conda utilizzando uno degli esempi di ambiente conda queue nel repository di esempi di Deadline Cloud. GitHub

Crea un pacchetto conda per un'applicazione o un plug-in

Un pacchetto conda è un archivio compresso di software scritto in qualsiasi lingua. Conda supporta una varietà di combinazioni di sistemi operativi e architetture, quindi puoi impacchettare applicazioni complete come Blender e Nuke insieme a librerie per Python e altri linguaggi. Maya Per ulteriori informazioni sui pacchetti conda, vedere Pacchetti nella documentazione di conda.

Per utilizzare un pacchetto conda, lo si installa in un ambiente virtuale. Un ambiente virtuale conda ha una directory di prefissi in cui sono installati i pacchetti. L'installazione di un pacchetto utilizza il collegamento fisico o il reflink dei file quando è supportata, quindi la creazione di più ambienti con gli stessi pacchetti non utilizza molto spazio aggiuntivo su disco. Per utilizzare un ambiente virtuale, è necessario attivarlo per impostare le variabili di ambiente. L'attivazione esegue gli script forniti dai pacchetti, dando a ciascun pacchetto l'opportunità di modificare PATH o altre variabili di ambiente. I pacchetti Conda in genere contengono applicazioni o librerie, ma l'attivazione flessibile significa che possono anche indirizzare ad applicazioni installate su un file system condiviso.

La creazione di un pacchetto personalizzato prevede tre fasi: una ricetta contiene le istruzioni di compilazione, un pacchetto è l'elemento (*`.conda`*o *`.tar.bz2`* file) creato e un canale ospita i pacchetti per l'installazione. Il `rattler-build publish` comando gestisce tutti e tre i passaggi: può creare una ricetta in un pacchetto e pubblicarla su un canale, oppure può richiedere direttamente un elemento del pacchetto per pubblicarla.

La comunità conda-forge gestisce le ricette dei pacchetti per un'ampia gamma di software open source e ospita gli elementi dei pacchetti nel canale. `conda-forge` È possibile configurare la coda per includerla `conda-forge` come sorgente del pacchetto e quindi creare pacchetti personalizzati che dipendono dall'esecuzione dei pacchetti conda-forge. InfattiLinux, conda-forge ospita una

toolchain completa di compilatori che include il supporto CUDA, con opzioni di compilazione e collegamento coerenti selezionate. È possibile utilizzare i pacchetti conda-forge come dipendenze nelle proprie ricette o installarli insieme ai pacchetti personalizzati nello stesso ambiente.

È possibile combinare un'intera applicazione, comprese le dipendenze, in un pacchetto conda. I pacchetti che Deadline Cloud fornisce nel canale deadline-cloud per le flotte gestite dai servizi utilizzano questo approccio di riconfezionamento binario. Questo organizza gli stessi file di un'installazione per adattarli all'ambiente virtuale conda.

> **Note**
>
> Le applicazioni di grandi dimensioni possono richiedere decine di GB di spazio libero su disco per l'archivio di origine, i file estratti e l'output di compilazione. Assicurati di utilizzare un disco con spazio disponibile sufficiente per l'output di compilazione del pacchetto.

Package di un'applicazione

Quando si riconfeziona un'applicazione per conda, ci sono due obiettivi:

- La maggior parte dei file dell'applicazione deve essere separata dalla struttura principale dell'ambiente virtuale conda. Gli ambienti possono quindi mescolare l'applicazione con pacchetti provenienti da altre fonti come conda-forge.
- Quando viene attivato un ambiente virtuale conda, l'applicazione dovrebbe essere disponibile dalla variabile di ambiente PATH.

Per riconfezionare un'applicazione per conda

1. Scrivi ricette di compilazione conda che installano l'applicazione in una sottodirectory come. `$CONDA_PREFIX/opt/<application-name>` Questo lo separa dalle directory di prefissi standard come e. `bin` `lib`
2. Aggiungi collegamenti simbolici o avvia script per `$CONDA_PREFIX/bin` eseguire i binari dell'applicazione.

   In alternativa, create degli script.d attivati che il `conda activate` comando eseguirà per aggiungere le directory binarie dell'applicazione al PATH. Se invece Windows i collegamenti simbolici non sono supportati ovunque sia possibile creare ambienti, utilizzate invece gli script di avvio dell'applicazione o activate.d.

3. Alcune applicazioni dipendono da librerie non installate di default sulle flotte gestite dai servizi Deadline Cloud. Ad esempio, il sistema a finestre X11 di solito non è necessario per lavori non interattivi, ma alcune applicazioni richiedono comunque che venga eseguito senza un'interfaccia grafica. È necessario fornire tali dipendenze all'interno del pacchetto creato.
4. Se l'applicazione supporta i plugin, fornite una convenzione chiara che i pacchetti di plugin devono seguire per integrarsi con l'applicazione in un ambiente virtuale. Ad esempio, la ricetta di esempio del Maya 2026 documenta questa convenzione per Maya i plugin.
5. Assicurati di rispettare gli accordi di copyright e licenza per le applicazioni che includi. Ti consigliamo di utilizzare un bucket Amazon S3 privato per il tuo canale conda per controllare la distribuzione e limitare l'accesso dei pacchetti alla tua farm.

Le ricette di esempio per i pacchetti del `deadline-cloud` canale sono disponibili nell'archivio di esempi di Deadline Cloud su. GitHub

Package di un plugin

I plugin delle applicazioni possono essere impacchettati come pacchetti conda propri. Quando crei un pacchetto di plugin, segui queste linee guida:

- Includi il pacchetto dell'applicazione host sia come dipendenza di compilazione che di esecuzione nella ricetta `recipe.yaml` di compilazione. Utilizzate un vincolo di versione in modo che la ricetta di compilazione venga installata solo con pacchetti compatibili.
- Segui le convenzioni del pacchetto dell'applicazione host per la registrazione del plug-in.

Pacchetti adattatori

Alcune integrazioni di applicazioni Deadline Cloud utilizzano un adattatore che estende l'interfaccia dell'applicazione per semplificare la scrittura di modelli di lavoro. Un adattatore è un'interfaccia a riga di comando che supporta l'esecuzione di un demone in background, la segnalazione dello stato e l'applicazione della mappatura dei percorsi. Per ulteriori informazioni, vedere Open Job Description Adaptor Runtime suGitHub. Ad esempio, deadline-cloud-for-mayaon GitHub include una GUI integrata per l'invio dei lavori e un Maya adattatore disponibile come `maya-openjd` pacchetto nelle flotte gestite dai servizi.

Gli invii di lavoro da Deadline Cloud submitter GUIs includono un valore di `CondaPackages` parametro che specifica i pacchetti conda da includere in un ambiente virtuale per l'esecuzione del lavoro. Il valore del `CondaPackages` parametro per Maya in genere è simile `maya=2026.*`

`maya-openjd=0.15.* maya-mtoa` e potrebbe contenere voci alternative per i pacchetti di plug-in. Quando l'ambiente di coda configura un ambiente virtuale conda per l'esecuzione del job, risolve questi nomi di pacchetto e vincoli di versione in modo che siano compatibili e aggiunge tutti i pacchetti di dipendenza necessari per l'esecuzione. Ogni pacchetto di adattatori e plugin specifica con cosa è compatibile, incluse le versioni, le versioni di Maya Python e altre dipendenze.

Per creare i tuoi pacchetti adattatori usando i nostri esempi come la ricetta maya-openjd onGitHub, puoi basarti sui pacchetti per Python e altre dipendenze fornite da conda-forge. Potrebbe essere necessario prima definire la scadenza e le ricette per soddisfare le dipendenze. openjd-adaptor-runtime

Crea una ricetta di costruzione di conda per Blender

Blenderè gratuito da usare e semplice da impacchettare con conda, il che lo rende un buon punto di partenza per imparare a creare pacchetti conda per Deadline Cloud (Deadline Cloud). AWS La Blender Foundation fornisce archivi di applicazioni per più sistemi operativi. La ricetta di esempio Blender 4.5 nel repository di esempi di Deadline Cloud su questi archivi racchiude questi archivi in un GitHub pacchetto conda.

Comprendere la ricetta

Il file recipe.yaml definisce i metadati del pacchetto, l'origine e le opzioni di compilazione nella sintassi del modello URLs rattler-build. La ricetta specifica il numero di versione una sola volta e fornisce una fonte diversa in base al sistema operativo. URLs

La `build` sezione in `recipe.yaml` disattiva i controlli di riposizionamento binario e di collegamento di oggetti condivisi dinamici (DSO). Queste opzioni controllano il funzionamento del pacchetto quando viene installato in un ambiente virtuale conda con qualsiasi prefisso di directory. I valori predefiniti utilizzati nella `build` sezione sono progettati per impacchettare ogni libreria di dipendenze separatamente, ma quando si riconfeziona un'applicazione in formato binario, è necessario modificarli. Blendernon richiede alcuna regolazione RPATH perché gli archivi dell'applicazione sono progettati pensando alla rilocabilità. Vedi Creare una ricetta conda per Maya per un esempio di aggiunta di rilocabilità.

Durante la compilazione del pacchetto, viene eseguito lo script build.sh o build_win.sh per installare i file nell'ambiente. Questi script copiano i file di installazione in`$PREFIX/opt/blender`, creano collegamenti simbolici da `$PREFIX/bin` (onLinux) e impostano script di attivazione che configurano variabili di ambiente come. `BLENDER_LOCATION` AttivatoWindows, lo script di attivazione aggiunge la Blender directory al PATH invece di creare collegamenti simbolici.

Lo script di Windows compilazione utilizza `bash` invece di un `cmd.exe` file.bat per garantire la coerenza tra le piattaforme. Puoi installare git for `bash` per provvedere Windows alla creazione di pacchetti.

La ricetta include anche un `deadline-cloud.yaml` file che specifica le piattaforme e i metadati conda per l'invio di lavori automatici di creazione di pacchetti a Deadline Cloud. Per ulteriori informazioni, consulta Inviare un processo di creazione del pacchetto.

Compilazione del Blender pacchetto

`rattler-build publish`Utilizzatelo per creare la ricetta Blender 4.5 e pubblicare il pacchetto su un canale. Puoi pubblicare su un canale di filesystem locale per i test o direttamente su un canale Amazon S3 per l'uso in produzione. Se hai completato la configurazione in Compila e testa i pacchetti localmente, esegui il seguente comando dalla directory. `conda_recipes`

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Per altre opzioni di pubblicazione:

- Per pubblicare su un canale Amazon S3, consulta Pubblicare pacchetti su un canale S3 conda.
- Per automatizzare le build utilizzando una coda per la creazione di pacchetti Deadline Cloud, consulta Automatizzare le build di pacchetti con Deadline Cloud.

Testa il tuo pacchetto con un lavoro di rendering Blender

Dopo aver creato il pacchetto Blender 4.5, puoi testarlo con un render job. Se non avete una Blender scena, scaricate la scena Blender 3.5 - Cozy Kitchen dalla pagina dei file Blender dimostrativi. Il repository di esempi di Deadline Cloud contiene un `blender_render` job bundle e un ambiente di coda conda che puoi utilizzare sia per i test locali che su cloud.

Test in locale

È possibile eseguire il modello di lavoro sulla workstation utilizzando l'Open Job Description CLI. Installa la CLI con. `pip`

```
pip install openjd-cli
```

Dalla `job_bundles` directory del repository degli esempi, esegui il comando seguente. *`/path/to/scene.blend`*Sostituitelo con il percorso del file di Blender scena.

```
openjd run blender_render/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages=blender=4.5 \
    -p CondaChannels=file://$HOME/my-conda-channel \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

L'`--environment`opzione applica l'ambiente di coda conda, che crea un ambiente virtuale conda con i pacchetti specificati in. `CondaPackages` Il `CondaChannels` parametro indica all'ambiente di coda dove trovare i pacchetti. Se hai pubblicato su un canale Amazon S3 anziché su un canale locale, sostituisci il `file://` percorso con l'URL del `s3://` canale.

Test su Deadline Cloud

Dopo aver configurato la coda di produzione per utilizzare il canale conda di Amazon S3, puoi inviare il lavoro di rendering a Deadline Cloud. Dalla `job_bundles` directory nel repository degli esempi, esegui il comando seguente.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Utilizza il monitor Deadline Cloud per monitorare lo stato di avanzamento del lavoro. Nel monitor, seleziona l'attività per il lavoro e scegli Visualizza registri. Seleziona l'azione Launch Conda session per verificare che il pacchetto sia stato trovato nel canale Amazon S3.

Crea una ricetta di costruzione di conda per Autodesk Maya

Applicazioni commerciali come ad esempio Autodesk Maya introducono requisiti di imballaggio aggiuntivi rispetto ad applicazioni open source come. Blender La Blenderricetta racchiude un semplice archivio riposizionabile con una licenza open source. Le applicazioni commerciali sono spesso distribuite tramite programmi di installazione e richiedono la configurazione della gestione delle licenze.

Considerazioni per le applicazioni commerciali

Le seguenti considerazioni si applicano al confezionamento di applicazioni commerciali. I dettagli illustrano come ciascuno si applica a. Maya

- Licenze: comprendi i diritti e le restrizioni di licenza dell'applicazione. Potrebbe essere necessario configurare un sistema di gestione delle licenze. Leggi le domande frequenti sui vantaggi dell'Autodeskabbonamento su Cloud Rights per comprendere a cosa servono i diritti cloudMaya. Autodeski prodotti si basano su un `ProductInformation.pit` file che in genere richiede l'accesso dell'amministratore per la configurazione. Le caratteristiche del prodotto per i thin client offrono un'alternativa trasferibile. Per ulteriori informazioni, consulta Thin Client Licensing for Maya. MotionBuilder
- Dipendenze delle librerie di sistema: alcune applicazioni dipendono da librerie non installate su host di fleet worker gestiti dal servizio. Mayadipende dalle librerie tra cui freetype e fontconfig. Quando queste librerie sono disponibili nel gestore di pacchetti di sistema, ad esempio `dnf` for AL2023, puoi usare il gestore di pacchetti come sorgente. Poiché i pacchetti RPM non sono progettati per essere rilocabili, è necessario utilizzare strumenti come quelli per risolvere le dipendenze `patchelf` all'interno del prefisso di installazione. Maya
- Accesso amministratore per l'installazione: alcuni programmi di installazione richiedono l'accesso come amministratore. Le flotte gestite dai servizi non forniscono l'accesso come amministratore, quindi è necessario installare l'applicazione su un sistema separato e creare un archivio dei file per la build del pacchetto. Il Windows programma di installazione richiede questo approccio. Maya Il README.md nella ricetta documenta una procedura ripetibile che utilizza un'istanza Amazon Elastic Compute Cloud (Amazon EC2) appena lanciata.
- Integrazione con plug-in: il Maya pacchetto di esempio definisce l'isolamento dell'applicazione dalla configurazione `MAYA_NO_HOME=1` a livello utente e aggiunge percorsi di ricerca dei moduli in `MAYA_MODULE_PATH` modo che i pacchetti di plug-in possano posizionare i file all'interno dell'ambiente virtuale. `.mod` Vedi la ricetta di esempio del Maya 2026 per la convenzione completa sull'integrazione dei plugin.

Comprendere la ricetta

Il file recipe.yaml definisce i metadati del pacchetto nella sintassi del modello rattler-build. Esamina le seguenti sezioni del file:

- source: fa riferimento agli archivi dell'installer, incluso l'hash sha256. SìLinux, la fonte è l'archivio del programma di installazione. Autodesk SìWindows, il codice sorgente include sia l'archivio del

programma di installazione sia uno `cleanMayaForCloud.py` script Autodesk che Maya prepara la distribuzione su cloud. Aggiorna gli hash quando modifichi i file sorgente, ad esempio quando crei il pacchetto di una nuova versione.

- build — Disattiva i controlli predefiniti di rilocazione binaria e collegamento DSO perché i meccanismi automatici non funzionano correttamente per la libreria e le directory binarie utilizzate. Maya SìLinux, la ricetta include `patchelf` come build la dipendenza per impostare manualmente il relativo. RPATHs
- about — Metadati relativi all'applicazione per la navigazione o l'elaborazione dei contenuti di un canale conda.

Gli script di compilazione (build.sh perLinux, build_win.sh perWindows) includono commenti che spiegano ogni passaggio. Gli script eseguono le seguenti attività chiave:

- Estrai il programma di installazione: estrae i file di Maya installazione nel prefisso conda. Gli Windows script Linux and lo gestiscono in modo diverso a causa dei formati di installazione. Vedi gli script di compilazione per i dettagli.
- Installa le dipendenze delle librerie di sistema: Linux attivato, lo script scarica ed estrae le librerie di sistema Maya necessarie ma che non sono presenti negli host del parco veicoli gestiti dai servizi. Lo script copia queste librerie nella Maya `lib` directory in modo che siano disponibili nell'ambiente conda.
- Set relative RPATHs con patchelf — OnLinux, lo script utilizza `patchelf --add-rpath` per aggiungere percorsi `$ORIGIN` -relative alle librerie condivise. Questo approccio segue la raccomandazione di conda di non `LD_LIBRARY_PATH` utilizzarlo mai in ambienti conda. Lo script corregge le librerie a più livelli di directory (`lib`,`lib/python*/site-packages`,`lib/python*/lib-dynload`) in modo che ogni libreria possa trovare le proprie dipendenze rispetto alla propria posizione. La ricetta segue la migliore pratica dell'impostazione `DT_RUNPATH` invece di`DT_RPATH`, che consente di sovrascrivere il percorso di ricerca quando necessario `LD_LIBRARY_PATH` per il debug.
- Configura le licenze thin client: lo script imposta le licenze thin client come documentato in Autodesk modo che il `ProductInformation.pit` file possa essere collocato all'interno dell'ambiente conda anziché richiedere l'accesso dell'amministratore a livello di sistema.
- Configurazione degli script di attivazione: gli script creano script di attivazione e disattivazione che impostano variabili di ambiente tra cui,, e. `MAYA_LOCATION` `MAYA_VERSION` `MAYA_NO_HOME` `MAYA_MODULE_PATH` SìWindows, gli script producono entrambi i file di `.bat` attivazione perché

gli `.sh` ambienti di coda di esempio di Deadline Cloud li utilizzano per attivare gli ambienti. `bash` Windows

Creazione del pacchetto Maya

Prima di creare il Maya pacchetto, scarica il Maya programma di installazione dal tuo Autodesk account. Ad Linux esempio, posiziona l'archivio direttamente nella `conda_recipes/archive_files` directory. Per creare l'archivioWindows, segui la procedura descritta nel file README.md.

Utilizzatelo `rattler-build publish` per creare e pubblicare il pacchetto. La Maya ricetta richiede `patchelf` come build una dipendenza daLinux, disponibile presso conda-forge. Aggiungi `-c conda-forge` per rendere disponibile la dipendenza durante la compilazione. Dalla `conda_recipes` directory, esegui il comando seguente.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Per altre opzioni di pubblicazione:

- Per pubblicare su un canale Amazon S3, consulta Pubblicare pacchetti su un canale S3 conda.
- Per automatizzare le build utilizzando una coda per la creazione di pacchetti Deadline Cloud, consulta Automatizzare le build di pacchetti con Deadline Cloud. Per creare entrambi i Windows pacchetti, usa Linux l'opzione con lo script. `--all-platforms` `submit-package-job`

Per renderizzare l'esempio del giradischi con Maya eArnold, compila sia il pacchetto MtoAplugin che quello Mayaadaptor. Dopo aver pubblicato tutti e tre i pacchetti, puoi inviare un lavoro di rendering di prova utilizzando il turntable conMaya/Arnoldjob bundle dal repository degli esempi di Deadline Cloud. Vedi Testa i tuoi pacchetti con un render job Maya.

Crea una ricetta di costruzione conda per l'adattatore Maya

Il `maya-openjd` pacchetto fornisce l'adattatore che si integra Maya con gli invii di lavoro di AWS Deadline Cloud (Deadline Cloud). Quando invii un lavoro di Maya rendering utilizzando una GUI del mittente di Deadline Cloud, il parametro viene incluso insieme al pacchetto. `CondaPackages` `maya-`

`openjd maya` L'adattatore gestisce l'avvioMaya, la comunicazione dei parametri di rendering e la gestione del ciclo di vita dell'applicazione durante una sessione di lavoro. Per ulteriori informazioni sugli adattatori, vedete Pacchetti adattatori.

Comprendere la ricetta

La ricetta di esempio maya-openjd crea l'adattatore dal pacchetto sorgente deadline-cloud-for-mayapubblicato su PyPI. Il recipe.yaml installa il pacchetto utilizzando l'ambiente conda. `pip`

La ricetta dipende da Python e da altri due pacchetti del repository di esempi di Deadline Cloud che devi prima creare:

- deadline — La libreria client di Deadline Cloud.
- openjd-adaptor-runtime— Il runtime dell'adattatore Open Job Description.

Python e altre dipendenze sono disponibili da conda-forge, quindi aggiungili `-c conda-forge` al `rattler-build publish` comando quando compili il pacchetto adaptor.

Compilazione del pacchetto adaptor

Il `maya-openjd` pacchetto dipende da altri due pacchetti del repository di esempi di Deadline Cloud. Crea tutti e tre i pacchetti in ordine dalla `conda_recipes` directory. L'`-c conda-forge`opzione su ogni comando è soddisfare le dipendenze delle ricette per le librerie Python e Python.

Compila il pacchetto. `deadline`

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Costruisci il `openjd-adaptor-runtime` pacchetto.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Costruisci il `maya-openjd` pacchetto.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Per altre opzioni di pubblicazione:

- Per pubblicare su un canale Amazon S3, consulta Pubblicare pacchetti su un canale S3 conda.
- Per automatizzare le build utilizzando una coda per la creazione di pacchetti Deadline Cloud, consulta Automatizzare le build di pacchetti con Deadline Cloud.

Crea una ricetta di compilazione conda per il plugin Autodesk Maya to Arnold (MtoA)

Il Maya to Arnold (MtoA) plugin aggiunge il Arnold renderer come opzione all'interno. Maya La ricetta di esempio MToA dimostra come impacchettare un plug-in come pacchetto conda separato che si integra con il pacchetto dell'applicazione host.

Comprendere la ricetta

Il file recipe.yaml specifica una dipendenza dal `maya` pacchetto per i requisiti di compilazione ed esecuzione. Questa dipendenza utilizza un vincolo di versione in modo che il plugin venga installato solo con una versione compatibile. Maya

La ricetta utilizza gli stessi archivi di origine della ricetta. Maya Lo script di compilazione installa MtoA e crea un `mtoa.mod` file nella `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` directory in cui è configurato il Maya pacchetto. `MAYA_MODULE_PATH` Arnolde Maya utilizzano la stessa tecnologia di licenza, quindi il Maya pacchetto include già le informazioni di licenza necessarie. Arnold

Creazione del pacchetto MtoA

Compila il Maya pacchetto prima di creare il MtoA pacchetto, perché MtoA dipende dal momento Maya della compilazione. Si usa `rattler-build publish` per creare e pubblicare il pacchetto. Dalla `conda_recipes` directory, esegui il seguente comando.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
```

```
    --build-number=+1
```

Il `rattler-build publish` comando utilizza il canale di destinazione come canale con la massima priorità per la risoluzione delle dipendenze, quindi il `maya` pacchetto pubblicato in precedenza è disponibile automaticamente.

Per altre opzioni di pubblicazione:

- Per pubblicare su un canale Amazon S3, consulta Pubblicare pacchetti su un canale S3 conda.
- Per automatizzare le build utilizzando una coda per la creazione di pacchetti Deadline Cloud, consulta Automatizzare le build di pacchetti con Deadline Cloud.

Metti alla prova i tuoi pacchetti con un lavoro di rendering Maya

Dopo aver creato i `maya-openjd` pacchetti MayaMtoA, e, puoi testarli con un lavoro di rendering. L'archivio di esempi di Deadline Cloud contiene un giradischi conMaya/Arnoldjob bundle che esegue il rendering di un'animazione utilizzando and. Maya Arnold Il job bundle serve anche FFmpeg per codificare un video, disponibile sul canale. `conda-forge`

Test in locale

È possibile eseguire il modello di lavoro sulla workstation utilizzando l'Open Job Description CLI. Installa la CLI con. `pip`

```
pip install openjd-cli
```

Dalla `job_bundles` directory del repository degli esempi, esegui il comando seguente. Il `ErrorOnArnoldLicenseFail=false` parametro indica Arnold di eseguire il rendering con filigrane anziché fallire quando non è disponibile alcuna licenza.

```
openjd run turntable_with_maya_arnold/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages="maya maya-mtoa maya-openjd ffmpeg" \
    -p CondaChannels="file://$HOME/my-conda-channel conda-forge" \
    -p ErrorOnArnoldLicenseFail=false \
    -p FrameRange=1-5
```

L'`--environment`opzione applica l'ambiente di coda conda, che crea un ambiente virtuale conda con i pacchetti specificati in. `CondaPackages` Il `CondaChannels` parametro include sia il canale

locale per i pacchetti personalizzati che per. `conda-forge` `ffmpeg` Se hai pubblicato su un canale Amazon S3 anziché su un canale locale, sostituisci il `file://` percorso con l'URL del `s3://` canale.

Al termine del processo, l'output renderizzato si trova nella directory. `turntable_with_maya_arnold/output/`

Test su Deadline Cloud

Dopo aver configurato la coda di produzione per utilizzare il canale conda di Amazon S3, invia il lavoro di rendering a Deadline Cloud. Aggiungi il `conda-forge` canale al `CondaChannels` parametro nel tuo ambiente di coda conda per fornire una fonte `ffmpeg` e le dipendenze Python richieste dall'adattatore. Dalla `job_bundles` directory nel repository degli esempi, esegui il comando seguente.

```
deadline bundle submit turntable_with_maya_arnold
```

Utilizza il monitor Deadline Cloud per monitorare lo stato di avanzamento del lavoro. Nel monitor, seleziona l'attività per il lavoro e scegli Visualizza registri. Seleziona l'azione Launch Conda session per verificare che i `maya-openjd` pacchetti `mayamaya-mtoa`, e siano stati trovati nel canale Amazon S3.

Automatizza la creazione di pacchetti con Deadline Cloud

Per i CI/CD flussi di lavoro o quando devi creare pacchetti per più sistemi operativi, puoi creare una coda per la creazione di pacchetti Deadline Cloud. Le pianificazioni delle code creano lavori sulla tua flotta, che creano i pacchetti e li pubblicano sul tuo canale conda Amazon Simple Storage Service (Amazon S3). Ciò semplifica il mantenimento di build continue di pacchetti per le versioni software in tutte le configurazioni richieste.

Puoi creare una coda per la creazione di pacchetti utilizzando un modello AWS CloudFormation (CloudFormation) o manualmente dalla console Deadline Cloud. Il CloudFormation modello implementa una farm completa con una coda di produzione e una coda di creazione di pacchetti già configurate. La creazione della coda dalla console offre un maggiore controllo sulle singole impostazioni.

Crea una coda per la creazione di pacchetti con CloudFormation

Puoi utilizzare un CloudFormation modello per creare una Deadline Cloud farm che includa una coda per la creazione di pacchetti. Il modello configura una coda di produzione e una coda di creazione di pacchetti con un canale conda privato Amazon S3.

Prima di distribuire il modello, crea un bucket Amazon S3 per contenere gli allegati dei lavori e il tuo canale conda. Puoi creare un bucket dalla console Amazon S3. È necessario il nome del bucket quando si distribuisce il modello.

Per distribuire il modello CloudFormation

1. Scarica il deadline-cloud-starter-farmmodello -template.yaml dal repository degli esempi di Deadline Cloud su. GitHub
2. Dalla CloudFormation console, scegli Create Stack, quindi Con nuove risorse (standard).
3. Seleziona l'opzione per caricare un file modello, quindi carica il `deadline-cloud-starter-farm-template.yaml` file.
4. Inserisci un nome per lo stack, ad esempio**`StarterFarm`**, e fornisci il nome di un bucket Amazon S3 per gli allegati dei lavori e il canale conda.
5. Segui i passaggi della CloudFormation console per completare la creazione dello stack.

Per ulteriori informazioni sui parametri del modello e sulle opzioni di personalizzazione, consulta lo starter farm README nell'archivio degli esempi di Deadline Cloud su. GitHub

Crea una coda per la creazione di pacchetti dalla console

Segui le istruzioni riportate in Creare una coda nella Guida per l'utente di Deadline Cloud. Apporta le seguenti modifiche:

- Nel passaggio 5, scegli un bucket Amazon S3 esistente. Specificate il nome della cartella principale **`DeadlineCloudPackageBuild`** in modo che gli artefatti della build rimangano separati dai normali allegati di Deadline Cloud.
- Nella fase 6, puoi associare la coda per la creazione dei pacchetti a una flotta esistente oppure puoi creare una flotta completamente nuova se la flotta attuale non è adatta.
- Nella fase 9, create un nuovo ruolo di servizio per la coda di creazione dei pacchetti. Modificherai le autorizzazioni per fornire alla coda le autorizzazioni necessarie per caricare pacchetti e reindicizzare un canale conda.

Configura i permessi di creazione della coda del pacchetto

Per consentire alla coda di creazione dei pacchetti di accedere al `/Conda` prefisso nel bucket Amazon S3 della coda, devi modificare il ruolo della coda per consentirle l'accesso. read/write Il ruolo

richiede le seguenti autorizzazioni in modo che i processi di creazione dei pacchetti possano caricare nuovi pacchetti e reindicizzare il canale.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Apri la console Deadline Cloud e vai alla pagina dei dettagli della coda per la coda di compilazione del pacchetto.
2. Scegli il ruolo del servizio di coda, quindi scegli Modifica coda.
3. Scorri fino alla sezione Queue service role, quindi scegli Visualizza questo ruolo nella console IAM.
4. Dall'elenco delle politiche di autorizzazione, scegli quella AmazonDeadlineCloudQueuePolicyper la tua coda.
5. Dalla scheda Autorizzazioni, scegli Modifica.
6. Aggiungi una nuova sezione al ruolo del servizio di coda come segue. Sostituisci *amzn-s3-demo-bucket* e *111122223333* con il tuo bucket e il tuo account.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadWrite",
   "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
```

```
    }
   }
  },
```

Invia un processo di compilazione del pacchetto

Dopo aver creato una coda per la creazione di pacchetti e configurato i permessi di coda, puoi inviare lavori per creare pacchetti conda. Lo `submit-package-job` script nel repository di esempi di Deadline Cloud GitHub invia un lavoro di compilazione per una ricetta conda.

È necessario quanto segue:

- La CLI Deadline Cloud installata sulla tua workstation.
- Una sessione di accesso attiva di AWS Deadline Cloud monitor (Deadline Cloud monitor).
- Un clone del repository di esempi di Deadline Cloud.

Per inviare un pacchetto, build job.

1. Apri la GUI di configurazione di Deadline Cloud e imposta la farm e la coda predefinite nella coda di creazione dei pacchetti.

   ```
   deadline config gui
   ```

2. Passa alla `conda_recipes` directory nel repository degli esempi.

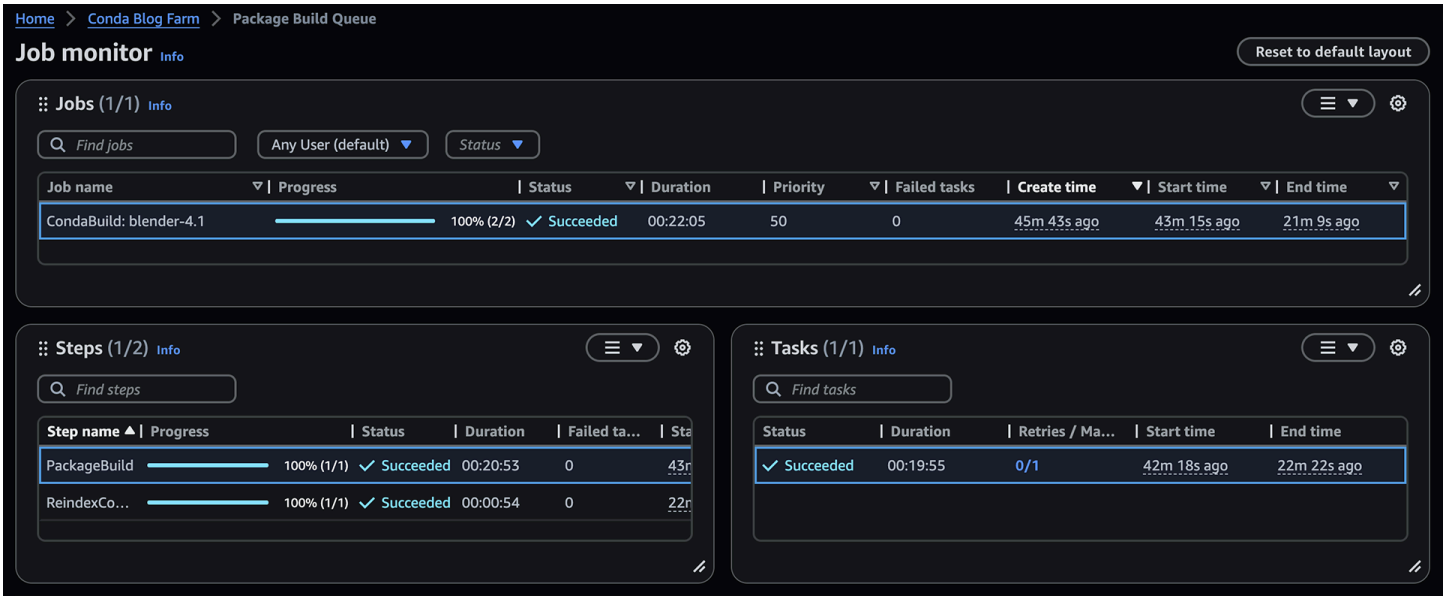
   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Esegui lo `submit-package-job` script con la directory delle ricette. L'esempio seguente crea la ricetta Blender 4.5.

   ```
   ./submit-package-job blender-4.5/
   ```

   Se la ricetta richiede un archivio sorgente che non è stato ancora scaricato, lo script fornisce le istruzioni per il download. Scaricate l'archivio ed eseguite nuovamente lo script.

Dopo aver inviato il lavoro, utilizza il monitor Deadline Cloud per visualizzare l'avanzamento e lo stato del lavoro.



Il monitor mostra le due fasi del lavoro: la creazione del pacchetto e la reindicizzazione del canale conda. Quando fai clic con il pulsante destro del mouse sull'attività per la fase di creazione del pacchetto e scegli Visualizza registri, il monitor mostra le azioni della sessione:

- Sincronizzazione degli allegati: copia gli allegati del processo di input o installa un file system virtuale.
- Avvia Conda — The Queue Environment Action. Il processo di compilazione non specifica i pacchetti conda, quindi questa azione termina rapidamente.
- Launch CondaBuild Env: crea un ambiente virtuale conda con il software necessario per creare un pacchetto conda e reindicizzare un canale.
- Task run: crea il pacchetto e carica i risultati su Amazon S3.

Man mano che le azioni vengono eseguite, inviano i log ad Amazon CloudWatch (CloudWatch). Quando un lavoro è completo, seleziona Visualizza i registri di tutte le attività per visualizzare registri aggiuntivi sulla configurazione e lo smontaggio dell'ambiente.

Esegui script di configurazione host con privilegi di amministratore

Gli script di configurazione dell'host consentono di eseguire attività amministrative, come l'installazione del software, sui dipendenti della flotta gestiti dal servizio. Questi script vengono eseguiti con privilegi elevati (`sudo`attivoLinux, amministratore attivoWindows), offrendoti la flessibilità necessaria per configurare i tuoi lavoratori per il tuo sistema.

Deadline Cloud esegue lo script dopo che il lavoratore è entrato nello `STARTING` stato e prima di eseguire qualsiasi attività.

> **Important**
>
> Lo script viene eseguito con autorizzazioni elevate. È responsabilità dell'utente assicurarsi che lo script non presenti problemi di sicurezza.
> Quando utilizzi uno script di configurazione dell'host, sei responsabile del monitoraggio dello stato della tua flotta.

Gli usi comuni degli script di configurazione dell'host includono:

- Installazione di software che richiede l'accesso da amministratore
- Installazione di Docker contenitori
- Installazione di soluzioni di archiviazione cloud di terze parti comeLucidLink. Per una procedura dettagliata, consulta Configurazione LucidLink con script di flotta gestiti dal servizio per Deadline Cloud sul blog for M&E. AWS

Puoi creare e aggiornare uno script di configurazione dell'host utilizzando la console o utilizzando il. AWS CLI

Console

1. Nella pagina dei dettagli della flotta, scegli la scheda Configurazioni.
2. Nel campo Script, inserisci lo script da eseguire con autorizzazioni elevate. Puoi scegliere Importa per caricare uno script dalla tua workstation.
3. Imposta un periodo di timeout in secondi per l'esecuzione dello script. Il valore predefinito è 300 secondi (5 minuti).
4. Scegli Salva modifiche per salvare lo script.

Create with CLI

Usa il seguente AWS CLI comando per creare una flotta con uno script di configurazione dell'host. Sostituisci il *`placeholder`* testo con le tue informazioni.

```
aws deadline create-fleet \
```

```
--farm-id farm-12345 \
--display-name "fleet-name" \
--max-worker-count 1 \
--configuration '{
"serviceManagedEc2": {
  "instanceCapabilities": {
    "vCpuCount": {"min": 2},
    "memoryMiB": {"min": 4096},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
  },
  "instanceMarketOptions": {"type":"spot"}
}
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Update with CLI

Usa il seguente AWS CLI comando per aggiornare lo script di configurazione dell'host di una flotta. Sostituisci il *placeholder* testo con le tue informazioni.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

I seguenti script dimostrano:

- Le variabili di ambiente disponibili per lo script
- Queste AWS credenziali funzionano nella shell
- Che lo script sia in esecuzione in una shell con privilegi elevati

Linux

Utilizzate lo script seguente per dimostrare che uno script è in esecuzione con `root` privilegi:

```
# Print environment variables
```

```
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Utilizzate lo PowerShell script seguente per dimostrare che uno script è in esecuzione con privilegi di amministratore:

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
  $currentUser = New-Object
 Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
  $isAdmin =
 $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

  return $isAdmin
}

if (Test-AdminPrivileges) {
  Write-Host "The current PowerShell session is elevated (running as
 Administrator)."
} else {
  Write-Host "The current PowerShell session is not elevated (not running as
 Administrator)."
}
exit 0
```

Risolvi i problemi relativi agli script di configurazione dell'host

Quando esegui lo script di configurazione dell'host:

- In caso di successo: il lavoratore esegue il lavoro
- In caso di errore (codice di uscita diverso da zero o arresto anomalo):
  - Il lavoratore si spegne

  La flotta avvia automaticamente un nuovo lavoratore utilizzando lo script di configurazione dell'host più recente

Per monitorare lo script:

1. Apri la pagina della flotta nella console Deadline Cloud.
2. Scegli Visualizza lavoratori per aprire il monitor Deadline Cloud.
3. Visualizza lo stato del lavoratore nella pagina di monitoraggio.

> **Tip**
>
> Quando testate gli script di configurazione dell'host, impostate il numero massimo di lavoratori del parco macchine su 1 per evitare di avviare più lavoratori durante l'iterazione dello script.

Note importanti:

- I lavoratori che si fermano a causa di un errore non sono disponibili nell'elenco dei lavoratori nel monitor. Utilizzate CloudWatch Logs per visualizzare i registri dei lavoratori nel seguente gruppo di registri:

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  All'interno di quel gruppo di log, cerca uno stream denominato. `worker-ZZZZZ`
- CloudWatch Logs conserva i registri dei lavoratori in base al periodo di conservazione configurato.

Monitora l'esecuzione dello script di configurazione dell'host

Con gli script di configurazione dell'host, puoi assumere il pieno controllo di un operatore di Deadline Cloud. Puoi installare qualsiasi pacchetto software, riconfigurare i parametri del sistema operativo o montare file system condivisi. Con questa funzionalità avanzata e la capacità di Deadline Cloud di scalare fino a migliaia di lavoratori, puoi monitorare se gli script di configurazione vengono eseguiti correttamente o se hanno avuto esito negativo.

Consigliamo le seguenti soluzioni per monitorare l'esecuzione degli script di configurazione dell'host.

CloudWatch Monitoraggio dei log

Tutti i log di configurazione del fleet host vengono trasmessi in streaming al gruppo di CloudWatch log della flotta e in particolare al flusso di log di un lavoratore. CloudWatch Ad esempio, `/aws/`

`deadline/farm-123456789012/fleet-777788889999` è il gruppo di log per l'azienda agricola`123456789012`, la flotta. `777788889999`

Ad esempio, ogni lavoratore fornisce un flusso di log dedicato`worker-123456789012`. I log di configurazione dell'host includono banner di log come Running Host Configuration Script e Finished running Host Configuration Script, codice di uscita: 0. Il codice di uscita dello script è incluso nel banner finito e può essere interrogato utilizzando gli strumenti. CloudWatch

CloudWatch Logs Insights

CloudWatch Logs Insights offre funzionalità avanzate per analizzare le informazioni di registro. Ad esempio, la seguente query di Log Insights analizza il codice di uscita della configurazione host, ordinato per ora:

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

Per ulteriori informazioni su CloudWatch Logs Insights, consulta Analyzing log data with CloudWatch Logs Insights nella Amazon CloudWatch Logs User Guide.

Registrazione strutturata degli agenti di lavoro

Il worker agent di Deadline Cloud pubblica log JSON strutturati su. CloudWatch L'agente di lavoro offre un'ampia gamma di log strutturati per l'analisi dello stato di salute dei lavoratori. Per ulteriori informazioni, consulta Accesso dell'agente di lavoro di Deadline Cloud. GitHub

Gli attributi dei log strutturati vengono decompressi nei campi di Log Insights. È possibile utilizzare questa CloudWatch funzionalità per contare e analizzare gli errori di avvio della configurazione dell'host. Ad esempio, è possibile utilizzare una query count and bin per determinare la frequenza con cui si verificano gli errori:

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

CloudWatch filtri metrici per metriche e allarmi

Puoi configurare filtri metrici per CloudWatch generare metriche dai log. CloudWatch I filtri metrici consentono di creare allarmi e dashboard per monitorare l'esecuzione degli script di configurazione dell'host.

Per creare un filtro parametrico

1. Apri la console. CloudWatch
2. Nel riquadro di navigazione, scegli Registri, quindi Gruppi di log.
3. Seleziona il gruppo di log del tuo parco veicoli.
4. Scegli Crea filtro parametri.
5. Definite lo schema di filtro utilizzando uno dei seguenti metodi:

   - Per le metriche di successo:

   ```
   {$.message = "*Worker Agent host configuration succeeded.*"}
   ```

   - Per le metriche degli errori:

   ```
   {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
    exit code*"}
   ```

6. Scegli Avanti per creare una metrica con i seguenti valori:

   - Spazio dei nomi metrico: lo spazio dei nomi delle metriche (ad esempio,) **`MyDeadlineFarm`**
   - Nome della metrica: il nome della metrica richiesto (ad esempio,) **`host_config_failure`**
   - Valore della metrica: **`1`** (ogni istanza corrisponde a 1)
   - Valore predefinito: lascia vuoto
   - Unità: **`Count`**

Dopo aver creato i filtri metrici, puoi configurare gli CloudWatch allarmi standard per intervenire in caso di elevati tassi di errore della configurazione host o aggiungere le metriche a una CloudWatch dashboard per day-to-day le operazioni e il monitoraggio.

Per ulteriori dettagli, consulta Filter and pattern syntax nella Amazon CloudWatch Logs User Guide.

Utilizzo delle licenze software con Deadline Cloud

Deadline Cloud offre due metodi per fornire licenze software per i tuoi lavori:

- Licenze basate sull'utilizzo (UBL): traccia e fattura in base al numero di ore impiegate dalla flotta per elaborare un lavoro. Non esiste un numero prestabilito di licenze, quindi la flotta può essere ampliata in base alle esigenze. UBL è lo standard per le flotte gestite dai servizi. Per le flotte gestite dai clienti, puoi connettere un endpoint di licenza Deadline Cloud per UBL. UBL fornisce licenze per il rendering ai dipendenti di Deadline Cloud, non fornisce licenze per le applicazioni DCC.
- Bring your own license (BYOL): consente di utilizzare le licenze software esistenti con le flotte gestite dai servizi o dai clienti. Puoi utilizzare BYOL per connetterti ai server di licenza per software non supportato dalle licenze basate sull'utilizzo di Deadline Cloud. Puoi utilizzare BYOL con flotte gestite dai servizi connettendoti a un server di licenze personalizzato.

Argomenti

- Combinazione di BYOL e UBL
- Connect flotte gestite dai servizi a un server di licenze personalizzato
- Connect le flotte gestite dai clienti a un endpoint di licenza

Combinazione di BYOL e UBL

Puoi combinare BYOL e UBL in modo che i tuoi dipendenti utilizzino prima le licenze esistenti e, una volta esaurite, ritornino automaticamente alle licenze basate sull'utilizzo di Deadline Cloud. Questo approccio è utile quando si dispone di un numero limitato di licenze esistenti ma è necessario ampliare tale capacità durante i picchi di carico di lavoro.

Come funzionano le licenze combinate

Quando si configurano le licenze combinate, l'ambiente di coda imposta le variabili di ambiente di licenza in modo che il server delle licenze BYOL sia elencato prima dell'endpoint delle licenze UBL. La maggior parte delle applicazioni di terze parti controlla i server di licenza nell'ordine in cui appaiono nella variabile di ambiente. Quando un lavoratore richiede una licenza, l'applicazione contatta innanzitutto il server delle licenze BYOL. Se non è disponibile alcuna licenza BYOL, l'applicazione torna all'endpoint della licenza UBL.

Il modello di ambiente di coda BYOL fornito in configura automaticamente questo comportamento di fallback. Connect flotte gestite dai servizi a un server di licenze personalizzato Lo script Python nell'ambiente di coda antepone l'indirizzo del server di licenza BYOL alle variabili di ambiente di licenza UBL esistenti. Per utilizzare le licenze combinate, conservate le sezioni UBL nello script di ambiente di coda per i prodotti per i quali desiderate che si verifichino dei fallback.

Per utilizzare solo BYOL senza fallback UBL per un prodotto specifico, rimuovete la sezione UBL per quel prodotto dallo script e aggiungete la variabile di ambiente di licenza direttamente alla sezione dell'ambiente di coda. `variables` Ad esempio, per utilizzare solo BYOL per Cinema 4D, rimuovete la sezione Cinema 4D dallo script e aggiungetela alla sezione. `g_licenseServerRLM: 127.0.0.1:7057` `variables`

Esempio: utilizzo delle licenze BYOL Cinema 4D con fallback UBL

Prendete in considerazione uno studio che ha già delle licenze di Cinema 4D su un server di licenze nella propria rete locale. Lo studio desidera utilizzare queste licenze per il rendering su Deadline Cloud, ma vuole anche andare oltre il numero di licenze disponibili ricorrendo a UBL quando tutte le licenze BYOL sono in uso.

Per configurare questa configurazione, segui i passaggi indicati Connect flotte gestite dai servizi a un server di licenze personalizzato e apporta le seguenti modifiche al modello di ambiente di coda:

Per configurare le licenze BYOL di Cinema 4D con il fallback UBL

1. Imposta il `LicenseInstanceId` parametro sull'ID dell'istanza Amazon Elastic Compute Cloud (Amazon EC2) del server di licenza o del proxy che ha accesso al server di licenza Cinema 4D.
2. Imposta il `LicensePorts` parametro per includere la porta `7057` (la porta di licenza RLM di Cinema 4D).
3. Nello script Python, mantenete la sezione Cinema 4D che precede il server BYOL alla configurazione UBL:

   ```
   # Cinema4D
   os.environ["g_licenseServerRLM"] = f"127.0.0.1:7057;
   {os.environ.get('g_licenseServerRLM', '')}"
   print(f"openjd_env: g_licenseServerRLM={os.environ['g_licenseServerRLM']}")
   ```

   Questa `g_licenseServerRLM` configurazione è impostata su. `127.0.0.1:7057;UBL_endpoint:7057` Cinema 4D controlla inizialmente il server BYOL. `127.0.0.1:7057` Se nessuna licenza è disponibile, Cinema 4D torna all'endpoint UBL.

4. Rimuovi le sezioni relative ai prodotti che non usi (ad esempio, Arnold, Nuke o SideFX) per mantenere pulita la configurazione.

Se disponi anche di altri prodotti che utilizzano solo BYOL senza fallback UBL, aggiungi quelle variabili di ambiente di licenza direttamente alla `variables` sezione dell'ambiente di coda e rimuovi le sezioni corrispondenti dallo script Python.

Considerazioni sulla concessione di licenze combinate

Tieni a mente le seguenti considerazioni quando utilizzi le licenze combinate:

- Alcune applicazioni non supportano più server di licenze in un'unica variabile di ambiente. Ad esempio, V-Ray utilizza invece un file di configurazione XML. Il modello di ambiente di coda gestisce la configurazione V-Ray separatamente. Per ulteriori informazioni, consulta la sezione V-Ray nel modello di ambiente di coda in. Connect flotte gestite dai servizi a un server di licenze personalizzato
- L'ordine dei server di licenza nella variabile di ambiente determina la priorità. Elenca innanzitutto il server BYOL in modo che le licenze esistenti vengano utilizzate prima delle licenze UBL.
- WindowsSui worker, separate le voci del server di licenza nelle variabili di ambiente con un punto e virgola () anziché i due punti (`;`). `:` Per ulteriori informazioni sulla configurazione delle variabili di ambiente della licenza, vedere. Connect le flotte gestite dai clienti a un endpoint di licenza
- Per utilizzare il fallback UBL con flotte gestite dal cliente, configura un endpoint di licenza. Per ulteriori informazioni, consulta Connect le flotte gestite dai clienti a un endpoint di licenza.

Connect flotte gestite dai servizi a un server di licenze personalizzato

Puoi portare il tuo server di licenza da utilizzare con una flotta gestita dai servizi Deadline Cloud. Per portare la tua licenza, puoi configurare un server di licenze utilizzando un ambiente di coda nella tua farm. Per configurare il server di licenza, è necessario disporre già di una farm e di una coda.

La modalità di connessione a un server di licenze software dipende dalla configurazione del parco macchine e dai requisiti del fornitore del software. In genere, si accede al server in due modi:

- Direttamente al server delle licenze. I tuoi dipendenti ottengono una licenza dal server di licenza del fornitore del software tramite Internet. Tutti i dipendenti devono essere in grado di connettersi al server.
- Tramite un proxy di licenza. I dipendenti si connettono a un server proxy nella rete locale. Solo il server proxy può connettersi al server di licenza del fornitore tramite Internet.

Con le istruzioni riportate di seguito, usi Amazon EC2 Systems Manager (SSM) per inoltrare le porte da un'istanza di lavoro al server di licenza o all'istanza proxy. Nell'esempio seguente, se il server di licenza non è in grado di fornire una licenza, verranno utilizzate le licenze basate sull'utilizzo di Deadline Cloud. Rimuovi le sezioni che non si applicano alla tua pipeline o ai prodotti per i quali non desideri utilizzare licenze basate sull'utilizzo dopo aver esaurito le licenze.

Argomenti

- Fase 1: Configurare l'ambiente di coda
- Fase 2: (Facoltativo) Configurazione dell'istanza del proxy di licenza
- CloudFormation Fase 3: configurazione del modello

Fase 1: Configurare l'ambiente di coda

Puoi configurare un ambiente di coda nella tua coda per accedere al tuo server di licenza. Innanzitutto, assicurati di avere un' AWS istanza configurata con l'accesso al server di licenza utilizzando uno dei seguenti metodi:

- Server di licenza: l'istanza ospita direttamente i server di licenza.
- Proxy di licenza: l'istanza ha accesso di rete al server delle licenze e inoltra le porte del server di licenza al server delle licenze. Per i dettagli su come configurare un'istanza del proxy di licenza, consultaFase 2: (Facoltativo) Configurazione dell'istanza del proxy di licenza.

Per informazioni sulla configurazione delle variabili di ambiente di licenza, vedereFase 3: Connettere un'applicazione di rendering a un endpoint. Per una configurazione personalizzata del server di licenza, l'indirizzo del server di licenza rimane localhost anziché l'endpoint Amazon VPC.

Per aggiungere le autorizzazioni necessarie al ruolo di coda

1. Dalla console Deadline Cloud, scegli Vai alla dashboard.

2. Dalla dashboard, seleziona la farm, quindi la coda che desideri configurare.
3. Da Queue details > service role, seleziona il ruolo.
4. Scegli Aggiungi autorizzazione, quindi scegli Crea politica in linea.
5. Seleziona l'editor di policy JSON, quindi copia e incolla il seguente testo nell'editor.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
                "arn:aws:ec2:us-east-1:111122223333:instance/instance_id"
            ]
        }
    ]
}
```

6. Prima di salvare la nuova policy, sostituisci i seguenti valori nel testo della policy:
   - Sostituisci `region` con la AWS regione in cui si trova la tua azienda
   - Sostituiscilo `instance_id` con l'ID dell'istanza del server di licenza o dell'istanza proxy che stai utilizzando
   - `account_id`Sostituiscilo con il numero di AWS conto contenente la tua fattoria
7. Scegli Next (Successivo).
8. Per il nome della politica, inserisci**`LicenseForwarding`**.
9. Scegli Crea politica per salvare le modifiche e creare la politica con le autorizzazioni richieste.

Per aggiungere un nuovo ambiente di coda alla coda

1. Dalla console Deadline Cloud, scegli Vai alla dashboard se non l'hai già fatto.

2. Dalla dashboard, seleziona la farm, quindi la coda che desideri configurare.
3. Scegli Ambienti di coda > Azioni > Crea nuovo con YAML.
4. Copia e incolla il seguente testo nell'editor di script YAML.

   Windows

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
```

```
      curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
windows/SessionManagerPlugin.zip" -o "{{Session.WorkingDirectory}}/ssm-
plugin.zip"
      powershell -Command "Expand-Archive -Path '{{Session.WorkingDirectory}}/
ssm-plugin.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin'
 -Force; Expand-Archive -Path '{{Session.WorkingDirectory}}/ssm-plugin/
package.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin/package'
 -Force"
      conda activate
      python "{{Env.File.StartSession}}" "{{Session.WorkingDirectory}}/ssm-
plugin/package/bin/session-manager-plugin.exe"
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
```

```
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost;
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Cinema4D
      os.environ["g_licenseServerRLM"] = f"localhost:7057;
{os.environ.get('g_licenseServerRLM', '')}"
      print(f"openjd_env:
 g_licenseServerRLM={os.environ['g_licenseServerRLM']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost;
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
```

```
      os.environ["redshift_LICENSE"] = f"7054@localhost;7055@localhost;
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')

      with open(xml_path, 'w') as f:
          f.write(xml_content)

      os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

      # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
 file is used.
      os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

      # Print out the created xml file's contents
```

```
      print(f"V-Ray configuration file: {xml_path}")
      with open(xml_path, 'r') as f:
          print(f"{f.read()}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
```

```
      curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
      chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
      conda activate
      python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
```

```
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
```

```
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

    if vray_license and vray_license.startswith('licset://'):
        server_parts = vray_license.removeprefix('licset://').split(':')
        if len(server_parts) >= 2:
            xml_content += f"""
        <Host1>{server_parts[0]}</Host1>
        <Port1>{server_parts[1]}</Port1>"""

    xml_content += """
        <User></User>
        <Pass></Pass>
      </LicServer>
    </VRLClient>"""

    temp_dir = tempfile.gettempdir()
    xml_path = os.path.join(temp_dir, 'vrlclient.xml')

    with open(xml_path, 'w') as f:
        f.write(xml_content)

    os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
    print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

    # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
file is used.
    os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
    print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

    # Print out the created xml file's contents
    print(f"V-Ray configuration file: {xml_path}")
    with open(xml_path, 'r') as f:
        print(f"{f.read()}")
```

5. Prima di salvare l'ambiente della coda, apportate le seguenti modifiche al testo dell'ambiente, se necessario:

   - Aggiorna i valori predefiniti per i seguenti parametri in modo che rispecchino il tuo ambiente:

- LicenseInstanceID: l'ID dell'istanza Amazon EC2 del server di licenza o dell'istanza proxy
- LicenseInstanceRegion— La AWS regione in cui si trova la tua fattoria
- LicensePorts— Un elenco di porte separate da virgole da inoltrare al server di licenza o all'istanza proxy (ad esempio 2700,2701)

- Se desideri utilizzare le licenze basate sull'utilizzo (UBL) dopo aver esaurito la licenza Bring your own license (BYOL), assicurati che la porta sia corretta per il tuo server di licenze. Se non desiderate utilizzare UBL dopo aver esaurito il BYOL, aggiungete le variabili di ambiente di licenza necessarie alla sezione variabili.

  Queste variabili dovrebbero DCCs indirizzarlo a localhost sulla porta del server di licenza. Ad esempio, se il server di licenza Foundry è in ascolto sulla porta 6101, dovresti aggiungere la variabile come. **`foundry_LICENSE: 6101@localhost`**

6. (Facoltativo) È possibile lasciare la priorità impostata su 0 oppure modificarla per ordinare la priorità in modo diverso tra più ambienti di coda.
7. Scegli Crea ambiente di coda per salvare il nuovo ambiente.

   Con l'ambiente di coda impostato, i lavori inviati a questa coda recupereranno le licenze dal server di licenza configurato.

Fase 2: (Facoltativo) Configurazione dell'istanza del proxy di licenza

In alternativa all'utilizzo di un server di licenza, puoi utilizzare un proxy di licenza. Per creare un proxy di licenza, crea una nuova istanza Amazon Linux 2023 con accesso di rete al server delle licenze. Se necessario, puoi configurare questo accesso utilizzando una connessione VPN. Per ulteriori informazioni, consulta le connessioni VPN nella Amazon VPC User Guide.

Per configurare un'istanza proxy di licenza per Deadline Cloud, segui i passaggi di questa procedura. Esegui i seguenti passaggi di configurazione su questa nuova istanza per consentire l'inoltro del traffico delle licenze al tuo server di licenza

1. Per installare il HAProxy pacchetto, inserisci

   ```
   sudo yum install haproxy
   ```

2. Aggiorna la sezione listen license-server del file di configurazione etc/haproxy/haproxy/.cfg con quanto segue:

a. Sostituisci LicensePort1 e LicensePort2 con i numeri di porta da inoltrare al server delle licenze. Aggiungi o rimuovi valori separati da virgole per soddisfare il numero di porte richiesto.

b. Sostituire LicenseServerHostcon il nome host o l'indirizzo IP del server di licenza.

```
lobal
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    user        haproxy
    group       haproxy
    daemon

defaults
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen license-server
     bind *:LicensePort1,*:LicensePort2
     server license-server LicenseServerHost
```

3. Per abilitare e avviare il HAProxy servizio, esegui i seguenti comandi:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Dopo aver completato i passaggi, le richieste di licenza inviate a localhost dall'ambiente della coda di inoltro devono essere inoltrate al server di licenza specificato.

CloudFormation Fase 3: configurazione del modello

Puoi utilizzare un CloudFormation modello per configurare un'intera farm in modo che utilizzi le tue licenze.

1. Modifica il modello fornito nel passaggio successivo per aggiungere eventuali variabili di ambiente di licenza richieste alla sezione BYOLQueuevariabili in Ambiente.

2. Utilizza il seguente CloudFormation modello.

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create &ADC; resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm

  QueuePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLQueuePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
              - s3:GetBucketLocation
            Resource:
```

```
              - !Sub ${JobAttachmentBucket.Arn}
              - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
            Condition:
              StringEquals:
                aws:ResourceAccount: !Sub ${AWS::AccountId}
          - Effect: Allow
            Action: logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          - Effect: Allow
            Action:
              - s3:ListBucket
              - s3:GetObject
            Resource:
              - "*"
            Condition:
              ArnLike:
                s3:DataAccessPointArn:
                  - arn:aws:s3:*:*:accesspoint/deadline-software-*
              StringEquals:
                s3:AccessPointNetworkOrigin: VPC

  BYOLSSMPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLSSMPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - ssm:StartSession
            Resource:
              - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
              - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}


  WorkerPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLWorkerPolicy
      PolicyDocument:
```

```
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - logs:CreateLogStream
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
            Condition:
              ForAnyValue:StringEquals:
                aws:CalledVia:
                  - deadline.amazonaws.com
          - Effect: Allow
            Action:
              - logs:PutLogEvents
              - logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*


  QueueRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLQueueRole
      ManagedPolicyArns:
        - !Ref QueuePolicy
        - !Ref BYOLSSMPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service:
                - credentials.deadline.amazonaws.com
                - deadline.amazonaws.com
            Condition:
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
              ArnEquals:
                aws:SourceArn: !Ref Farm

  WorkerRole:
    Type: AWS::IAM::Role
```

```
    Properties:
      RoleName: BYOLWorkerRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
        - !Ref WorkerPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service: credentials.deadline.amazonaws.com


  Queue:
    Type: AWS::Deadline::Queue
    Properties:
      DisplayName: BYOLQueue
      FarmId: !GetAtt Farm.FarmId
      RoleArn: !GetAtt QueueRole.Arn
      JobRunAsUser:
        Posix:
          Group: ""
          User: ""
        RunAs: WORKER_AGENT_USER
      JobAttachmentSettings:
        RootPrefix: job-attachments
        S3BucketName: !Ref JobAttachmentBucket

  Fleet:
    Type: AWS::Deadline::Fleet
    Properties:
      DisplayName: BYOLFleet
      FarmId: !GetAtt Farm.FarmId
      MinWorkerCount: 1
      MaxWorkerCount: 2
      Configuration:
        ServiceManagedEc2:
          InstanceCapabilities:
            VCpuCount:
              Min: 4
              Max: 16
            MemoryMiB:
```

```
              Min: 4096
              Max: 16384
            OsFamily: LINUX
            CpuArchitectureType: x86_64
          InstanceMarketOptions:
            Type: on-demand
      RoleArn: !GetAtt WorkerRole.Arn

  QFA:
    Type: AWS::Deadline::QueueFleetAssociation
    Properties:
      FarmId: !GetAtt Farm.FarmId
      FleetId: !GetAtt Fleet.FleetId
      QueueId: !GetAtt Queue.QueueId

  CondaQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 5
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: |
        specificationVersion: 'environment-2023-09'
        parameterDefinitions:
        - name: CondaPackages
          type: STRING
          description: >
            This is a space-separated list of conda package match specifications to
 install for the job.
            E.g. "blender=3.6" for a job that renders frames in Blender 3.6.

            See https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/
pkg-specs.html#package-match-specifications
          default: ""
          userInterface:
            control: LINE_EDIT
            label: Conda Packages
        - name: CondaChannels
          type: STRING
          description: >
            This is a space-separated list of conda channels from which to install
 packages. &ADC; SMF packages are
```

```
            installed from the "deadline-cloud" channel that is configured by
 &ADC;.

            Add "conda-forge" to get packages from the https://conda-forge.org/
 community, and "defaults" to get packages
            from Anaconda Inc (make sure your usage complies with https://
www.anaconda.com/terms-of-use).
          default: "deadline-cloud"
          userInterface:
            control: LINE_EDIT
            label: Conda Channels
        environment:
          name: Conda
          script:
            actions:
              onEnter:
                command: "conda-queue-env-enter"
                args: ["{{Session.WorkingDirectory}}/.env", "--packages",
 "{{Param.CondaPackages}}", "--channels", "{{Param.CondaChannels}}"]
              onExit:
                command: "conda-queue-env-exit"

  BYOLQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 10
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: !Sub |
        specificationVersion: "environment-2023-09"
        parameterDefinitions:
        - name: LicenseInstanceId
          type: STRING
          description: >
            The Instance ID of the license server/proxy instance
          default: ""
        - name: LicenseInstanceRegion
          type: STRING
          description: >
            The region containing this farm
          default: ""
        - name: LicensePorts
          type: STRING
```

```
          description: >
            Comma-separated list of ports to be forwarded to the license server/
proxy
            instance. Example:
 "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
          default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
        environment:
        name: BYOL License Forwarding
        variables:
          example_LICENSE: 2701@localhost
        script:
          actions:
          onEnter:
            command: bash
            args: [ "{{Env.File.Enter}}" ]
          onExit:
            command: bash
            args: [ "{{Env.File.Exit}}" ]
          embeddedFiles:
          - name: Enter
            type: TEXT
            runnable: True
            data: |
              curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
              chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
              conda activate
              python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
session-manager-plugin
          - name: Exit
            type: TEXT
            runnable: True
            data: |
              echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
              for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
          - name: StartSession
            type: TEXT
            data: |
              import boto3
              import json
              import subprocess
              import sys
```

```
              import os
              import tempfile

              instance_id = "{{Param.LicenseInstanceId}}"
              region = "{{Param.LicenseInstanceRegion}}"
              license_ports_list = "{{Param.LicensePorts}}".split(",")

              ssm_client = boto3.client("ssm", region_name=region)
              pids = []

              for port in license_ports_list:
                session_response = ssm_client.start_session(
                  Target=instance_id,
                  DocumentName="AWS-StartPortForwardingSession",
                  Parameters={"portNumber": [port], "localPortNumber": [port]}
                )

                cmd = [
                  sys.argv[1],
                  json.dumps(session_response),
                  region,
                  "StartSession",
                  "",
                  json.dumps({"Target": instance_id}),
                  f"https://ssm.{region}.amazonaws.com"
                ]

                process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
                pids.append(process.pid)
                print(f"SSM Port Forwarding Session started for port {port}")

              print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in
pids)}")

              # Enabling UBL after the "bring your own license" (BYOL) has run out
requires prepending the BYOL configuration to the existing license setup
              # Remove the sections that do not apply to your pipeline, or you do
not want to use UBL after exhausting the BYOL licenses.
              # The port numbers used may not match what your license server is
serving.

              # Arnold
```

```
            os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
            print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

            # Nuke
            os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
            print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

            # SideFX
            os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
            print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

            # Redshift and Red Giant
            os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
            print(f"openjd_env:
 redshift_LICENSE={os.environ['redshift_LICENSE']}")

            # V-Ray doesn't support multiple license servers in a single
 environment variable
            # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
            vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
            xml_content = """<VRLClient>
              <LicServer>
                <Host>localhost</Host>
                <Port>30304</Port>"""

            if vray_license and vray_license.startswith('licset://'):
                server_parts = vray_license.removeprefix('licset://').split(':')
                if len(server_parts) >= 2:
                    xml_content += f"""
                <Host1>{server_parts[0]}</Host1>
                <Port1>{server_parts[1]}</Port1>"""

            xml_content += """
                <User></User>
                <Pass></Pass>
              </LicServer>
            </VRLClient>"""
```

```
            temp_dir = tempfile.gettempdir()
            xml_path = os.path.join(temp_dir, 'vrlclient.xml')

            with open(xml_path, 'w') as f:
                f.write(xml_content)

            os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
            print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

            # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the
vrlclient.xml file is used.
            os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
            print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

            # Print out the created xml file's contents
            print(f"V-Ray configuration file: {xml_path}")
            with open(xml_path, 'r') as f:
                print(f"{f.read()}")
```

3. Quando distribuisci il CloudFormation modello, fornisci i seguenti parametri:

   - Aggiorna l'LicenseInstanceID con l'ID dell'istanza Amazon EC2 del server di licenza o dell'istanza proxy
   - Aggiorna il file LicensePortscon un elenco di porte separate da virgole da inoltrare al server di licenza o all'istanza proxy (ad esempio 2700,2701)
   - Aggiungi le variabili di ambiente di licenza sostituendole nel modello **`example_LICENSE: 2700@localhost`**

4. Implementa il modello per configurare la tua farm con funzionalità Bring Your Own License.

Connect le flotte gestite dai clienti a un endpoint di licenza

Il server di licenze basato sull'utilizzo AWS di Deadline Cloud fornisce licenze su richiesta per determinati prodotti di terze parti. Con le licenze basate sull'utilizzo, puoi pagare in base al consumo. Ti viene addebitato solo il tempo che utilizzi. Le licenze basate sull'utilizzo forniscono licenze per il rendering ai dipendenti di Deadline Cloud, non le licenze per le applicazioni DCC.

Il server di licenza basato sull'utilizzo di Deadline Cloud può essere utilizzato con qualsiasi tipo di flotta, purché i lavoratori di Deadline Cloud possano comunicare con il server delle licenze. Il

server delle licenze viene configurato automaticamente in flotte gestite dai servizi. La seguente configurazione è necessaria solo per le flotte gestite dal cliente.

Per creare il server di licenza, è necessario un gruppo di sicurezza per il VPC della farm che consenta il traffico per licenze di terze parti.

Argomenti

- Fase 1: Creare un gruppo di sicurezza
- Passaggio 2: configura l'endpoint della licenza
- Fase 3: Connettere un'applicazione di rendering a un endpoint
- Fase 4: Eliminare un endpoint di licenza

Fase 1: Creare un gruppo di sicurezza

Usa la console Amazon VPC per creare un gruppo di sicurezza per il VPC della tua fattoria. Configura il gruppo di sicurezza per consentire le seguenti regole in entrata:

- Autodesk Maya e Arnold — 2701 - 2702, TCP, IPv4 IPv6
- Cinema 4D — 7057, TCP, IPv4 IPv6
- Foundry Nuke — 6101, TCP,, IPv4 IPv6
- Gigante rosso — 7055, TCP, IPV4
- Redshift — 7054, TCP, IPv4 IPv6
- SideFX Houdini, Mantra e Karma — 1715 - 1717, TCP, IPv4 IPv6
- VRay — 30304, TCP, IPV4

L'origine di ogni regola in entrata è il gruppo di sicurezza dei lavoratori del parco macchine.

Per ulteriori informazioni sulla creazione di un gruppo di sicurezza, consulta Creare un gruppo di sicurezza nella guida per l'utente di Amazon Virtual Private Cloud.

Passaggio 2: configura l'endpoint della licenza

Un endpoint di licenza fornisce l'accesso ai server di licenza per prodotti di terze parti. Le richieste di licenza vengono inviate all'endpoint di licenza. L'endpoint le indirizza al server di licenza appropriato. Il server delle licenze tiene traccia dei limiti di utilizzo e dei diritti. La creazione di un endpoint di

licenza in Deadline Cloud fornisce un endpoint di AWS PrivateLink interfaccia nel tuo VPC. Questi endpoint vengono fatturati in base ai prezzi standard. AWS PrivateLink Per ulteriori informazioni, consultare Prezzi di AWS PrivateLink.

Con le autorizzazioni appropriate, puoi creare il tuo endpoint di licenza. Per la politica richiesta per creare un endpoint di licenza, vedi Politica per consentire la creazione di un endpoint di licenza.

Puoi creare il tuo endpoint di licenza dalla dashboard nella console Deadline Cloud.

1. Dal riquadro di navigazione a sinistra, scegli Endpoint di licenza, quindi scegli Crea endpoint di licenza.
2. Dalla pagina Crea endpoint di licenza, completa quanto segue:
   - Selezionare un VPC.
   - Seleziona le sottoreti che contengono i tuoi lavoratori di Deadline Cloud. Puoi selezionare fino a 10 sottoreti.
   - Seleziona il gruppo di sicurezza creato nel passaggio 1. Puoi selezionare fino a 10 gruppi di sicurezza per scenari più complicati.
   - (Facoltativo) Scegli Aggiungi nuovo tag e aggiungi uno o più tag. Puoi aggiungere fino a 50 tag.
3. Scegli Crea endpoint di licenza. Quando l'endpoint di licenza viene creato, viene visualizzato nella pagina degli endpoint di licenza.
4. Nella sezione prodotti misurati, scegli Aggiungi prodotti, quindi seleziona i prodotti che desideri aggiungere all'endpoint di licenza. Scegliere Aggiungi.

Per rimuovere un prodotto da un endpoint di licenza, nella sezione Prodotti misurati, seleziona il prodotto e quindi scegli Rimuovi. Nella conferma, scegli nuovamente Rimuovi.

Fase 3: Connettere un'applicazione di rendering a un endpoint

Dopo aver configurato l'endpoint di licenza, le applicazioni lo utilizzano nello stesso modo in cui utilizzano un server di licenze di terze parti. In genere si configura il server di licenza per l'applicazione impostando una variabile di ambiente o un'altra impostazione di sistema, ad esempio una chiave di registro di Microsoft Windows, su una porta e un indirizzo del server di licenza.

Per ottenere il nome DNS dell'endpoint della licenza, seleziona l'endpoint della licenza nella console, quindi scegli l'icona di copia nella sezione Nome DNS.

Esempi di configurazione

Example— Autodesk Maya e Arnold

> **Note**
>
> È possibile utilizzare Autodesk Maya e Arnold insieme o separatamente. Usa la porta 2702 per Autodesk Maya e la porta 2701 per Arnold.

Per Autodesk Maya, imposta la variabile di ambiente su: `ADSKFLEX_LICENSE_FILE`

```
2702@VPC_Endpoint_DNS_Name
```

Per Arnold, imposta la variabile di ambiente su: `ADSKFLEX_LICENSE_FILE`

```
2701@VPC_Endpoint_DNS_Name
```

Per Autodesk Maya e Arnold, imposta la variabile di ambiente su: `ADSKFLEX_LICENSE_FILE`

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Per Windows i lavoratori, utilizzate un punto e virgola (;) anziché i due punti (:) per separare gli endpoint.

Example— Cinema 4D

Imposta la variabile d'ambiente `g_licenseServerRLM` su:

```
VPC_Endpoint_DNS_Name:7057
```

Dopo aver creato la variabile di ambiente, dovresti essere in grado di renderizzare un'immagine usando una riga di comando simile a questa:

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^
```

```
    "C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^
    -oimage "C:\Users\Administrator\User\MyOutputImage.png
```

Example— Foundry Nuke

Imposta la variabile `foundry_LICENSE` di ambiente su:

```
6101@VPC_Endpoint_DNS_Name
```

Per verificare che le licenze funzionino correttamente, puoi eseguire Nuke in un terminale:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Gigante rosso

Imposta la variabile di ambiente `redshift_LICENSE` su:

```
7055@VPC_Endpoint_DNS_Name
```

Nota che Red Giant e Redshift hanno la stessa variabile di `redshift_LICENSE` ambiente. Se desideri utilizzare entrambe le applicazioni, puoi impostare la variabile di ambiente su:

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Per Windows i lavoratori, utilizzate un punto e virgola (;) anziché i due punti (:) per separare gli endpoint.

Per verificare che le licenze funzionino correttamente, assicuratevi di aver installato After Effects e Red Giant. Quindi, potete renderizzare un progetto usando un comando simile a questo:

```
C:\Program Files\Adobe\Adobe After Effects 2025\Support Files\aerender.exe -comp "Comp
 1" -project
         C:\Users\MyUser\myAfterEffectsProjectUsingRedGiant.aep -output
           C:\Users\MyUser\myMovieWithRedGiant.mp4
```

Example— Redshift

Imposta la variabile di ambiente `redshift_LICENSE` su:

```
7054@VPC_Endpoint_DNS_Name
```

Dopo aver creato la variabile di ambiente, dovresti essere in grado di renderizzare un'immagine usando una riga di comando simile a questa:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SideFX Houdini, Mantra e Karma

Esegui il comando seguente:

```
/opt/hfs19.5.640/bin/hserver -S
 "http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://
VPC_Endpoint_DNS_Name:1717;"
```

Per verificare che le licenze funzionino correttamente, puoi renderizzare una scena di Houdini tramite questo comando:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Imposta la variabile di ambiente su: `VRAY_AUTH_CLIENT_SETTINGS`

```
licset://VPC_Endpoint_DNS_Name:30304
```

Imposta la variabile di ambiente `VRAY_AUTH_CLIENT_FILE_PATH` su:

```
/null
```

Per verificare che le licenze funzionino correttamente, puoi renderizzare un'immagine VRay usando un comando simile a questo:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Fase 4: Eliminare un endpoint di licenza

Quando elimini il tuo parco veicoli gestito dal cliente, ricordati di eliminare l'endpoint della licenza. Se non elimini l'endpoint della licenza, continueranno a esserti addebitati i costi fissi AWS PrivateLink

Puoi eliminare l'endpoint della licenza dalla dashboard nella console Deadline Cloud.

1. Dal riquadro di navigazione a sinistra, scegli License endpoints.
2. Seleziona l'endpoint che desideri eliminare e scegli elimina, quindi scegli nuovamente elimina per confermare.

Utilizzo di agenti AI con Deadline Cloud

Usa gli agenti AI per scrivere pacchetti di lavoro, sviluppare pacchetti conda e risolvere i problemi dei lavori in Deadline Cloud. Questo argomento spiega cosa sono gli agenti di intelligenza artificiale, i punti chiave per lavorare con loro in modo efficace e le risorse per aiutare gli agenti a comprendere Deadline Cloud.

Un agente AI è uno strumento software che utilizza un modello di linguaggio di grandi dimensioni (LLM) per eseguire attività in modo autonomo. Gli agenti di intelligenza artificiale possono leggere e scrivere file, eseguire comandi e iterare sulle soluzioni in base al feedback. Gli esempi includono strumenti da riga di comando come Kiro e gli assistenti integrati con IDE.

Punti chiave per lavorare con agenti di intelligenza artificiale

I seguenti punti chiave ti aiutano a ottenere risultati migliori quando utilizzi agenti AI con Deadline Cloud.

- Fornisci le basi necessarie: gli agenti di intelligenza artificiale ottengono le migliori prestazioni quando hanno accesso alla documentazione, alle specifiche e agli esempi pertinenti. È possibile fornire informazioni di base indirizzando l'agente verso pagine di documentazione specifiche, condividendo il codice di esempio esistente come riferimento, clonando i repository open source pertinenti nell'area di lavoro locale e fornendo documentazione per applicazioni di terze parti.
- Specificare i criteri di successo: definire il risultato previsto e i requisiti tecnici per l'agente. Ad esempio, quando chiedi a un agente di sviluppare un job bundle, specifica gli input, i parametri e gli output previsti per il lavoro. Se non sei sicuro delle specifiche, chiedi prima all'agente di proporre delle opzioni, quindi perfeziona insieme i requisiti.
- Abilita un ciclo di feedback: gli agenti di intelligenza artificiale eseguono iterazioni in modo più efficace quando possono testare le loro soluzioni e ricevere feedback. Invece di aspettarti una soluzione funzionante al primo tentativo, offri all'agente la possibilità di eseguire la sua soluzione e di esaminarne i risultati. Questo approccio funziona bene quando l'agente può accedere agli aggiornamenti di stato, ai log e agli errori di convalida. Ad esempio, quando sviluppi un pacchetto di offerte di lavoro, consenti all'agente di inviare il lavoro e rivedere i log.
- Aspettatevi di iterare: anche in presenza di un buon contesto, gli agenti possono sbagliare rotta o formulare ipotesi che non corrispondono al vostro ambiente. Osserva come l'agente affronta l'attività e fornisci indicazioni lungo il percorso. Aggiungi il contesto mancante se l'agente ha difficoltà, aiuta a trovare gli errori indicando file di registro specifici, perfeziona i requisiti man mano che li scopri e aggiungi requisiti negativi per indicare esplicitamente cosa l'agente deve evitare.

Risorse per il contesto dell'agente

Le seguenti risorse aiutano gli agenti di intelligenza artificiale a comprendere i concetti di Deadline Cloud e a produrre risultati accurati.

- Server Deadline Cloud Model Context Protocol (MCP): per gli agenti che supportano il Model Context Protocol, l'archivio deadline-cloud contiene il client Deadline Cloud che include un server MCP per interagire con i lavori.
- AWSServer MCP per la documentazione: per gli agenti che supportano MCP, configura il server Documentation MCP per consentire all'agente l'accesso diretto alla AWSdocumentazione, inclusa la Deadline Cloud User Guide e la Developer GuideAWS.
- Specificazione Open Job Description — La specifica Open Job Description su GitHub definisce lo schema per i modelli di lavoro. Fate riferimento a questo repository quando gli agenti devono comprendere la struttura e la sintassi dei modelli di lavoro.
- deadline-cloud-samples— Il deadline-cloud-samplesrepository contiene esempi di job bundle, ricette conda e CloudFormation modelli per applicazioni e casi d'uso comuni.
- aws-deadline organization: GitHub l'organizzazione aws-deadline GitHub contiene plugin di riferimento per molte applicazioni di terze parti che puoi usare come esempi per altre integrazioni.

Richiesta di esempio: scrivere un pacchetto di lavoro

Il seguente prompt di esempio mostra come utilizzare un agente AI per creare pacchetti di lavoro che addestrano un adattatore LoRa (Low-Rank Adaptation) per la generazione di immagini AI. Il prompt illustra i punti chiave discussi in precedenza: fornisce le basi indicando gli archivi pertinenti, definisce i criteri di successo per i risultati del job bundle e delinea un ciclo di feedback per lo sviluppo iterativo.

```
Write a pair of job bundles for Deadline Cloud that use the diffusers Python library to
 train a LoRA adapter on a set of images and then generate images from it.

Requirements:
- The training job takes a set of JPEG images as input, uses an image description,
 LoRA rank, learning rate, batch size, and number of training steps as parameters, and
 outputs a `.safetensors` file.
- The generation job takes the `.safetensors` file as input and the number of images
 to generate, then outputs JPEG images. The jobs use Stable Diffusion 1.5 as the base
 model.
- The jobs run `diffusers` as a Python script. Install the necessary packages using
 conda by setting the job parameters:
    - `CondaChannels`: `conda-forge`
```

```
    - `CondaPackages`: list of conda packages to install

For context, clone the following repositories to your workspace and review their
 documentation and code:
- OpenJobDescription specification: https://github.com/OpenJobDescription/openjd-
specifications/blob/mainline/wiki/2023-09-Template-Schemas.md
- Deadline Cloud sample job bundles: https://github.com/aws-deadline/deadline-cloud-
samples/tree/mainline/job_bundles
- diffusers library: https://github.com/huggingface/diffusers

Read through the provided context before you start. To develop a job bundle, iterate
 with the following steps until the submitted job succeeds. If a step fails, update the
 job bundle and restart the loop:
1. Create a job bundle.
2. Validate the job template syntax: `openjd check`
3. Submit the job to Deadline Cloud: `deadline bundle submit`
4. Wait for the job to complete: `deadline job wait`
5. View the job status and logs: `deadline job logs`
6. Download the job output: `deadline job download-output`

To verify the training and generation jobs work together, iterate with the following
 steps until the generation job produces images that resemble the dog in the training
 data:
1. Develop and submit a training job using the training images in `./exdog`
2. Wait for the job to succeed then download its output.
3. Develop and submit a generation job using the LoRA adapter from the training job.
4. Wait for the job to succeed then download its output.
5. Inspect the generated images. If they resemble the dog in the training data, you're
 done. Otherwise, review the job template, job parameters, and job logs to identify and
 fix the issue.
```

Monitoraggio di AWS Deadline Cloud

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di AWS Deadline Cloud (Deadline Cloud) e delle tue soluzioni. AWS Raccogli i dati di monitoraggio da tutte le parti della tua AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. Prima di iniziare a monitorare Deadline Cloud, devi creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

AWSe Deadline Cloud forniscono strumenti che puoi utilizzare per monitorare le tue risorse e rispondere a potenziali incidenti. Alcuni di questi strumenti eseguono il monitoraggio per te, altri richiedono un intervento manuale. È necessario automatizzare il più possibile le attività di monitoraggio.

- Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. È possibile raccogliere e tenere traccia dei parametri, creare pannelli di controllo personalizzati e impostare allarmi per inviare una notifica o intraprendere azioni quando un parametro specificato raggiunge una determinata soglia. Ad esempio, puoi tenere CloudWatch traccia dell'utilizzo della CPU o di altri parametri delle tue EC2 istanze Amazon e avviare automaticamente nuove istanze quando necessario. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.

  Deadline Cloud ha tre CloudWatch metriche.

- Amazon CloudWatch Logs ti consente di monitorare, archiviare e accedere ai tuoi file di registro da EC2 istanze Amazon e altre fonti. CloudTrail CloudWatch I log possono monitorare le informazioni nei file di registro e avvisarti quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulta la Amazon CloudWatch Logs User Guide.

- Amazon EventBridge può essere utilizzato per automatizzare i AWS servizi e rispondere automaticamente agli eventi di sistema, come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi AWS relativi ai servizi vengono forniti quasi EventBridge in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali operazioni automatizzate intraprendere quando un evento corrisponde a una regola. Per ulteriori informazioni, consulta Amazon EventBridge User Guide.
- AWS CloudTrailacquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo AWS account e invia i file di log a un bucket Amazon S3 da te specificato. Puoi identificare quali utenti e account hanno chiamatoAWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.

Argomenti

- Registrazione delle chiamate Deadline Cloud API utilizzando AWS CloudTrail
- Monitoraggio con CloudWatch
- Gestione degli eventi Deadline Cloud utilizzando Amazon EventBridge

Registrazione delle chiamate Deadline Cloud API utilizzando AWS CloudTrail

Deadline Cloud è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o un Servizio AWS. CloudTrail acquisisce tutte le chiamate API Deadline Cloud come eventi. Le chiamate acquisite includono chiamate dalla Deadline Cloud console e chiamate di codice alle operazioni Deadline Cloud API. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare a quale richiesta è stata effettuata Deadline Cloud, l'indirizzo IP da cui è stata effettuata la richiesta, quando è stata effettuata e ulteriori dettagli.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente del Centro identità IAM.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWSPer ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per una registrazione continua degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi CloudTrailLake.

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il Console di gestione AWS sono multiregionali. È possibile creare un trail per una singola Regione o per più Regioni tramite AWS CLI. La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio Regioni AWS account. Se si crea un trail per una singola Regione, è possibile visualizzare solo gli eventi registrati nella Regione AWS del trail. Per ulteriori informazioni sui trail, consulta Creating a trail for your Account AWS e Creating a trail for an organization nella Guida per l'utente di AWS CloudTrail .

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail Per informazioni sui prezzi di Amazon S3, consulta Prezzi di Amazon S3.

CloudTrail Archivi di dati sugli eventi di Lake

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC. ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i selettori di eventi avanzati. I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta Working with AWS CloudTrail Lake nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'opzione di prezzo da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di

conservazione predefinito e quello massimo per il datastore di eventi. Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione Prezzi.AWS CloudTrail

Deadline Cloud eventi relativi ai dati in CloudTrail

Gli eventi di dati forniscono informazioni sulle operazioni delle risorse eseguite su o in una risorsa (ad esempio, lettura o scrittura su un oggetto Amazon S3). Queste operazioni sono definite anche operazioni del piano dei dati. Gli eventi di dati sono spesso attività che interessano volumi elevati di dati. Per impostazione predefinita, CloudTrail non registra gli eventi relativi ai dati. La cronologia CloudTrail degli eventi non registra gli eventi relativi ai dati.

Per gli eventi di dati sono previsti costi aggiuntivi. Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione AWS CloudTrail Prezzi.

Puoi registrare gli eventi relativi ai dati per i tipi di Deadline Cloud risorse utilizzando la CloudTrail console o AWS CLI le operazioni CloudTrail dell'API. Per ulteriori informazioni su come registrare gli eventi relativi ai dati, consulta Registrazione di eventi relativi ai dati con Console di gestione AWS e Registrazione di eventi di dati con AWS Command Line Interface nella Guida per l'utente AWS CloudTrail .

La tabella seguente elenca i tipi di Deadline Cloud risorse per i quali è possibile registrare gli eventi relativi ai dati. La colonna Data event type (console) mostra il valore da scegliere dall'elenco Data event type (console) sulla CloudTrail console. La colonna del valore resources.type mostra il `resources.type` valore da specificare durante la configurazione dei selettori di eventi avanzati utilizzando o. AWS CLI CloudTrail APIs La CloudTrail colonna Dati APIs registrati mostra le chiamate API registrate per il tipo di risorsa. CloudTrail

Tipo di evento di dati (console)	valore resources.type	Dati registrati APIs su CloudTrail
Deadline Fleet	`AWS::Deadline::Fleet`	• SearchWorkers
Coda di scadenze	`AWS::Deadline::Fleet`	• SearchJobs
Addetto alle scadenze	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity

Tipo di evento di dati (console)	valore resources.type	Dati registrati APIs su CloudTrail
		• ListWorkers
Deadline Job	`AWS::Deadline::Job`	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

È possibile configurare selettori di eventi avanzati per filtrare i campi `eventName`, `readOnly` e `resources.ARN` per registrare solo gli eventi importanti per l'utente. Per ulteriori informazioni su questi campi, consulta AdvancedFieldSelector in Riferimento API AWS CloudTrail .

Deadline Cloud eventi gestionali in CloudTrail

Gli eventi di gestione forniscono informazioni sulle operazioni di gestione eseguite sulle risorse dell'azienda Account AWS. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

AWS Deadline Cloud registra le seguenti operazioni del piano Deadline Cloud di controllo CloudTrail come eventi di gestione.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-leggi
- assume-fleet-role-for-lavoratore
- assume-queue-role-for-leggi
- assume-queue-role-for-utente
- assume-queue-role-for-lavoratore
- creare un budget
- crea-fattoria
- create-fleet
- create-license-endpoint
- crea un limite
- crea-monitor
- crea coda
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- create-worker
- elimina-budget
- elimina-farm
- delete-fleet

- delete-license-endpoint
- elime-limite
- delete-metered-product
- elimina-monitora
- cancella coda
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- delete-worker
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- ottenere un budget
- prendere in fattoria
- get-feature-map
- prendi una flotta
- get-license-endpoint
- ottieni un limite
- richiedi il monitoraggio
- get-queue
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-coda
- list-available-metered-products

- elenca-budget
- list-farm-members
- elenca-fattorie
- list-fleet-members
- list-flotte
- list-job-members
- list-license-endpoints
- limite di lista
- list-metered-products
- list-monitor
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- list-queues
- list-storage-profiles
- list-storage-profiles-for-coda
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- aggiorna il budget
- update-farm
- aggiorna la flotta
- limite di aggiornamento
- aggiornamento-monitor
- coda di aggiornamento
- update-queue-environment
- update-queue-fleet-association

- update-queue-limit-association
- update-storage-profile
- update-worker

Deadline Cloud esempi di eventi

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'operazione API richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi gli eventi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail evento che dimostra l'`CreateFarm`operazione.

```
{
    "eventVersion": "0",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
        "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-
Session",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE-PrincipalID",
                "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
                "accountId": "111122223333",
                "userName": "EXAMPLE-UserName"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-03-08T23:25:49Z"
            }
        }
    },
    "eventTime": "2021-03-08T23:25:49Z",
    "eventSource": "deadline.amazonaws.com",
    "eventName": "CreateFarm",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "EXAMPLE-userAgent",
    "requestParameters": {
        "displayName": "example-farm",
        "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
        "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
        "description": "example-description",
        "tags": {
            "purpose_1": "e2e"
            "purpose_2": "tag_test"
        }
    },
    "responseElements": {
        "farmId": "EXAMPLE-farmID"
    },
    "requestID": "EXAMPLE-requestID",
    "eventID": "EXAMPLE-eventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
}
```

L'esempio JSON mostra Regione AWS l'indirizzo IP e altri "`requestParameters`" come "e" `displayName` `kmsKeyArn` "che possono aiutare a identificare l'evento.

Per informazioni sul contenuto dei CloudTrail record, consulta i contenuti dei CloudTrail record nella Guida per l'AWS CloudTrail utente.

Monitoraggio con CloudWatch

Amazon CloudWatch (CloudWatch) raccoglie dati grezzi e li elabora in parametri leggibili quasi in tempo reale. Puoi aprire la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/per visualizzare e filtrare le metriche di Deadline Cloud.

Queste statistiche vengono conservate per 15 mesi in modo da poter accedere alle informazioni storiche per avere una prospettiva migliore sulle prestazioni della tua applicazione o del tuo servizio web. È anche possibile impostare allarmi che controllano determinate soglie e inviare notifiche o intraprendere azioni quando queste soglie vengono raggiunte. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.

Deadline Cloud ha due tipi di registri: i registri delle attività e i registri dei lavoratori. Un registro delle attività si verifica quando si eseguono i registri di esecuzione come script o come esecuzione di DCC. Un registro delle attività potrebbe mostrare eventi come il caricamento delle risorse, il rendering dei riquadri o il mancato rilevamento delle texture.

Un registro dei lavoratori mostra i processi degli agenti di lavoro. Questi potrebbero includere elementi come l'avvio degli agenti di lavoro, la registrazione, i report sullo stato di avanzamento, il caricamento delle configurazioni o il completamento delle attività.

Lo spazio dei nomi per questi registri è. `/aws/deadline/*`

Per Deadline Cloud, i lavoratori caricano questi registri in Logs. CloudWatch Per impostazione predefinita, i log non scadono mai. Se un lavoro produce un volume elevato di dati, è possibile incorrere in costi aggiuntivi. Per ulteriori informazioni, consulta i CloudWatch prezzi di Amazon.

Puoi modificare la politica di conservazione per ogni gruppo di log. Una conservazione più breve rimuove i vecchi log e può aiutare a ridurre i costi di archiviazione. Per conservare i log, puoi archiviarli su Amazon Simple Storage Service prima di rimuoverli. Per ulteriori informazioni, consulta Esportazione dei dati di registro in Amazon S3 utilizzando la console nella guida per l' CloudWatch utente di Amazon.

> **Note**
>
> CloudWatch le letture dei log sono limitate da. AWS Se hai intenzione di inserire molti artisti, ti suggeriamo di contattare l'assistenza AWS clienti e richiedere un aumento della `GetLogEvents` quota di iscrizione. CloudWatch Inoltre, ti consigliamo di chiudere il portale di log tailing quando non stai eseguendo il debug.

Per ulteriori informazioni, consulta CloudWatch Logs quotas nella guida per CloudWatch l'utente di Amazon.

CloudWatch metriche

Deadline Cloud invia i parametri ad Amazon. CloudWatch Puoi utilizzare il Console di gestione AWSAWS CLI, il o un'API per elencare le metriche a cui Deadline Cloud invia. CloudWatch Per impostazione predefinita, ogni punto dati copre il minuto successivo all'ora di inizio dell'attività. Per informazioni su come visualizzare i parametri disponibili utilizzando Console di gestione AWS o ilAWS CLI, consulta Visualizza i parametri disponibili nella Amazon CloudWatch User Guide.

Metriche della flotta gestite dal cliente

Il `AWS/DeadlineCloud` namespace contiene le seguenti metriche per le flotte gestite dai clienti:

Metrica	Description	Unità
`RecommendedFleetSize`	Il numero di lavoratori che Deadline Cloud consiglia di utilizzare per elaborare i lavori. Puoi utilizzare questa metrica per espandere o contrarre il numero di lavoratori della tua flotta.	Conteggio
`UnhealthyWorkerCount`	Il numero di lavoratori assegnati a processare lavori non salutari.	Conteggio

Puoi utilizzare le seguenti dimensioni per perfezionare le metriche della flotta gestita dal cliente:

Dimensione	Description
FarmId	Questa dimensione filtra i dati richiesti all'azien da agricola specificata.
FleetId	Questa dimensione filtra i dati richiesti alla flotta di lavoratori specificata.

Metriche relative alle licenze

Il `AWS/DeadlineCloud` namespace contiene le seguenti metriche per le licenze:

Metrica	Description	Unità
`LicensesInUse`	Il numero di sessioni di licenza in uso.	Conteggio

È possibile utilizzare le seguenti dimensioni per perfezionare le metriche di licenza:

Dimensione	Description
FleetId	Utilizza questa dimensione per filtrare i dati in base al parco veicoli gestito dal servizio specificato. Per le flotte gestite dal cliente, utilizza invece la dimensione. LicenseEn dpointId
LicenseEndpointId	Utilizza questa dimensione per filtrare i dati fino all'endpoint di licenza specificato.
Prodotto	Utilizza questa dimensione per filtrare i dati relativi al prodotto misurato specificato.

Metriche relative al limite delle risorse

Il `AWS/DeadlineCloud` namespace contiene le seguenti metriche per i limiti delle risorse:

Metrica	Description	Unità
`CurrentCount`	Il numero di risorse modellate da questo limite in uso.	Conteggio
`MaxCount`	Il numero massimo di risorse modellate da questo limite. Se imposti il `maxCount` valore su -1 utilizzando l'API, Deadline Cloud non emette la metrica. `MaxCount`	Conteggio

Puoi utilizzare le seguenti dimensioni per rifinire le metriche dei limiti concorrenti:

Dimensione	Description
FarmId	Questa dimensione filtra i dati richiesti alla farm specificata.
LimitId	Questa dimensione filtra i dati richiesti fino al limite specificato.

Allarmi raccomandati

Con CloudWatch, puoi creare allarmi che controllano le metriche e ti inviano una notifica o eseguono un'altra azione quando viene superata una soglia. Per ulteriori informazioni sulla configurazione degli CloudWatch allarmi, consulta la Amazon CloudWatch User Guide.

Ti consigliamo di impostare allarmi per le seguenti metriche di Deadline Cloud:

LicensesInUse

Dimensioni:, FleetId LicenseEndpointId

Descrizione dell'allarme: questo allarme rileva quando le sessioni di licenza attive per un parco veicoli o un endpoint di licenze gestito dal servizio si avvicinano alla quota dell'account. In tal caso, puoi aumentare la quota dell'account per le sessioni di licenza. Visualizza le tue quote attuali e richiedi aumenti utilizzando Service Quotas. Per ulteriori informazioni, consulta la Guida per l'utente di Service Quotas.

Scopo: prevenire gli errori di checkout delle licenze monitorando l'utilizzo prima che raggiunga il limite di quota.

Statistica: Maximum

Soglia consigliata: 90% della quota della sessione di licenza

Giustificazione della soglia: imposta la soglia su una percentuale della quota, in modo da poter agire prima che raggiunga il limite.

Periodo: 1 minuto

Punti dati su cui attivare allarmi: 1

Periodi di valutazione: 1

Operatore di confronto: `GREATER_THAN_THRESHOLD`

Risorse aggiuntive

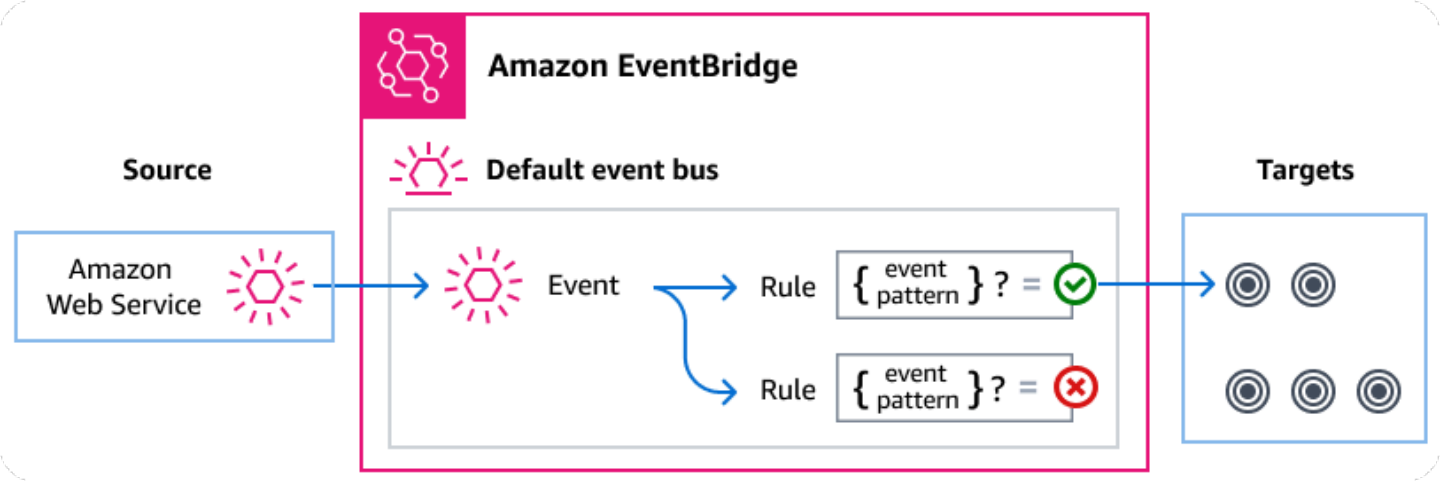
- Guida per CloudWatch l'utente di Amazon
- Guida per l’utente di Service Quotas

Gestione degli eventi Deadline Cloud utilizzando Amazon EventBridge

Amazon EventBridge è un servizio serverless che utilizza gli eventi per connettere tra loro i componenti delle applicazioni, semplificando la creazione di applicazioni scalabili basate sugli eventi. L’architettura basata sugli eventi è uno stile di creazione di sistemi software ad accoppiamento debole che interagiscono emettendo e rispondendo agli eventi. Gli eventi rappresentano un cambiamento in una risorsa o in un ambiente.

Come funziona:

Come molti AWS servizi, Deadline Cloud genera e invia eventi al bus di eventi predefinito. EventBridge (Il bus degli eventi predefinito viene fornito automaticamente in ogni AWS account.) Un router di eventi è un router che riceve eventi e li invia a zero o più destinazioni o target. Le regole associate al router di eventi valutano questi ultimi man mano che arrivano. Ogni regola verifica se un evento corrisponde al modello della regola. Se l’evento corrisponde, il router di eventi invia l’evento al/ai target specificato/i.



Argomenti

- Eventi Deadline Cloud
- Fornitura di eventi Deadline Cloud utilizzando regole EventBridge
- Riferimento dettagliato agli eventi di Deadline Cloud

Eventi Deadline Cloud

Deadline Cloud invia automaticamente i seguenti eventi al bus EventBridge eventi predefinito. Gli eventi che corrispondono allo schema di eventi di una regola vengono consegnati agli obiettivi specificati con la massima diligenza possibile. Gli eventi potrebbero essere consegnati fuori servizio.

Per ulteriori informazioni, consulta EventBridge gli eventi nella Guida Amazon EventBridge per l'utente.

Tipo di dettaglio dell'evento	Description
Soglia di budget raggiunta	Inviato quando una coda raggiunge una percentuale del budget assegnato.
Modifica dello stato del ciclo di vita del Job	Inviato quando viene apportata una modifica allo stato del ciclo di vita di un lavoro.
Modifica dello stato di Job Run	Inviato quando cambia lo stato generale delle attività di un job.
Fase: modifica dello stato del ciclo di vita	Inviato quando viene apportata una modifica allo stato del ciclo di vita di una fase di un lavoro.
Modifica dello stato di Step Run	Inviato quando lo stato generale delle attività in una fase cambia.
Modifica dello stato di esecuzione dell'operazione	Inviato quando lo stato di un'attività cambia.

Fornitura di eventi Deadline Cloud utilizzando regole EventBridge

Per fare in modo che il bus degli eventi EventBridge predefinito invii gli eventi di Deadline Cloud a una destinazione, devi creare una regola. Ogni regola contiene uno schema di eventi, che

EventBridge corrisponde a ogni evento ricevuto sull'event bus. Se i dati dell'evento corrispondono al modello di evento specificato, EventBridge invia quell'evento ai destinatari della regola.

Per istruzioni complete sulla creazione di regole per il bus degli eventi, consultate Creazione di regole che reagiscono agli eventi nella Guida per l'EventBridge utente.

Creazione di modelli di eventi che corrispondano agli eventi di Deadline Cloud

Ogni modello di eventi è un oggetto in formato JSON che contiene:

- Un attributo `source` che identifica il servizio che invia l'evento. Per gli eventi Deadline Cloud, la fonte è. `aws.deadline`
- (Facoltativo): Un attributo `detail-type` che contiene una serie di tipi di eventi da abbinare.
- (Facoltativo): Un attributo `detail` contenente qualsiasi altro dato relativo all'evento da abbinare.

Ad esempio, il seguente schema di eventi corrisponde a tutti gli eventi Fleet Size Recommendation Change per quanto specificato `farmId` per Deadline Cloud:

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

Per ulteriori informazioni sulla scrittura di modelli di eventi, consulta Event patterns nella Guida per l'EventBridge utente.

Riferimento dettagliato agli eventi di Deadline Cloud

Tutti gli eventi generati dai AWS servizi hanno un set comune di campi contenenti metadati relativi all'evento, ad esempio il AWS servizio che è l'origine dell'evento, l'ora in cui l'evento è stato generato, l'account e la regione in cui si è svolto l'evento e altri. Per le definizioni di questi campi generali, consultate il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

Inoltre, ogni evento ha un campo `detail` che contiene dati specifici per quel particolare evento. Il riferimento seguente definisce i campi di dettaglio per i vari eventi di Deadline Cloud.

Quando si utilizza EventBridge per selezionare e gestire gli eventi Deadline Cloud, è utile tenere presente quanto segue:

- Il `source` campo per tutti gli eventi di Deadline Cloud è impostato su. `aws.deadline`
- Il campo `detail-type` specifica il tipo di evento.

  Ad esempio, `Fleet Size Recommendation Change`.
- Il campo `detail` contiene i dati specifici di quel particolare evento.

Per informazioni sulla creazione di modelli di eventi che consentano alle regole di corrispondere agli eventi di Deadline Cloud, consulta Modelli di eventi nella Guida per l'Amazon EventBridge utente.

Per ulteriori informazioni sugli eventi e su come li EventBridge elabora, consulta Amazon EventBridge gli eventi nella Guida per l'Amazon EventBridge utente.

Argomenti

- Evento Soglia di budget raggiunta
- Evento relativo alla modifica delle dimensioni del parco veicoli
- Evento Job Lifecycle Status Change
- Evento di modifica dello stato di Job Run
- Evento Step Lifecycle Status Change
- Evento Step Run Status Change
- Evento di modifica dello stato di Task Run

Evento Soglia di budget raggiunta

È possibile utilizzare l'evento Budget Threshold Reached per monitorare la percentuale di budget utilizzata. Deadline Cloud invia eventi quando la percentuale utilizzata supera le seguenti soglie:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

La frequenza con cui Deadline Cloud invia gli eventi Budget Threshold Reached aumenta man mano che il budget si avvicina al limite. Questa frequenza consente di monitorare attentamente un budget man mano che si avvicina al limite e di agire per evitare spese eccessive. È inoltre possibile impostare soglie di budget personalizzate. Deadline Cloud invia un evento quando l'utilizzo supera le soglie personalizzate.

Se modifichi l'importo di un budget, la prossima volta che Deadline Cloud invia un evento Budget Threshold Reached, questo si basa sulla percentuale corrente del budget che è stato utilizzato. Ad esempio, se aggiungi $50 a un budget di $100 che ha raggiunto il limite, il successivo evento Budget Threshold Reached indica che il budget è al 75%.

Di seguito sono riportati i campi di dettaglio dell'`Budget Threshold Reached`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Budget Threshold Reached",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "budgetId": "budget-12345678900000000000000000000000",
        "thresholdInPercent": 0
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Budget Threshold Reached`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`budgetId`

L'identificatore del budget che ha raggiunto una soglia.

`thresholdInPercent`

La percentuale del budget che è stata utilizzata.

Evento relativo alla modifica delle dimensioni del parco veicoli

Quando configuri la tua flotta per utilizzare la scalabilità automatica basata sugli eventi, Deadline Cloud invia eventi che puoi utilizzare per gestire le tue flotte. Ciascuno di questi eventi contiene informazioni sulla dimensione attuale e sulla dimensione richiesta di una flotta. Per un esempio di utilizzo di un EventBridge evento e di un esempio di funzione Lambda per gestire l'evento, consulta Ridimensionamento automatico della flotta Amazon EC2 con la funzione di raccomandazione di scalabilità di Deadline Cloud.

L'evento di modifica dei consigli sulle dimensioni della flotta viene inviato quando si verifica quanto segue:

- Quando la dimensione del parco veicoli consigliata cambia e oldFleetSize differisce da newFleetSize.
- Quando il servizio rileva che la dimensione effettiva della flotta non corrisponde alla dimensione della flotta consigliata. È possibile ottenere le dimensioni effettive della flotta da WorkerCount nella risposta dell' GetFleet operazione. Ciò può accadere quando un'istanza Amazon EC2 attiva non riesce a registrarsi come operatore Deadline Cloud.

Di seguito sono riportati i campi relativi ai dettagli dell'`Fleet Size Recommendation Change`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
```

```
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Fleet Size Recommendation Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "fleetId": "fleet-12345678900000000000000000000000",
        "oldFleetSize": 1,
        "newFleetSize": 5,
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Fleet Size Recommendation Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`fleetId`

L'identificatore della flotta che necessita di un cambio di dimensione.

`oldFleetSize`

Le dimensioni attuali della flotta.

`newFleetSize`

La nuova dimensione consigliata per la flotta.

Evento Job Lifecycle Status Change

Quando crei o aggiorni un lavoro, Deadline Cloud imposta lo stato del ciclo di vita in modo che mostri lo stato dell'azione più recente avviata dall'utente.

Viene inviato un evento di modifica dello stato del ciclo di vita del lavoro per qualsiasi modifica dello stato del ciclo di vita, incluso quando il lavoro viene creato.

Di seguito sono riportati i campi di dettaglio dell'evento. `Job Lifecycle Status Change`

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Job Lifecycle Status Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`queueId`

L'identificatore della coda che contiene il lavoro.

`jobId`

L'identificatore del lavoro.

`previousLifecycleStatus`

Lo stato del ciclo di vita alla fine del lavoro. Questo campo non è incluso quando si invia un lavoro per la prima volta.

`lifecycleStatus`

Lo stato del ciclo di vita in cui il lavoro sta entrando.

Evento di modifica dello stato di Job Run

Un lavoro è composto da molte attività. Ogni attività ha uno stato. Lo stato di tutte le attività viene combinato per fornire uno stato generale di un lavoro. Per ulteriori informazioni, consulta Job states in Deadline Cloud nella Guida per l'utente di AWS Deadline Cloud.

Un evento di modifica dello stato del job run viene inviato quando:

- Il `taskRunStatus` campo combinato cambia.
- Il lavoro è richiesto, a meno che il lavoro non sia nello stato READY.

Un evento di modifica dello stato di esecuzione del job NON viene inviato quando:

- Il lavoro viene creato per la prima volta. Per monitorare la creazione di posti di lavoro, monitora gli eventi Job Lifecycle Status Change per individuare eventuali modifiche.
- Il `taskRunStatusCounts` campo del processo cambia ma lo stato di esecuzione del job task combinato rimane invariato.

Di seguito sono riportati i campi di dettaglio dell'`Job Run Status Change`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
       }
```

```
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Job Run Status Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`queueId`

L'identificatore della coda che contiene il lavoro.

`jobId`

L'identificatore del lavoro.

`previousTaskRunStatus`

L'operazione in esecuzione indica che il lavoro sta per terminare.

`taskRunStatus`

Lo stato di esecuzione dell'attività in cui è in corso il processo.

`taskRunStatusCounts`

Il numero di attività del lavoro in ogni stato.

Evento Step Lifecycle Status Change

Quando crei o aggiorni un evento, Deadline Cloud imposta lo stato del ciclo di vita del lavoro per descrivere lo stato dell'azione più recente avviata dall'utente.

Un evento di modifica dello stato del ciclo di vita di una fase viene inviato quando:

- Viene avviato un aggiornamento graduale (UPDATE_IN_PROGRESS).
- Un aggiornamento di un passaggio è stato completato con successo (UPDATE_SUCCEEDED).
- Aggiornamento di un passaggio non riuscito (UPDATE_FAILED).

Un evento non viene inviato quando il passo viene creato per la prima volta. Per monitorare la creazione dei passaggi, monitora gli eventi Job Lifecycle Status Change per verificare se sono state apportate modifiche.

Di seguito sono riportati i campi di dettaglio dell'`Step Lifecycle Status Change`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Step Lifecycle Status Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`queueId`

L'identificatore della coda che contiene il lavoro.

`jobId`

L'identificatore del lavoro.

`stepId`

L'identificatore della fase di lavoro corrente.

`previousLifecycleStatus`

Lo stato del ciclo di vita a cui sta per uscire il passaggio.

`lifecycleStatus`

Lo stato del ciclo di vita a cui sta entrando la fase.

Evento Step Run Status Change

Ogni fase di un lavoro è composta da molte attività. Ogni attività ha uno stato. Gli stati delle attività vengono combinati per fornire uno stato generale delle fasi e dei lavori.

Un evento di modifica dello stato di Step Run viene inviato quando:

- Le `taskRunStatus` modifiche combinate.
- Il passaggio è richiesto, a meno che non sia nello stato READY.

Un evento non viene inviato quando:

- Il passo viene creato per la prima volta. Per monitorare la creazione dei passaggi, monitora gli eventi Job Lifecycle Status Change per verificare se sono state apportate modifiche.
- La fase viene `taskRunStatusCounts` modificata, ma lo stato di esecuzione dell'operazione a fasi combinate rimane invariato.

Di seguito sono riportati i campi di dettaglio dell'`Step Run Status Change`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
```

```
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
        }
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Step Run Status Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`queueId`

L'identificatore della coda che contiene il lavoro.

`jobId`

L'identificatore del lavoro.

`stepId`

L'identificatore della fase di lavoro corrente.

`previousTaskRunStatus`

Lo stato di esecuzione da cui esce il passaggio.

`taskRunStatus`

Lo stato di esecuzione a cui sta entrando la fase.

`taskRunStatusCounts`

Il numero di attività della fase in ogni stato.

Evento di modifica dello stato di Task Run

Il `runStatus` campo viene aggiornato durante l'esecuzione dell'attività. Un evento viene inviato quando:

- Lo stato di esecuzione dell'attività cambia.
- L'attività è richiesta, a meno che non sia nello stato READY.

Un evento non viene inviato quando:

- L'attività viene creata per la prima volta. Per monitorare la creazione di attività, monitora gli eventi Job Lifecycle Status Change per individuare eventuali modifiche.

Di seguito sono riportati i campi di dettaglio dell'`Task Run Status Change`evento.

I `detail-type` campi `source` e sono inclusi di seguito perché contengono valori specifici per gli eventi di Deadline Cloud. Per le definizioni degli altri campi di metadati inclusi in tutti gli eventi, consulta il riferimento alla struttura degli eventi nella Guida per l'Amazon EventBridge utente.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Task Run Status Change",
    "source": "aws.aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
```

```
        "taskId": "task-12345678900000000000000000000000-0",
        "previousRunStatus": "RUNNING",
        "runStatus": "SUCCEEDED"
    }
}
```

`detail-type`

Identifica il tipo di evento.

Per questo evento, questo valore è`Fleet Size Recommendation Change`.

`source`

Identifica il servizio che ha generato l'evento. Per gli eventi Deadline Cloud, questo valore è`aws.deadline`.

`detail`

Un oggetto JSON contenente informazioni sull'evento. Il servizio che genera l'evento determina il contenuto di questo campo.

Per questo evento, questi dati includono:

`farmId`

L'identificatore dell'azienda agricola che contiene il lavoro.

`queueId`

L'identificatore della coda che contiene il lavoro.

`jobId`

L'identificatore del lavoro.

stepId

L'identificatore della fase di lavoro corrente.

`taskId`

L'identificatore dell'attività in esecuzione.

`previousRunStatus`

Lo stato di esecuzione in cui l'attività sta uscendo.

`runStatus`

Lo stato di esecuzione a cui sta entrando l'attività.

Interrogazione dei dati aggregati relativi alle statistiche di sessione utilizzando AWS CLI

Per tenere traccia dei costi, analizzare l'utilizzo delle risorse o identificare quali utenti consumano la maggior parte delle risorse, puoi utilizzare il AWS Command Line Interface (AWS CLI) per richiedere statistiche aggregate sulle sessioni per le tue farm AWS Deadline Cloud (Deadline Cloud). L'API per le statistiche delle sessioni fornisce dati su costi, runtime e utilizzo che puoi raggruppare in base a varie dimensioni come coda, flotta, tipo di istanza o utente.

L'interrogazione delle statistiche di sessione è un processo asincrono. Innanzitutto, si avvia una richiesta di aggregazione, quindi si recuperano i risultati utilizzando l'ID di aggregazione.

Avvio di una richiesta di aggregazione

Per avviare una richiesta di aggregazione, esegui il `start-sessions-statistics-aggregation` comando. L'esempio seguente raggruppa le statistiche in base all'ID utente per una coda specifica. Sostituisci il *`placeholder`* testo con le tue informazioni.

```
aws deadline start-sessions-statistics-aggregation \
    --farm-id farm-id \
    --resource-ids '{"queueIds":["queue-id"]}' \
    --start-time 2025-11-24T10:00:00Z \
    --end-time 2025-11-25T18:00:00Z \
    --group-by '["USER_ID"]' \
    --period HOURLY \
    --statistics '["SUM"]' \
    --timezone UTC-08:00 \
    --region region-name
```

Puoi raggruppare le statistiche in base ad altre dimensioni come `QUEUE_IDFLEET_ID`,`JOB_ID`,`INSTANCE_TYPE`, o`LICENSE_PRODUCT`. Per ulteriori informazioni su tutti i parametri disponibili, consulta start-sessions-statistics-aggregationla sezione AWS CLI Command Reference.

La risposta contiene un ID di aggregazione:

```
{
```

```
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Recupero dei risultati

Esegui il `get-sessions-statistics-aggregation` comando con l'ID di aggregazione per recuperare i risultati. Sostituisci il *`placeholder`* testo con le tue informazioni.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

L'esempio seguente mostra una risposta quando si raggruppano le statistiche per ID utente. Il `userId` campo contiene un UUID che è necessario mappare a un nome utente per identificare l'utente:

```
{
    "statistics": [
        {
            "userId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
            "count": 1,
            "costInUsd": {
                "sum": 0.0
            },
            "runtimeInSeconds": {
                "sum": 53.773
            },
            "aggregationStartTime": "2025-11-24T22:00:00Z",
            "aggregationEndTime": "2025-11-24T23:00:00Z"
        }
    ],
    "status": "COMPLETED"
}
```

Per trovare il nome utente associato a`userId`, vedere. the section called “Recupero dei metadati utente tramite UserID”

Per ulteriori informazioni sull'API, consulta il Deadline Cloud API Reference.

Argomenti

- the section called “Recupero dei metadati utente tramite UserID”

Recupero dei metadati e degli attributi degli utenti tramite UserID in un archivio di identità

> **Note**
>
> Questa procedura si applica anche al `createdBy` campo restituito dall'SearchJobsAPI, che utilizza lo stesso formato di ID utente.

Il `userId` campo delle statistiche della sessione contiene uno dei seguenti valori:

- Un ruolo AWS Identity and Access Management (IAM) ARN, ad esempio:. `arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`
- Un ID utente (UUID) di IAM Identity Center, ad esempio:. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Per il ruolo IAM ARNs, il nome utente è visibile nell'ARN stesso. Per gli utenti IAM Identity Center IDs, puoi cercare il nome utente utilizzando l'API IAM Identity Center Identity Store.

Per identificare il nome utente associato a un ID utente IAM Identity Center, utilizza la seguente procedura. Prima di iniziare, ottieni l'ID Identity Store dalle impostazioni del tuo IAM Identity Center. Per ulteriori informazioni, consulta the section called “Come trovare il tuo ID Identity Store”.

Per mappare un ID utente

1. Esegui il comando seguente, sostituendolo *`IdentityStoreId`* con il tuo ID Identity Store e *`userUUID`* con la `userId` risposta delle statistiche della sessione:

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Controlla la risposta, che include il nome utente:

```
{
    "UserName": "jdoe",
    "UserId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
    "Name": {
        "FamilyName": "Doe",
        "GivenName": "Jane"
    },
    "DisplayName": "Jane Doe",
    "Emails": [{
        "Value": "jdoe@example.com",
        "Type": "work",
        "Primary": true
    }],
    "IdentityStoreId": "d-xxxxxxxxxx"
}
```

Come trovare il tuo ID Identity Store

Per mappare gli utenti IDs ai nomi utente, è necessario l'ID Identity Store. Puoi trovare l'ID Identity Store utilizzando la console IAM Identity Center o il AWS CLI.

Console

Per trovare il tuo ID Identity Store utilizzando la console, utilizza la seguente procedura.

1. Accedi alla console di AWS gestione e apri la console IAM Identity Center.
2. Nel pannello di navigazione scegli Impostazioni.
3. Copia il valore dell'ID IAM Identity Center Identity Store. Il formato è `d-xxxxxxxxxx`.

AWS CLI

Esegui il comando seguente, sostituendolo *`region-name`* con la regione in cui è configurata l'istanza di IAM Identity Center:

```
aws sso-admin list-instances --region region-name
```

La risposta include`IdentityStoreId`:

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
    ]
}
```

Verifica della mappatura degli utenti

Dopo aver mappato un ID utente a un nome utente, puoi verificare nella console IAM Identity Center che l'ID utente corrisponda all'utente previsto. Per verificare la mappatura degli utenti, utilizza la seguente procedura.

1. Accedi alla console di AWS gestione e apri la console IAM Identity Center.
2. Nel pannello di navigazione, seleziona Utenti.
3. Scegli il nome utente dalla AWS CLI risposta.
4. Nella sezione Informazioni generali, verifica che l'ID utente `userId` corrisponda alle statistiche della sessione.

Risorse aggiuntive

- Guida per l'utente di IAM Identity Center
- Riferimento all'API IAM Identity Center Identity Store

Sicurezza in Deadline Cloud

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gira Servizi AWS su Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. Third-party i revisori testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per maggiori informazioni sui programmi di conformità applicabili AWS Deadline Cloud, consulta Scope by Compliance Program Servizi AWSServizi AWS in Scope .
- Sicurezza nel cloud: la tua responsabilità è determinata dal Servizio AWS materiale che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione aiuta a capire come applicare il modello di responsabilità condivisa durante l'utilizzo Deadline Cloud. Negli argomenti seguenti viene illustrato come eseguire la configurazione Deadline Cloud per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a utilizzarne altri Servizi AWS che ti aiutano a monitorare e proteggere Deadline Cloud le tue risorse.

Argomenti

- Protezione dei dati in Deadline Cloud
- Identity and Access Management in Deadline Cloud
- Convalida della conformità per Deadline Cloud
- Resilienza in Deadline Cloud
- Sicurezza dell'infrastruttura in Deadline Cloud
- Configurazione e analisi delle vulnerabilità in Deadline Cloud
- Cross-service confusa prevenzione sostitutiva
- Accesso AWS Deadline Cloud utilizzando un endpoint di interfaccia (AWS PrivateLink)
- Ambienti di rete con restrizioni

- Le migliori pratiche di sicurezza per Deadline Cloud

Protezione dei dati in Deadline Cloud

Il modello di responsabilità AWS condivisa Il modello di si applica alla protezione dei dati in AWS Deadline Cloud. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta Domande frequenti sulla privacy dei dati . Per informazioni sulla protezione dei dati in Europa, consulta il General Data Protection Regulation (GDPR) Center.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e di configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori o Servizi AWS utilizzi la console, l'API Deadline Cloud o gli SDK. AWS CLI AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per

i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

I dati inseriti nei campi dei nomi nei modelli di Deadline Cloud lavoro possono anche essere inclusi nei registri di fatturazione o diagnostica e non devono contenere informazioni riservate o sensibili.

Argomenti

- Crittografia dei dati a riposo
- Crittografia dei dati in transito
- Gestione delle chiavi
- Inter-network privacy del traffico
- Disattivazione

Crittografia dei dati a riposo

AWS Deadline Cloud protegge i dati sensibili crittografandoli quando sono inattivi utilizzando le chiavi di crittografia memorizzate in AWS Key Management Service ()AWS KMS. La crittografia a riposo è disponibile Regioni AWS ovunque Deadline Cloud sia disponibile.

La crittografia dei dati significa che i dati sensibili salvati su disco non sono leggibili da un utente o da un'applicazione senza una chiave valida. Solo chi dispone di una chiave gestita valida può decrittografare i dati.

Deadline Cloud elimina i volumi di Amazon Elastic Block Store quando le istanze di fleet worker gestite dal servizio terminano.

Per informazioni su come vengono Deadline Cloud utilizzati i dati inattivi AWS KMS per crittografare i dati inattivi, consulta. Gestione delle chiavi

Crittografia dei dati in transito

Per i dati in transito, AWS Deadline Cloud utilizza Transport Layer Security (TLS) 1.2 o 1.3 per crittografare i dati inviati tra il servizio e i lavoratori. È richiesto TLS 1.2 ed è consigliato TLS 1.3. Inoltre, se utilizzi un cloud privato virtuale (VPC), puoi utilizzare AWS PrivateLink per stabilire una connessione privata tra il tuo VPC e. Deadline Cloud

Gestione delle chiavi

Quando crei una nuova farm, puoi scegliere una delle seguenti chiavi per crittografare i dati della tua fattoria:

- AWS chiave KMS proprietaria: tipo di crittografia predefinito se non si specifica una chiave quando si crea la farm. La chiave KMS è di proprietà di. AWS Deadline Cloud Non puoi visualizzare, gestire o utilizzare chiavi AWS di proprietà. Tuttavia, non è necessario intraprendere alcuna azione per proteggere le chiavi che crittografano i dati. Per ulteriori informazioni, consulta le chiavi AWS possedute nella guida per gli AWS Key Management Service sviluppatori.
- Chiave KMS gestita dal cliente: si specifica una chiave gestita dal cliente quando si crea una farm. Tutto il contenuto all'interno della farm è crittografato con la chiave KMS. La chiave è memorizzata nel tuo account e viene creata, posseduta e gestita da te e vengono applicati dei AWS KMS costi. Hai il pieno controllo sulla chiave KMS. Puoi eseguire attività come:
  - Stabilire e mantenere le politiche chiave
  - Stabilire e mantenere le policy e le sovvenzioni IAM
  - Abilitare e disabilitare le policy delle chiavi
  - Aggiungere tag
  - Creare alias delle chiavi

  Non è possibile ruotare manualmente una chiave di proprietà del cliente utilizzata in un' Deadline Cloud azienda agricola. È supportata la rotazione automatica della chiave.

  Per ulteriori informazioni, consulta Customer owned keys nella AWS Key Management Service Developer Guide.

  Per creare una chiave gestita dal cliente, segui i passaggi per la creazione di chiavi gestite dal cliente simmetriche nella Guida per gli AWS Key Management Service sviluppatori.

In che modo Deadline Cloud utilizzi AWS KMS borse di studio

Deadline Cloud richiede una concessione per utilizzare la chiave gestita dal cliente. Quando crei una farm crittografata con una chiave gestita dal cliente, Deadline Cloud crea una concessione per tuo conto inviando una `CreateGrant` richiesta AWS KMS per ottenere l'accesso alla chiave KMS specificata.

Deadline Cloud utilizza più sovvenzioni. Ogni concessione viene utilizzata da una parte diversa Deadline Cloud che deve crittografare o decrittografare i dati. Deadline Cloud utilizza anche sovvenzioni per consentire l'accesso ad altri AWS servizi utilizzati per archiviare dati per tuo conto, come Amazon Simple Storage Service, Amazon Elastic Block Store o OpenSearch.

Le sovvenzioni che consentono Deadline Cloud di gestire le macchine in una flotta gestita dai servizi includono un numero di Deadline Cloud account e un ruolo `GranteePrincipal` anziché un responsabile del servizio. Sebbene non sia tipico, ciò è necessario per crittografare i volumi Amazon EBS per i lavoratori delle flotte gestite dai servizi utilizzando la chiave KMS gestita dal cliente specificata per la farm.

Policy della chiave gestita dal cliente

Le policy della chiave controllano l'accesso alla chiave gestita dal cliente. Ogni chiave deve avere esattamente una policy chiave che contenga istruzioni che determinano chi può utilizzare la chiave e come può usarla. Quando si crea la chiave gestita dal cliente, è possibile specificare una politica chiave. Per ulteriori informazioni, consulta Gestione dell'accesso alle chiavi gestite dal cliente nella Guida per gli sviluppatori di AWS Key Management Service .

Policy IAM minima per CreateFarm

Per utilizzare la chiave gestita dal cliente per creare farm utilizzando la console o il funzionamento dell'`CreateFarm`API, devono essere consentite le seguenti operazioni AWS KMS API:

- `kms:CreateGrant`: aggiunge una concessione a una chiave gestita dal cliente. Concede l'accesso della console a una AWS KMS chiave specificata. Per maggiori informazioni, consulta Using grants nella guida per AWS Key Management Service sviluppatori.
- `kms:Decrypt`— Consente di Deadline Cloud decrittografare i dati nella fattoria.
- `kms:DescribeKey`— Fornisce i dettagli chiave gestiti dal cliente per consentire Deadline Cloud la convalida della chiave.
- `kms:GenerateDataKey`— Consente di Deadline Cloud crittografare i dati utilizzando una chiave dati unica.

La seguente dichiarazione politica concede le autorizzazioni necessarie per l'operazione. `CreateFarm`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Policy IAM minima per operazioni di sola lettura

Utilizzare la chiave gestita dal cliente per Deadline Cloud operazioni di sola lettura, ad esempio per ottenere informazioni su fattorie, code e flotte. Le seguenti operazioni AWS KMS API devono essere consentite:

- `kms:Decrypt`— Consente di Deadline Cloud decrittografare i dati nella farm.
- `kms:DescribeKey`— Fornisce i dettagli chiave gestiti dal cliente per consentire Deadline Cloud la convalida della chiave.

La seguente dichiarazione politica concede le autorizzazioni necessarie per le operazioni di sola lettura.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Policy IAM minima per le operazioni di lettura/scrittura

Utilizzare la chiave gestita dal cliente per Deadline Cloud operazioni di lettura/scrittura, come la creazione e l'aggiornamento di fattorie, code e flotte. Le seguenti operazioni AWS KMS API devono essere consentite:

- `kms:Decrypt`— Consente di Deadline Cloud decrittografare i dati nella farm.
- `kms:DescribeKey`— Fornisce i dettagli chiave gestiti dal cliente per consentire Deadline Cloud la convalida della chiave.
- `kms:GenerateDataKey`— Consente di Deadline Cloud crittografare i dati utilizzando una chiave dati unica.

La seguente dichiarazione politica concede le autorizzazioni necessarie per l'operazione. `CreateFarm`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Monitoraggio delle chiavi di crittografia

Quando utilizzi una chiave gestita AWS KMS dal cliente con le tue Deadline Cloud farm, puoi utilizzare AWS CloudTrailAmazon CloudWatch Logs per tenere traccia delle richieste Deadline Cloud inviate a AWS KMS.

CloudTrail evento per borse di studio

L' CloudTrail evento di esempio seguente si verifica quando vengono create le sovvenzioni, in genere quando si chiama l'`CreateFarm`operazione`CreateMonitor`, or`CreateFleet`.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


CloudTrail evento per la decrittografia

L' CloudTrail evento di esempio seguente si verifica quando si decrittografano i valori utilizzando la chiave KMS gestita dal cliente.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
```

```
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

CloudTrail evento per la crittografia

L' CloudTrail evento di esempio seguente si verifica quando si crittografano i valori utilizzando la chiave KMS gestita dal cliente.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 32,
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
```

```
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Eliminazione di una chiave KMS gestita dal cliente

L'eliminazione di una chiave KMS gestita dal cliente in AWS Key Management Service (AWS KMS) è distruttiva e potenzialmente pericolosa. Elimina in modo irreversibile il materiale chiave e tutti i metadati associati alla chiave. Dopo l'eliminazione di una chiave KMS gestita dal cliente, non è più possibile decrittografare i dati crittografati con quella chiave. L'eliminazione della chiave significa che i dati diventano irrecuperabili.

Questo è il motivo per cui AWS KMS offre ai clienti un periodo di attesa fino a 30 giorni prima di eliminare la chiave KMS. Il periodo di attesa predefinito è di 30 giorni.

Informazioni sul periodo di attesa

Poiché eliminare una chiave KMS gestita dal cliente è distruttivo e potenzialmente pericoloso, ti chiediamo di impostare un periodo di attesa di 7—30 giorni. Il periodo di attesa predefinito è di 30 giorni.

Tuttavia, il periodo di attesa effettivo potrebbe essere fino a 24 ore più lungo del periodo pianificato. Per ottenere la data e l'ora effettive in cui la chiave verrà eliminata, utilizzare l'DescribeKeyoperazione. È inoltre possibile visualizzare la data di eliminazione pianificata di una

chiave nella AWS KMS console nella pagina di dettaglio della chiave, nella sezione Configurazione generale. Nota il fuso orario.

Durante il periodo di attesa, lo stato e lo stato della chiave gestita dal cliente sono In attesa di eliminazione.

- Una chiave KMS gestita dal cliente in attesa di eliminazione non può essere utilizzata in alcuna operazione crittografica.
- AWS KMS non ruota le chiavi di supporto delle chiavi KMS gestite dal cliente in attesa di eliminazione.

Per ulteriori informazioni sull'eliminazione di una chiave KMS gestita dal cliente, consulta Eliminazione delle chiavi principali del cliente nella Guida per gli sviluppatori.AWS Key Management Service

Inter-network privacy del traffico

AWS Deadline Cloud supporta Amazon Virtual Private Cloud (Amazon VPC) per proteggere le connessioni. Amazon VPC offre funzionalità che puoi utilizzare per aumentare e monitorare la sicurezza del tuo cloud privato virtuale (VPC).

Puoi configurare una flotta gestita dal cliente (CMF) con istanze Amazon Elastic Compute Cloud (Amazon EC2) eseguite all'interno di un VPC. Implementando gli endpoint Amazon VPC da AWS PrivateLink utilizzare, il traffico tra i lavoratori del tuo CMF e l'endpoint rimane all'interno Deadline Cloud del tuo VPC. Inoltre, puoi configurare il tuo VPC per limitare l'accesso a Internet alle tue istanze.

Nelle flotte gestite dai servizi, i lavoratori non sono raggiungibili da Internet, ma hanno accesso a Internet e si connettono al servizio tramite Internet. Deadline Cloud Ogni flotta gestita dai servizi opera nella propria rete isolata e le istanze Worker rimangono dedicate ai singoli clienti.

Disattivazione

AWS Deadline Cloud raccoglie determinate informazioni operative per aiutarci a svilupparci e migliorare. Deadline Cloud I dati raccolti includono elementi come l'ID del tuo AWS account e l'ID utente, in modo che possiamo identificarti correttamente in caso di problemi con. Deadline Cloud Raccogliamo anche informazioni Deadline Cloud specifiche, come i Resource ID (un FarmID o QueueID se applicabile), il nome del prodotto (ad esempio, JobAttachments WorkerAgent, e altro) e la versione del prodotto.

Puoi scegliere di rinunciare a questa raccolta di dati utilizzando la configurazione dell'applicazione. Ogni computer con cui interagisce Deadline Cloud, sia le postazioni di lavoro dei clienti che gli addetti alla flotta, deve disattivarlo separatamente.

Deadline Cloud monitor - desktop

Deadline Cloud monitor - desktop raccoglie informazioni operative, ad esempio quando si verificano arresti anomali e quando l'applicazione viene aperta, per aiutarci a sapere quando si verificano problemi con l'applicazione. Per disattivare la raccolta di queste informazioni operative, vai alla pagina delle impostazioni e deseleziona Attiva la raccolta dei dati per misurare le prestazioni di Deadline Cloud Monitor.

Dopo la disattivazione, il monitor desktop non invia più i dati operativi. Tutti i dati raccolti in precedenza vengono conservati e possono ancora essere utilizzati per migliorare il servizio. Per ulteriori informazioni, consulta le Domande frequenti sulla privacy dei dati.

AWS Deadline Cloud CLI e strumenti

La AWS Deadline Cloud CLI, i mittenti e l'agente di lavoro raccolgono tutti informazioni operative, ad esempio quando si verificano arresti anomali e quando vengono inviati lavori, per aiutarci a sapere quando si verificano problemi con queste applicazioni. Per rinunciare alla raccolta di queste informazioni operative, utilizza uno dei seguenti metodi:

- Nel terminale, inserisci**`deadline config set telemetry.opt_out true`**.

  Ciò disattiverà la CLI, i mittenti e il worker agent quando viene eseguito come utente corrente.
- Quando installi il Deadline Cloud worker agent, aggiungi l'argomento della **`--telemetry-opt-out`** riga di comando. Ad esempio, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.
- Prima di eseguire l'agente di lavoro, la CLI o il mittente, imposta una variabile di ambiente: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Dopo la disattivazione, gli Deadline Cloud strumenti non inviano più i dati operativi. Tutti i dati raccolti in precedenza vengono conservati e possono ancora essere utilizzati per migliorare il servizio. Per ulteriori informazioni, consulta le Domande frequenti sulla privacy dei dati.

Identity and Access Management in Deadline Cloud

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Deadline Cloud. IAM è uno strumento Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso tramite policy
- Come funziona Deadline Cloud con IAM
- Identity-based esempi di policy per Deadline Cloud
- AWS politiche gestite per Deadline Cloud
- Ruoli di servizio
- Risoluzione dei problemi AWS Identità e accesso a Deadline Cloud

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi AWS Identità e accesso a Deadline Cloud)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come funziona Deadline Cloud con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Identity-based esempi di policy per Deadline Cloud)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/

Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Identity-based politiche

Identity-based le politiche sono documenti di policy sulle autorizzazioni JSON che alleghi a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Identity-based le politiche possono essere politiche in linea (incorporate direttamente in una singola identità) o politiche gestite (politiche autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Resource-based politiche

Resource-based le politiche sono documenti di policy JSON allegati a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Resource-based le politiche sono politiche in linea che si trovano in quel servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): specifica il numero massimo di autorizzazioni per un'organizzazione o un'unità organizzativa (OU) in AWS Organizations. Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Policy di controllo delle risorse (RCP): imposta le autorizzazioni massime disponibili per le risorse degli account. Per ulteriori informazioni, consulta Policy di controllo delle risorse (RCP) nella Guida per l'utente di AWS Organizations .
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Deadline Cloud con IAM

Prima di utilizzare IAM per gestire l'accesso a Deadline Cloud, scopri quali funzionalità IAM sono disponibili per l'uso con Deadline Cloud.

Funzionalità IAM che puoi utilizzare con AWS Deadline Cloud

Funzionalità IAM	Supporto Deadline Cloud
Identity-based politiche	Sì
Resource-based politiche	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Inoltro delle sessioni di accesso (FAS)	Sì
Ruoli di servizio	Sì
Service-linked ruoli	No

Per avere una visione di alto livello di come Deadline Cloud e altri Servizi AWS funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Identity-based politiche per Deadline Cloud

Supporta le policy basate sull'identità: sì

Identity-based le policy sono documenti di policy sulle autorizzazioni JSON che puoi allegare a un'identità, ad esempio un utente IAM, un gruppo di utenti o un ruolo. Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Identity-based esempi di policy per Deadline Cloud

Per visualizzare esempi di politiche basate sull'identità di Deadline Cloud, consulta. Identity-based esempi di policy per Deadline Cloud

Resource-based politiche all'interno di Deadline Cloud

Supporta le policy basate su risorse: no

Resource-based le politiche sono documenti di policy JSON allegati a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni politiche per Deadline Cloud

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni di eseguire l'operazione associata.

Per visualizzare un elenco delle azioni di Deadline Cloud, consulta Azioni definite da AWS Deadline Cloud nel Service Authorization Reference.

Le azioni politiche in Deadline Cloud utilizzano il seguente prefisso prima dell'azione:

```
deadline
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Per visualizzare esempi di politiche basate sull'identità di Deadline Cloud, consulta. Identity-based esempi di policy per Deadline Cloud

Risorse politiche per Deadline Cloud

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Deadline Cloud e dei relativi ARN, consulta Risorse definite da AWS Deadline Cloud nel Service Authorization Reference. Per sapere con quali azioni puoi specificare l'ARN di ogni risorsa, vedi Azioni definite da AWS Deadline Cloud.

Per visualizzare esempi di politiche basate sull'identità di Deadline Cloud, consulta. Identity-based esempi di policy per Deadline Cloud

Chiavi relative alle condizioni delle policy per Deadline Cloud

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco delle chiavi di condizione di Deadline Cloud, consulta Condition keys for AWS Deadline Cloud nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, vedi Azioni definite da AWS Deadline Cloud.

Per visualizzare esempi di politiche basate sull'identità di Deadline Cloud, consulta. Identity-based esempi di policy per Deadline Cloud

ACL in Deadline Cloud

Supporta le ACL: no

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Deadline Cloud

Supporta ABAC (tag nelle policy): sì

Attribute-based il controllo degli accessi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base ad attributi chiamati tag. È possibile allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Deadline Cloud

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono un accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Sessioni di accesso diretto per Deadline Cloud

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale chiamante e Servizio AWS, in combinazione con la richiesta, di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Ruoli di servizio per Deadline Cloud

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità di Deadline Cloud. Modifica i ruoli di servizio solo quando Deadline Cloud fornisce indicazioni in tal senso.

Service-linked ruoli per Deadline Cloud

Supporta i ruoli collegati ai servizi: no

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo di eseguire un'azione per conto dell'utente. Service-linked i ruoli appaiono nel tuo Account AWS account e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un servizio `Yes` nella colonna del Service-linked ruolo. Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Identity-based esempi di policy per Deadline Cloud

Per impostazione predefinita, gli utenti e i ruoli non sono autorizzati a creare o modificare le risorse di Deadline Cloud. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per i dettagli sulle azioni e sui tipi di risorse definiti da Deadline Cloud, incluso il formato degli ARN per ciascun tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per AWS Deadline Cloud nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console Deadline Cloud
- Politica di accesso alla console

- Politica per l'invio di lavori a una coda
- Politica per consentire la creazione di un endpoint di licenza
- Politica per consentire il monitoraggio di una coda specifica della fattoria

Best practice per le policy

Identity-based le politiche determinano se qualcuno può creare, accedere o eliminare le risorse di Deadline Cloud nel tuo account. Queste operazioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.

- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Deadline Cloud

Per accedere alla console AWS Deadline Cloud, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Deadline Cloud presenti nel tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso AWS CLI o l'API. AWS Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano ancora utilizzare la console Deadline Cloud, collega anche Deadline Cloud *ConsoleAccess* o la policy *ReadOnly* AWS gestita alle entità. Per maggiori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente di IAM.

Politica di accesso alla console

Per concedere l'accesso a tutte le funzionalità della console Deadline Cloud, collega questa politica di identità a un utente o ruolo a cui desideri avere accesso completo.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
```

```
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
            "vpc-lattice:GetResourceConfiguration",
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
```

```
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ValidateDependencies",
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "RoleSelection",
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles",
"iam:ListAttachedRolePolicies"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
```

```
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
        "Resource": "*"
    },
    {
        "Sid": "OrganizationAndIdentityCenterIdentification",
        "Effect": "Allow",
        "Action": [
            "sso:ListDirectoryAssociations",
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "sso:DescribeRegisteredRegions",
            "sso:GetManagedApplicationInstance",
            "sso:GetSharedSsoConfiguration",
            "sso:ListInstances",
            "sso:GetApplicationAssignmentConfiguration",
            "sso:GetSSOStatus",
            "sso:ListRegions",
            "sso:DescribeRegion"
    ],
        "Resource": "*"
    },
    {
        "Sid": "ManagedDeadlineCloudIDCApplication",
        "Effect": "Allow",
        "Action": [
            "sso:CreateApplication",
            "sso:PutApplicationAssignmentConfiguration",
            "sso:PutApplicationAuthenticationMethod",
            "sso:PutApplicationGrant",
            "sso:DeleteApplication",
            "sso:UpdateApplication"
    ],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
```

```
    },
    {
        "Sid": "ChooseSecret",
        "Effect": "Allow",
        "Action": ["secretsmanager:ListSecrets"],
        "Resource": "*"
    },
    {
        "Sid": "DeadlineMembershipActions",
        "Effect": "Allow",
        "Action": [
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
            "deadline:ListFarmMembers",
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
```

```
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
            "deadline:UpdateLimit",
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
```

```
            "deadline:DeleteMeteredProduct",
            "deadline:GetMonitorSettings",
            "deadline:UpdateMonitorSettings"
        ],
        "Resource": ["*"]
      }]
}
```

Politica per l'invio di lavori a una coda

In questo esempio, si crea una politica ristretta che concede l'autorizzazione a inviare lavori a una coda specifica in una fattoria specifica.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
}
```

Politica per consentire la creazione di un endpoint di licenza

In questo esempio, si crea una policy ristretta che concede le autorizzazioni necessarie per creare e gestire gli endpoint di licenza. Utilizza questa politica per creare l'endpoint di licenza per il VPC associato alla tua farm.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
```

```
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
    }]
}
```

Politica per consentire il monitoraggio di una coda specifica della fattoria

In questo esempio, si crea una politica ristretta che concede l'autorizzazione a monitorare i lavori in una coda specifica per una determinata azienda agricola.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
```

```
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
            "deadline:GetSessionAction"
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
    }]
}
```

AWS politiche gestite per Deadline Cloud

Una politica AWS gestita è una politica autonoma creata e amministrata da. AWS AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

AWS politica gestita: AWSDeadlineCloud-FleetWorker

Puoi allegare la `AWSDeadlineCloud-FleetWorker` policy alle tue identità AWS Identity and Access Management (IAM).

Questa politica concede ai lavoratori di questa flotta le autorizzazioni necessarie per connettersi e ricevere attività dal servizio.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente ai dirigenti di gestire i lavoratori di una flotta.

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-FleetWorkerla guida di riferimento di AWS Managed Policy.

AWS politica gestita: AWSDeadlineCloud-WorkerHost

È possibile allegare la policy `AWSDeadlineCloud-WorkerHost` alle identità IAM.

Questa politica concede le autorizzazioni necessarie per connettersi inizialmente al servizio. Può essere usato come profilo di istanza Amazon Elastic Compute Cloud (Amazon EC2).

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente all'utente di creare lavoratori, assumere il ruolo della flotta per i lavoratori e applicare tag ai lavoratori

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-WorkerHostla guida di riferimento di AWS Managed Policy.

AWS politica gestita: AWSDeadlineCloud-UserAccessFarms

È possibile allegare la policy `AWSDeadlineCloud-UserAccessFarms` alle identità IAM.

Questa politica consente agli utenti di accedere ai dati delle aziende agricole in base alle aziende agricole di cui sono membri e al loro livello di iscrizione.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente all'utente di accedere ai dati dell'azienda agricola.
- `ec2`— Consente agli utenti di visualizzare i dettagli sui tipi di istanze Amazon EC2.
- `identitystore`— Consente agli utenti di visualizzare i nomi di utenti e gruppi.
- `kms`— Consente agli utenti di configurare AWS Key Management Service (AWS KMS) chiavi gestite dal cliente per la loro istanza AWS IAM Identity Center (IAM Identity Center).

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-UserAccessFarmsla guida di riferimento di AWS Managed Policy.

AWS politica gestita: AWSDeadlineCloud-UserAccessFleets

È possibile allegare la policy `AWSDeadlineCloud-UserAccessFleets` alle identità IAM.

Questa politica consente agli utenti di accedere ai dati della flotta in base alle aziende agricole di cui sono membri e al loro livello di iscrizione.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente all'utente di accedere ai dati dell'azienda agricola.
- `ec2`— Consente agli utenti di visualizzare i dettagli sui tipi di istanze Amazon EC2.
- `identitystore`— Consente agli utenti di visualizzare i nomi di utenti e gruppi.

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-UserAccessFleetsla guida di riferimento di AWS Managed Policy.

AWS politica gestita: AWSDeadlineCloud-UserAccessJobs

È possibile allegare la policy `AWSDeadlineCloud-UserAccessJobs` alle identità IAM.

Questa politica consente agli utenti di accedere ai dati sulle offerte di lavoro in base alle aziende agricole di cui sono membri e al loro livello di iscrizione.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente all'utente di accedere ai dati dell'azienda agricola.
- `ec2`— Consente agli utenti di visualizzare i dettagli sui tipi di istanze Amazon EC2.
- `identitystore`— Consente agli utenti di visualizzare i nomi di utenti e gruppi.

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-UserAccessJobsla guida di riferimento di AWS Managed Policy.

AWS politica gestita: AWSDeadlineCloud-UserAccessQueues

È possibile allegare la policy `AWSDeadlineCloud-UserAccessQueues` alle identità IAM.

Questa politica consente agli utenti di accedere ai dati delle code in base alle farm di cui sono membri e al loro livello di iscrizione.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `deadline`— Consente all'utente di accedere ai dati dell'azienda agricola.
- `ec2`— Consente agli utenti di visualizzare i dettagli sui tipi di istanze Amazon EC2.
- `identitystore`— Consente agli utenti di visualizzare i nomi di utenti e gruppi.

Per un elenco in JSON dei dettagli della policy, consulta AWSDeadlineCloud-UserAccessQueuesla guida di riferimento di AWS Managed Policy.

Deadline Cloud si aggiorna a AWS policy gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Deadline Cloud da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina della cronologia dei documenti di Deadline Cloud.

Modifica	Descrizione	Data
AWSDeadlineCloud-UserAccessFarms— Modifica	Deadline Cloud ha aggiunto una nuova azione `kms:Decry`	22 dicembre 2025

Modifica	Descrizione	Data
	`pt` che ti consente di utilizzare una chiave AWS KMS gestita dal cliente con la tua istanza IAM Identity Center.	
AWSDeadlineCloud-WorkerHost— Cambia	Deadline Cloud ha aggiunto nuove azioni `deadline:TagResource` e ti ha `deadline:ListTagsForResource` permesso di aggiungere e visualizzare i tag associati ai lavoratori della tua flotta.	30 maggio 2025
AWSDeadlineCloud-UserAccessFarms— Modifica AWSDeadlineCloud-UserAccessJobs— Cambiare AWSDeadlineCloud-UserAccessQueues— Cambiare	Deadline Cloud ha aggiunto nuove azioni `deadline:GetJobTemplate` e ti ha consentito di `deadline:ListJobParameterDefinitions` inviare nuovamente i lavori.	7 ottobre 2024
Deadline Cloud ha iniziato a tracciare le modifiche	Deadline Cloud ha iniziato a tenere traccia delle modifiche alle sue politiche AWS gestite.	2 aprile 2024

Ruoli di servizio

In che modo Deadline Cloud utilizza i ruoli del servizio IAM

Deadline Cloud assume automaticamente i ruoli IAM e fornisce credenziali temporanee ai lavoratori, ai lavori e al monitor di Deadline Cloud. Questo approccio elimina la gestione manuale delle credenziali mantenendo al contempo la sicurezza attraverso il controllo degli accessi basato sui ruoli.

Quando crei monitor, flotte e code, specifichi i ruoli IAM che Deadline Cloud assume per tuo conto. I lavoratori e il monitor Deadline Cloud ricevono quindi le credenziali temporanee di accesso da questi ruoli. Servizi AWS

Ruolo della flotta

Configura un ruolo della flotta per concedere ai lavoratori di Deadline Cloud le autorizzazioni necessarie per ricevere il lavoro e segnalare lo stato di avanzamento del lavoro.

Di solito non devi configurare questo ruolo da solo. Questo ruolo può essere creato per te nella console Deadline Cloud per includere le autorizzazioni necessarie. Utilizza la seguente guida per comprendere le specifiche di questo ruolo per la risoluzione dei problemi.

Quando crei o aggiorni le flotte a livello di codice, specifica l'ARN del ruolo della flotta utilizzando le operazioni o API. `CreateFleet` `UpdateFleet`

Cosa fa il ruolo della flotta

Il ruolo della flotta fornisce ai lavoratori le autorizzazioni per:

- Ricevi nuovi lavori e segnala lo stato di avanzamento del lavoro in corso al servizio Deadline Cloud
- Gestisci il ciclo di vita e lo status dei lavoratori
- Registra gli eventi di registro su Amazon CloudWatch Logs per i log dei lavoratori

Imposta la policy di fiducia dei ruoli della flotta

Il tuo ruolo nella flotta deve basarsi sul servizio Deadline Cloud ed essere circoscritto alla tua azienda agricola specifica.

Come best practice, la politica di fiducia dovrebbe includere le condizioni di sicurezza per la protezione di Confused Deputy. Per saperne di più sulla protezione di Confused Deputy, consulta Confused Deputy nella Guida per l'utente di Deadline Cloud.

- `aws:SourceAccount`assicura che solo le risorse dello stesso Account AWS possano assumere questo ruolo.
- `aws:SourceArn`limita l'assunzione del ruolo a una specifica Deadline Cloud farm.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

Allega le autorizzazioni del ruolo Fleet

Allega la seguente politica AWS gestita al tuo ruolo nella flotta:

AWSDeadlineCloud-FleetWorker

Questa politica gestita fornisce le autorizzazioni per:

- `deadline:AssumeFleetRoleForWorker`- Consente ai lavoratori di aggiornare le proprie credenziali.
- `deadline:UpdateWorker`- Consente ai lavoratori di aggiornare il proprio stato (ad esempio, su STOPED all'uscita).
- `deadline:UpdateWorkerSchedule`- Per ottenere lavoro e segnalare i progressi.
- `deadline:BatchGetJobEntity`- Per recuperare informazioni sul lavoro.
- `deadline:AssumeQueueRoleForWorker`- Per accedere alle credenziali del ruolo di coda durante l'esecuzione del lavoro.

Aggiungi le autorizzazioni KMS per le farm crittografate

Se la tua fattoria è stata creata utilizzando una chiave KMS, aggiungi queste autorizzazioni al ruolo del tuo parco macchine per garantire che il lavoratore possa accedere ai dati crittografati presenti nella fattoria.

Le autorizzazioni KMS sono necessarie solo se alla fattoria è associata una chiave KMS. La `kms:ViaService` condizione deve utilizzare il formato. `deadline.{region}.amazonaws.com`

Quando si crea una flotta, viene creato un gruppo di log di CloudWatch Logs per tale flotta. Le autorizzazioni del lavoratore vengono utilizzate dal servizio Deadline Cloud per creare un flusso di log specifico per quel particolare lavoratore. Dopo la configurazione e l'esecuzione, il lavoratore utilizzerà queste autorizzazioni per inviare gli eventi di registro direttamente ai registri. CloudWatch

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
```

```
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

Modifica del ruolo della flotta

Le autorizzazioni per il ruolo della flotta non sono personalizzabili. Le autorizzazioni descritte sono sempre necessarie e l'aggiunta di autorizzazioni aggiuntive non ha alcun effetto.

Customer-managed ruolo di fleet host

Configura un WorkerHost ruolo se utilizzi flotte gestite dal cliente su istanze Amazon EC2 o host locali.

Cosa fa il ruolo WorkerHost

Il WorkerHost ruolo coinvolge i dipendenti degli host di flotte gestite dai clienti. Fornisce le autorizzazioni minime necessarie a un host per:

- Crea un lavoratore in Deadline Cloud
- Assumi il ruolo della flotta per recuperare le credenziali operative
- Tagga i lavoratori con i tag della flotta (se la propagazione dei tag è abilitata)

Imposta le autorizzazioni dei WorkerHost ruoli

Allega la seguente politica AWS gestita al tuo WorkerHost ruolo:

AWSDeadlineCloud-WorkerHost

Questa politica gestita fornisce le autorizzazioni per:

- `deadline:CreateWorker`- Consente all'host di registrare un nuovo lavoratore.
- `deadline:AssumeFleetRoleForWorker`- Consente all'host di assumere il ruolo della flotta.
- `deadline:TagResource`- Consente di taggare i lavoratori durante la creazione (se abilitata).
- `deadline:ListTagsForResource`- Consente la lettura dei tag della flotta per la propagazione.

Comprendi il processo di bootstrap

Il WorkerHost ruolo viene utilizzato solo durante l'avvio iniziale del lavoratore:

1. L'agente di lavoro si avvia sull'host utilizzando WorkerHost le credenziali.
2. Richiama la registrazione con `deadline:CreateWorker` Deadline Cloud.
3. Quindi richiama per recuperare le credenziali del ruolo della `deadline:AssumeFleetRoleForWorker` flotta.
4. Da questo momento in poi, il lavoratore utilizza solo le credenziali del ruolo della flotta per tutte le operazioni.

Il WorkerHost ruolo non viene utilizzato dopo che il lavoratore ha iniziato a correre. Questa politica non è obbligatoria per le Service-managed flotte. Nelle Service-managed flotte, il bootstrap viene eseguito automaticamente.

Ruolo di coda

Il ruolo di coda viene assunto dal lavoratore durante l'elaborazione di un'operazione. Questo ruolo fornisce le autorizzazioni necessarie per completare l'attività.

Quando si creano o si aggiornano le code a livello di programmazione, specificare l'ARN del ruolo di coda utilizzando le operazioni o API. `CreateQueue` `UpdateQueue`

Imposta la politica di attendibilità dei ruoli di coda

Il tuo ruolo in coda deve fidarsi del servizio Deadline Cloud.

Come best practice, la politica di fiducia dovrebbe includere le condizioni di sicurezza per la protezione di Confused Deputy. Per saperne di più sulla protezione di Confused Deputy, consulta Confused Deputy nella Guida per l'utente di Deadline Cloud.

- `aws:SourceAccount`assicura che solo le risorse dello stesso Account AWS possano assumere questo ruolo.
- `aws:SourceArn`limita l'assunzione del ruolo a una specifica Deadline Cloud farm.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

Comprendi le autorizzazioni dei ruoli in coda

Il ruolo di coda non utilizza un'unica policy gestita. Invece, quando configuri la coda nella console, Deadline Cloud crea una politica personalizzata per la coda in base alla tua configurazione.

Questa politica creata automaticamente fornisce l'accesso a:

Allegati Job

Accesso in lettura e scrittura al bucket Amazon S3 specificato per i file di input e output dei job:

```
{
  "Effect": "Allow",
```

```
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
    }
  }
}
```

Job log

Leggi l'accesso ai CloudWatch registri dei lavori in questa coda. Ogni coda ha il proprio gruppo di log e ogni sessione ha il proprio flusso di log:

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Third-party software

Accesso al download di software di terze parti supportato da Deadline Cloud (come Maya, Blender e altri):

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
```

```
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Aggiungi le autorizzazioni per i tuoi lavori

Aggiungi le autorizzazioni al tuo ruolo in coda affinché i tuoi lavori Servizi AWS abbiano bisogno di accedere. Quando scrivi gli script delle OpenJobDescription fasi, l'SDK AWS CLI e l'SDK utilizzeranno automaticamente le credenziali del tuo ruolo di coda. Utilizzatelo per accedere ai servizi aggiuntivi necessari per completare il lavoro.

Ecco alcuni esempi di casi d'uso:

- per recuperare dati personalizzati
- Autorizzazioni SSM per il tunneling verso un server di licenze personalizzato
- CloudWatch per l'emissione di metriche personalizzate
- Autorizzazione Deadline Cloud a creare nuovi lavori per flussi di lavoro dinamici

Come vengono utilizzate le credenziali del ruolo di coda

Deadline Cloud fornisce le credenziali del ruolo di coda per:

- Lavoratori durante l'esecuzione del lavoro
- Utenti tramite CLI e monitoraggio di Deadline Cloud quando interagiscono con gli allegati e i registri dei lavori

Deadline Cloud crea gruppi di CloudWatch log Logs separati per ogni coda. I job utilizzano le credenziali del ruolo di coda per scrivere i log nel gruppo di log della propria coda. La CLI e il monitor di Deadline Cloud utilizzano il ruolo di coda (`deadline:AssumeQueueRoleForRead`through) per leggere i log dei lavori dal gruppo di log della coda. La CLI e il monitor di Deadline Cloud utilizzano il ruolo di coda (`deadline:AssumeQueueRoleForUser`through) per caricare o scaricare i dati degli allegati del lavoro.

Ruolo di monitoraggio

Configura un ruolo di monitoraggio per consentire alle applicazioni web e desktop di monitoraggio di Deadline Cloud di accedere alle tue risorse Deadline Cloud.

Quando crei o aggiorni i monitor a livello di codice, specifica l'ARN del ruolo di monitoraggio utilizzando le operazioni o API. `CreateMonitor` `UpdateMonitor`

Cosa fa il ruolo del monitor

Il ruolo di monitoraggio consente a Deadline Cloud Monitor di fornire agli utenti finali l'accesso a:

- Funzionalità di base richieste per Deadline Cloud Integrated Submitters, CLI e monitor
- Funzionalità personalizzate per gli utenti finali

Imposta la politica di fiducia del ruolo di monitoraggio

Il tuo ruolo di monitor deve fidarsi del servizio Deadline Cloud.

Come best practice, la politica di fiducia dovrebbe includere le condizioni di sicurezza per la protezione di Confused Deputy. Per saperne di più sulla protezione di Confused Deputy, consulta Confused Deputy nella Guida per l'utente di Deadline Cloud.

`aws:SourceAccount`assicura che solo le risorse dello stesso Account AWS possano assumere questo ruolo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
```

```
  ]
}
```

Allega le autorizzazioni del ruolo di monitoraggio

Associa tutte le seguenti politiche AWS gestite al tuo ruolo di monitor per le operazioni di base:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

Come funziona il ruolo di monitoraggio

Quando si utilizza il monitor Deadline Cloud, un utente del servizio accede utilizzando AWS IAM Identity Center (IAM Identity Center) e viene assunto il ruolo di monitor. Le credenziali del ruolo assunto vengono utilizzate dall'applicazione di monitoraggio per visualizzare l'interfaccia utente del monitor, incluso l'elenco di fattorie, flotte, code e altre informazioni.

Quando si utilizza l'applicazione desktop di monitoraggio Deadline Cloud, queste credenziali vengono inoltre rese disponibili sulla workstation utilizzando un profilo di AWS credenziali denominato corrispondente al nome del profilo fornito dall'utente finale. Scopri di più sui profili denominati nella guida di riferimento AWS SDK and Tools.

Questo profilo denominato è il modo in cui la CLI di Deadline e i mittenti accedono alle risorse di Deadline Cloud.

Personalizzazione del ruolo del monitor per casi d'uso avanzati

È possibile personalizzare il ruolo di monitoraggio per modificare ciò che gli utenti possono fare a ciascun livello di accesso (visualizzatore, collaboratore, responsabile, proprietario) o per aggiungere autorizzazioni per flussi di lavoro avanzati.

Personalizzazione delle autorizzazioni a livello di accesso

Le quattro policy AWS gestite allegate al ruolo di monitoraggio controllano cosa può fare ogni livello di accesso. È possibile aggiungere politiche personalizzate al ruolo di monitoraggio per concedere o limitare le autorizzazioni per livelli di accesso specifici utilizzando la chiave di `deadline:MembershipLevel` condizione.

Ad esempio, per consentire ai collaboratori di aggiornare e annullare i lavori (che di solito sono limitati a Manager e Proprietari), aggiungi una politica come la seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Con questa politica, i collaboratori possono aggiornare e annullare le offerte di lavoro oltre a inviarle.

Aggiungere autorizzazioni per flussi di lavoro avanzati

Puoi aggiungere policy IAM personalizzate al ruolo di monitoraggio per concedere autorizzazioni aggiuntive a tutti gli utenti del monitor. Ciò è utile per flussi di lavoro di scripting avanzati in cui gli utenti devono accedere a funzionalità Servizi AWS oltre a quelle standard di Deadline Cloud.

Segui queste linee guida quando modifichi il tuo ruolo di monitor:

- Non rimuovere nessuna delle politiche gestite. La rimozione di queste politiche interrompe la funzionalità del monitor.

In che modo Deadline Cloud Monitor utilizza le credenziali del ruolo di monitoraggio

Deadline Cloud monitor ottiene automaticamente le credenziali del ruolo di monitoraggio al momento dell'autenticazione. Questa funzionalità consente all'applicazione desktop di fornire funzionalità di monitoraggio avanzate oltre a quelle disponibili in un browser Web standard.

Quando accedi con Deadline Cloud Monitor, crea automaticamente un profilo che puoi utilizzare con AWS CLI o con qualsiasi altro AWS strumento. Questo profilo utilizza le credenziali del ruolo di

monitoraggio, offrendoti l'accesso programmatico in Servizi AWS base alle autorizzazioni del ruolo di monitor.

I mittenti di Deadline Cloud funzionano allo stesso modo: utilizzano il profilo creato da Deadline Cloud Monitor per accedere con le autorizzazioni di ruolo appropriate. Servizi AWS

Personalizzazione avanzata dei ruoli di Deadline Cloud

Puoi estendere i ruoli di Deadline Cloud con autorizzazioni aggiuntive per abilitare casi d'uso avanzati oltre ai flussi di lavoro di rendering di base. Questo approccio sfrutta il sistema di gestione degli accessi di Deadline Cloud per controllare l'accesso ad altri Servizi AWS utenti in base all'appartenenza alla coda.

Collaborazione in team con AWS CodeCommit

Aggiungi AWS CodeCommit le autorizzazioni al tuo ruolo Queue per consentire la collaborazione in team sugli archivi dei progetti. Questo approccio utilizza il sistema di gestione degli accessi di Deadline Cloud per casi d'uso aggiuntivi: solo gli utenti con accesso alla coda specifica riceveranno queste AWS CodeCommit autorizzazioni, consentendoti di gestire l'accesso al repository per progetto tramite l'iscrizione alla coda di Deadline Cloud.

Questo è utile negli scenari in cui gli artisti devono accedere a risorse, script o file di configurazione specifici del progetto archiviati negli archivi come parte del loro flusso di lavoro di rendering. AWS CodeCommit

Add (Aggiungi) AWS CodeCommit autorizzazioni per il ruolo di coda

Aggiungi le seguenti autorizzazioni al tuo ruolo di coda per abilitare l'accesso: AWS CodeCommit

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Configura il fornitore di credenziali sulle postazioni di lavoro degli artisti

Configura ogni postazione di lavoro dell'artista per utilizzare le credenziali di coda di Deadline Cloud per l'accesso. AWS CodeCommit Questa configurazione viene eseguita una volta per workstation.

Per configurare il fornitore di credenziali

1. Aggiungi un profilo di fornitore di credenziali al tuo file di AWS configurazione (): `~/.aws/config`

```
[profile queue-codecommit]
credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

2. Configura Git per utilizzare questo profilo per i AWS CodeCommit repository:

```
git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
 '!aws codecommit credential-helper --profile queue-codecommit $@'
git config --global credential.https://git-
codecommit.REGION.amazonaws.com.UseHttpPath true
```

Sostituisci *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* e *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* con gli ID effettivi della fattoria e della coda. Sostituiscilo *REGION* con la tua AWS regione (ad esempio,`us-west-2`).

Utilizzo AWS CodeCommit con credenziali di coda

Una volta configurate, le operazioni Git utilizzeranno automaticamente le credenziali del ruolo di coda quando accedono ai AWS CodeCommit repository. Il `deadline queue export-credentials` comando restituisce credenziali temporanee simili alle seguenti:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Queste credenziali vengono aggiornate automaticamente secondo necessità e le operazioni Git funzioneranno perfettamente:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Gli artisti possono ora accedere agli archivi dei progetti utilizzando le loro autorizzazioni di coda senza bisogno di credenziali separate. AWS CodeCommit Solo gli utenti con accesso alla coda specifica potranno accedere al repository associato, abilitando un controllo granulare degli accessi tramite il sistema di iscrizione alla coda di Deadline Cloud.

Risoluzione dei problemi AWS Identità e accesso a Deadline Cloud

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Deadline Cloud e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in Deadline Cloud
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire l'accesso a persone esterne al mio Account AWS per accedere alle mie risorse Deadline Cloud

Non sono autorizzato a eseguire un'azione in Deadline Cloud

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `deadline:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `deadline:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo a Deadline Cloud.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Deadline Cloud. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire l'accesso a persone esterne al mio Account AWS per accedere alle mie risorse Deadline Cloud

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo degli accessi (ACL), utilizzare tali policy per concedere alle persone l'accesso alle proprie risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Deadline Cloud supporta queste funzionalità, consulta. Come funziona Deadline Cloud con IAM

- Per scoprire come fornire l'accesso alle tue risorse su tutto Account AWS ciò che possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Convalida della conformità per Deadline Cloud

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta la Documentazione AWS sulla sicurezza.

Resilienza in Deadline Cloud

L'infrastruttura AWS globale è costruita attorno a Regioni AWS zone di disponibilità. Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità è possibile progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni sulle zone di disponibilità, vedere Global Regioni AWS Infrastructure.AWS

AWS Deadline Cloud non esegue il backup dei dati memorizzati nel bucket S3 degli allegati di lavoro. Puoi abilitare i backup dei dati dei tuoi allegati di lavoro utilizzando qualsiasi meccanismo di backup standard di Amazon S3, come S3 Versioning o. AWS Backup

Sicurezza dell'infrastruttura in Deadline Cloud

In quanto servizio gestito, AWS Deadline Cloud è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere a Deadline Cloud attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di crittografia con Perfect Forward Secrecy (PFS) come DHE (Ephemeral) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Deadline Cloud non supporta l'utilizzo di policy per gli endpoint del cloud privato AWS PrivateLink virtuale (VPC). Utilizza la politica AWS PrivateLink predefinita, che garantisce l'accesso completo all'endpoint. Per ulteriori informazioni, consulta la policy predefinita per gli endpoint nella guida per l'AWS PrivateLink utente.

Configurazione e analisi delle vulnerabilità in Deadline Cloud

AWS gestisce le attività di sicurezza di base come l'applicazione di patch al sistema operativo guest (OS) e al database, la configurazione del firewall e il disaster recovery. Queste procedure sono state riviste e certificate dalle terze parti appropriate. Per ulteriori dettagli, consulta le seguenti risorse :

- Modello di responsabilità condivisa
- Amazon Web Services: panoramica dei processi di sicurezza (whitepaper)

AWS Deadline Cloud gestisce le attività su flotte gestite dai servizi o dai clienti:

- Per le flotte gestite dai servizi, Deadline Cloud gestisce il sistema operativo ospite.

- Per le flotte gestite dai clienti, sei responsabile della gestione del sistema operativo.

Per ulteriori informazioni sulla configurazione e l'analisi delle vulnerabilità per AWS Deadline Cloud, consulta

- Le migliori pratiche di sicurezza per Deadline Cloud

Cross-service confusa prevenzione sostitutiva

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Nel AWS, l'impersonificazione intersettoriale può portare al confuso problema del vicesceriffo. Cross-service l'impersonificazione può verificarsi quando un servizio (il servizio chiamante) chiama un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account.

Si consiglia di utilizzare aws:SourceArnle chiavi di contesto della condizione aws:SourceAccountglobale nelle politiche delle risorse per limitare le autorizzazioni che AWS Deadline Cloud forniscono un altro servizio alla risorsa. Utilizzare `aws:SourceArn` se si desidera consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'Amazon Resource Name (ARN) completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (`*`) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:deadline:*:123456789012:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni.

L'esempio seguente mostra come utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition Deadline Cloud per evitare il confuso problema del vice.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

Accesso AWS Deadline Cloud utilizzando un endpoint di interfaccia (AWS PrivateLink)

Puoi usarlo AWS PrivateLink per creare una connessione privata tra il tuo VPC e. AWS Deadline Cloud Puoi accedere Deadline Cloud come se fosse nel tuo VPC, senza l'uso di un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. Direct Connect Le istanze del tuo VPC non necessitano di indirizzi IP pubblici per accedervi. Deadline Cloud

Stabilisci questa connessione privata creando un endpoint di interfaccia attivato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Queste sono interfacce di rete gestite dal richiedente che fungono da punto di ingresso per il traffico destinato a Deadline Cloud.

Deadline Cloud dispone anche di endpoint dual-stack. Dual-stack gli endpoint supportano le richieste su IPv6 e IPv4.

Per ulteriori informazioni, consulta la sezione Accesso a Servizi AWS tramite AWS PrivateLink nella Guida di AWS PrivateLink .

Considerazioni per Deadline Cloud

Prima di configurare un endpoint di interfaccia per Deadline Cloud, consulta Accedere a un servizio AWS utilizzando un endpoint VPC di interfaccia nella Guida.AWS PrivateLink

Deadline Cloud supporta l'esecuzione di chiamate a tutte le sue azioni API tramite l'endpoint dell'interfaccia.

Per impostazione predefinita, l'accesso completo a Deadline Cloud è consentito tramite l'endpoint dell'interfaccia. In alternativa, è possibile associare un gruppo di sicurezza alle interfacce di rete dell'endpoint per controllare il traffico che Deadline Cloud attraversa l'endpoint dell'interfaccia.

Deadline Cloud supporta anche le policy degli endpoint VPC. Per ulteriori informazioni, consulta Control access to VPC endpoints using endpoint policies nella Guida di AWS PrivateLink .

Deadline Cloud endpoint

Deadline Cloud utilizza quattro endpoint per l'accesso al servizio utilizzando AWS PrivateLink : due per IPv4 e due per IPv6.

I lavoratori utilizzano l'`scheduling.deadline.`*`region`*`.amazonaws.com`endpoint per prelevare le attività dalla coda, segnalarne lo stato di avanzamento e rispedirne l' Deadline Cloud output. Se si utilizza una flotta gestita dal cliente, l'endpoint di pianificazione è l'unico endpoint da creare, a meno che non si utilizzino operazioni di gestione. Ad esempio, se un job crea più lavori, è necessario abilitare l'endpoint di gestione a richiamare l'operazione. `CreateJob`

Il Deadline Cloud monitor utilizza il `management.deadline.`*`region`*`.amazonaws.com` per gestire le risorse della fattoria, ad esempio per creare e modificare code e flotte o ottenere elenchi di lavori, fasi e attività.

Gli AWS SDK e la CLI aggiungono automaticamente i `management` prefissi `scheduling` and all'endpoint. Se desideri disabilitare questo comportamento, consulta la sezione Host Prefix Injection nella Guida di riferimento agli SDK and Tools.AWS

Deadline Cloud richiede anche endpoint per i seguenti endpoint di servizio: AWS

- Se configuri la tua flotta gestita dai clienti in una sottorete senza connessione Internet, devi creare un endpoint VPC per CloudWatch Amazon Logs in modo che gli operatori possano scrivere i log. Per ulteriori informazioni, consulta Monitoraggio con. CloudWatch
- Se utilizzi gli allegati di lavoro, devi creare un endpoint VPC per Amazon Simple Storage Service (Amazon S3) Let's Amazon S3) in modo che i lavoratori possano accedere agli allegati. Per ulteriori informazioni, vedere Job attachments in Deadline Cloud.

Crea endpoint per Deadline Cloud

Puoi creare endpoint di interfaccia per Deadline Cloud utilizzare la console Amazon VPC o AWS Command Line Interface ().AWS CLI Per ulteriori informazioni, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di AWS PrivateLink .

Crea endpoint di gestione e pianificazione per l' Deadline Cloud utilizzo dei seguenti nomi di servizio. *region*Sostituiscilo con quello Regione AWS che hai distribuito. Deadline Cloud

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud supporta endpoint dual-stack.

Se abiliti il DNS privato per gli endpoint dell'interfaccia, puoi effettuare richieste API Deadline Cloud utilizzando il nome DNS regionale predefinito. Ad esempio, `scheduling.deadline.us-east-1.amazonaws.com` per le operazioni dei lavoratori o `management.deadline.us-east-1.amazonaws.com` per tutte le altre operazioni.

Se la flotta gestita dal cliente si trova su una sottorete senza una connessione Internet, è necessario creare un endpoint CloudWatch Logs utilizzando il seguente nome di servizio:

```
com.amazonaws.region.logs
```

Se utilizzi gli allegati di lavoro per trasferire file, devi creare un endpoint Amazon S3 utilizzando il seguente nome di servizio:

```
com.amazonaws.region.s3
```

Ambienti di rete con restrizioni

Deadline Cloud fornisce strumenti che vengono utilizzati dagli artisti o da altri utenti sulle loro postazioni di lavoro locali. Questi strumenti richiedono l'accesso all' AWS API e agli endpoint web per svolgere la loro funzione. Se si filtra l'accesso a AWS domini o endpoint URL specifici utilizzando una soluzione di filtraggio dei contenuti Web come i firewall di nuova generazione (NGFW) o i Secure Web Gateways (SWG), è necessario aggiungere i seguenti domini o endpoint URL agli elenchi consentiti della soluzione di filtraggio dei contenuti Web.

AWS Endpoint API da inserire nella lista consentita

Gli strumenti client di Deadline Cloud, come monitor Console di gestione AWS, CLI e submitters integrati, richiedono l'accesso alle API oltre AWS a Deadline Cloud. Questi endpoint supportano solo IPv4.

- `scheduling.deadline.[Region].amazonaws.com`
- `management.deadline.[Region].amazonaws.com`
- `logs.[Region].amazonaws.com`
- `ec2.[Region].amazonaws.com`
- `s3.[Region].amazonaws.com`
- `sts.[Region].amazonaws.com`
- `identitystore.[Region].amazonaws.com`

Domini Web da inserire nella lista consentita

Il monitor Deadline Cloud richiede l'accesso ai seguenti domini per funzionare.

Per ulteriori informazioni sull'elenco dei domini consentiti per AWS Sign-In, consulta Domini da aggiungere all'elenco dei domini consentiti nella Guida per l'utente.AWS Sign-In

- `downloads.deadlinecloud.amazonaws.com`
- `d2ev1rdnjzhmnr.cloudfront.net`
- `prod.log.shortbread.aws.dev`
- `prod.tools.shortbread.aws.dev`
- `prod.log.shortbread.analytics.console.aws.a2z.com`

- prod.tools.shortbread.analytics.console.aws.a2z.com
- global.help-panel.docs.aws.a2z.com
- *[Region]*.signin.aws
- *[Region]*.signin.aws.amazon.com
- sso.*[Region]*.amazonaws.com
- portal.sso.*[Region]*.amazonaws.com
- oidc.*[Region]*.amazonaws.com
- assets.sso-portal.*[Region]*.amazonaws.com

Environment-specific endpoint da inserire nella lista consentita

Questi domini variano a seconda della configurazione specifica di Deadline Cloud. Se vengono creati monitor o code aggiuntivi di Deadline Cloud, sarà necessario inserire altri domini nell'elenco consentito.

- *[Directory ID or alias]*.awsapps.com

  Questo dominio è legato alla configurazione di IAM Identity Center e dovrebbe essere lo stesso per tutte le configurazioni che utilizzano la stessa istanza di IAM Identity Center. Il valore esatto può essere trovato dall'amministratore aziendale nella console IAM Identity Center in Impostazioni → Portale di accesso AWS URL.
- *[Monitor alias]*.*[Region]*.deadlinecloud.amazonaws.com

  Questo dominio serve per la configurazione di Monitor in Deadline Cloud. Gli artisti inseriscono questo link nel loro browser o nell'applicazione di monitoraggio Deadline Cloud. Se Deadline Cloud verrà configurato in account o regioni aggiuntivi in futuro, questo dominio cambierà. Puoi trovare questo valore nella console Deadline Cloud in Dashboard → Panoramica del monitor → Dettagli del monitor → URL.
- *[Bucket name]*.*[Region]*.s3.amazonaws.com

  Questo è il dominio per il bucket degli allegati di lavoro utilizzato dalle code di Deadline Cloud. Ogni coda può avere il proprio bucket di allegati di lavoro configurato. Il nome esatto del bucket è disponibile nella console Deadline Cloud in Queues → Queue details → Job attachments. Per ulteriori informazioni sugli allegati di lavoro, consulta la documentazione sulle code.

Le migliori pratiche di sicurezza per Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) offre una serie di funzionalità di sicurezza da considerare durante lo sviluppo e l'implementazione delle proprie politiche di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l'ambiente, sono da considerare come considerazioni utili anziché prescrizioni.

> **Note**
>
> Per ulteriori informazioni sull'importanza di molti argomenti relativi alla sicurezza, consulta il Modello di responsabilità condivisa.

Protezione dei dati

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e configurare account individuali con AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza servizi di sicurezza gestiti avanzati come Amazon Macie, che aiuta a scoprire e proteggere i dati personali archiviati in Amazon Simple Storage Service (Amazon S3).
- Se necessiti di moduli crittografici convalidati FIPS 140-2 quando accedi ad AWS attraverso un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Consigliamo di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero come un campo Nome. Questa raccomandazione include quando lavori con AWS Deadline Cloud o altro Servizi AWS utilizzando la console, l'API o gli SDK. AWS

CLI AWS Tutti i dati che inserisci in Deadline Cloud o in altri servizi potrebbero essere raccolti per essere inclusi nei registri di diagnostica. Quando fornisci un URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

AWS Identity and Access Management autorizzazioni

Gestisci l'accesso alle AWS risorse utilizzando gli utenti, i ruoli AWS Identity and Access Management (IAM) e concedendo il minimo privilegio agli utenti. Stabilisci politiche e procedure di gestione delle credenziali per creare, distribuire, ruotare e revocare le credenziali di accesso. AWS Per ulteriori informazioni, consulta Best practice IAM nella Guida per l'utente di IAM.

Esegui lavori come utenti e gruppi

Quando si utilizza la funzionalità di coda in Deadline Cloud, è consigliabile specificare un utente del sistema operativo (OS) e il relativo gruppo primario in modo che l'utente del sistema operativo disponga delle autorizzazioni con privilegi minimi per i lavori della coda.

Quando specifichi un «Esegui come utente» (e gruppo), tutti i processi per i lavori inviati alla coda verranno eseguiti utilizzando quell'utente del sistema operativo e erediteranno le autorizzazioni del sistema operativo associate a quell'utente.

Le configurazioni della flotta e della coda si combinano per stabilire un livello di sicurezza. Sul lato della coda, è possibile specificare il ruolo «Job run as user» e IAM per utilizzare il sistema operativo e AWS le autorizzazioni per i lavori della coda. La flotta definisce l'infrastruttura (worker host, reti, storage condiviso montato) che, se associata a una particolare coda, esegue i lavori all'interno della coda. I job di una o più code associate devono accedere ai dati disponibili sugli host dei worker. Specificare un utente o un gruppo aiuta a proteggere i dati nei lavori da altre code, da altri software installati o da altri utenti con accesso agli host di lavoro. Quando una coda è priva di un utente, viene eseguita come utente agente che può impersonare () `sudo` qualsiasi utente della coda. In questo modo, una coda senza utente può trasferire i privilegi a un'altra coda.

Rete

Per evitare che il traffico venga intercettato o reindirizzato, è essenziale proteggere come e dove viene instradato il traffico di rete.

Ti consigliamo di proteggere il tuo ambiente di rete nei seguenti modi:

- Proteggi le tabelle di routing delle sottoreti di Amazon Virtual Private Cloud (Amazon VPC) per controllare come viene instradato il traffico a livello IP.

- Se utilizzi Amazon Route 53 (Route 53) come provider DNS nella configurazione della tua farm o workstation, accedi in modo sicuro all'API Route 53.
- Se ti connetti a Deadline Cloud all'esterno, AWS ad esempio utilizzando workstation locali o altri data center, proteggi qualsiasi infrastruttura di rete locale. Ciò include server DNS e tabelle di routing su router, switch e altri dispositivi di rete.

Lavori e dati sui lavori

I job di Deadline Cloud vengono eseguiti all'interno delle sessioni sugli host dei lavoratori. Ogni sessione esegue uno o più processi sull'host di lavoro, che in genere richiedono l'immissione di dati per produrre l'output.

Per proteggere questi dati, è possibile configurare gli utenti del sistema operativo con code. L'agente di lavoro utilizza l'utente del sistema operativo in coda per eseguire i sottoprocessi della sessione. Questi sottoprocessi ereditano le autorizzazioni dell'utente del sistema operativo di coda.

Ti consigliamo di seguire le migliori pratiche per proteggere l'accesso ai dati a cui accedono questi sottoprocessi. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

Struttura dell'azienda

Puoi organizzare le flotte e le code di Deadline Cloud in molti modi. Tuttavia, alcune disposizioni hanno implicazioni in termini di sicurezza.

Una farm ha uno dei confini più sicuri perché non può condividere le risorse di Deadline Cloud con altre aziende agricole, tra cui flotte, code e profili di archiviazione. Tuttavia, puoi condividere AWS risorse esterne all'interno di una farm, il che compromette i limiti di sicurezza.

È inoltre possibile stabilire limiti di sicurezza tra le code all'interno della stessa farm utilizzando la configurazione appropriata.

Segui queste best practice per creare code sicure nella stessa farm:

- Associa una flotta solo alle code all'interno dello stesso limite di sicurezza. Tenere presente quanto segue:
  - Dopo l'esecuzione del processo sull'host del lavoratore, i dati potrebbero rimanere indietro, ad esempio in una directory temporanea o nella home directory dell'utente in coda.
  - Lo stesso utente del sistema operativo esegue tutti i lavori su un host Fleet Worker di proprietà del servizio, indipendentemente dalla coda a cui viene inviato il lavoro.

  - Un job può lasciare i processi in esecuzione su un worker host, permettendo ai job di altre code di osservare altri processi in esecuzione.
- Assicurati che solo le code all'interno dello stesso limite di sicurezza condividano un bucket Amazon S3 per gli allegati dei lavori.
- Assicurati che solo le code all'interno dello stesso limite di sicurezza condividano un utente del sistema operativo.
- Proteggi tutte AWS le altre risorse integrate nella farm fino al limite.

Code di allegati Job

Gli allegati Job sono associati a una coda, che utilizza il tuo bucket Amazon S3.

- Gli allegati di lavoro scrivono e leggono da un prefisso root nel bucket Amazon S3. È necessario specificare questo prefisso root nella chiamata API. `CreateQueue`
- Il bucket ha un corrispondente`Queue Role`, che specifica il ruolo che concede agli utenti della coda l'accesso al bucket e al prefisso root. Quando crei una coda, specifichi l'`Queue Role`Amazon Resource Name (ARN) insieme al bucket degli allegati del lavoro e al prefisso root.
- Le chiamate autorizzate a `AssumeQueueRoleForReadAssumeQueueRoleForUser`, e le operazioni `AssumeQueueRoleForWorker` API restituiscono una serie di credenziali di sicurezza temporanee per. `Queue Role`

Se crei una coda e riutilizzi un bucket Amazon S3 e un prefisso root, c'è il rischio che le informazioni vengano divulgate a parti non autorizzate. Ad esempio, QueueA e QueueB condividono lo stesso bucket e lo stesso prefisso root. In un flusso di lavoro sicuro, Artista ha accesso a QueueA ma non a QueueB. Tuttavia, quando più code condividono un bucket, Artista può accedere ai dati nei dati di QueueB perché utilizza lo stesso bucket e lo stesso prefisso root di QueueA.

La console imposta code sicure per impostazione predefinita. Assicurati che le code abbiano una combinazione distinta di bucket Amazon S3 e prefisso root, a meno che non facciano parte di un limite di sicurezza comune.

Per isolare le code, devi configurare per consentire l'accesso alla coda solo `Queue Role` al bucket e al prefisso root. Nell'esempio seguente, sostituisci ciascuno *`placeholder`* di essi con le informazioni specifiche della risorsa.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

È inoltre necessario impostare una politica di fiducia per il ruolo. Nell'esempio seguente, sostituisci il *placeholder* testo con le informazioni specifiche della risorsa.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
```

```
}
```

Bucket Amazon S3 software personalizzati

Puoi aggiungere la seguente dichiarazione alla tua richiesta di accesso `Queue Role` al software personalizzato nel tuo bucket Amazon S3. Nell'esempio seguente, sostituiscilo *SOFTWARE_BUCKET_NAME* con il nome del bucket S3 e *BUCKET_ACCOUNT_OWNER* con l' Account AWS ID proprietario del bucket.

```
"Statement": [
    {
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

Per ulteriori informazioni sulle best practice di sicurezza di Amazon S3, consulta le best practice di sicurezza per Amazon S3 nella Amazon Simple Storage Service User Guide.

Operatori ospitanti

Proteggi gli host per i lavoratori per garantire che ogni utente possa eseguire operazioni solo per il ruolo assegnato.

Consigliamo le seguenti best practice per proteggere gli host dei lavoratori:

- L'utilizzo di uno script di configurazione dell'host può modificare la sicurezza e le operazioni di un lavoratore. Una configurazione errata può causare l'instabilità o l'interruzione del lavoro del lavoratore. È responsabilità dell'utente eseguire il debug di tali errori.
- Non utilizzare lo stesso `jobRunAsUser` valore con più code a meno che i lavori inviati a tali code non rientrino nello stesso limite di sicurezza.
- Non impostate la coda `jobRunAsUser` sul nome dell'utente del sistema operativo con cui viene eseguito il worker agent.
- Concedi agli utenti della coda le autorizzazioni del sistema operativo con i privilegi minimi necessarie per i carichi di lavoro in coda previsti. Assicurati che non dispongano delle autorizzazioni di scrittura del filesystem per i file di programma Work Agent o altro software condiviso.
- Assicurati che solo l'utente root Linux e il suo account siano `Administrator` proprietari e che possano modificare Windows i file di programma del worker agent.
- Sugli host Linux worker, valuta la possibilità di configurare un `umask` override `/etc/sudoers` che consenta all'utente worker agent di avviare i processi come utenti in coda. Questa configurazione aiuta a garantire che altri utenti non possano accedere ai file scritti nella coda.
- Concedi a persone fidate l'accesso con i privilegi minimi agli host dei lavoratori.
- Limita le autorizzazioni al DNS locale, sostituisci i file di configurazione (`/etc/hosts`attivati e attivatiWindows) Linux e instrada le tabelle `C:\Windows\system32\etc\hosts` sulle workstation e sui sistemi operativi degli host di lavoro.
- Limita le autorizzazioni alla configurazione DNS sulle workstation e sui sistemi operativi degli host di lavoro.
- Aggiorna regolarmente il sistema operativo e tutto il software installato. Questo approccio include software utilizzati specificamente con Deadline Cloud, come mittenti, adattatori, agenti di lavoro, OpenJD pacchetti e altro.
- Usa password complesse per la coda. Windows `jobRunAsUser`
- Ruota regolarmente le password per la coda. `jobRunAsUser`
- Garantisci l'accesso con il minimo privilegio alle Windows password segrete ed elimina quelle inutilizzate.
- Non `jobRunAsUser` autorizzate la coda a eseguire i comandi di pianificazione in futuro:
  - SìLinux, nega a questi account l'accesso a `cron` e. `at`
  - SìWindows, nega a questi account l'accesso al Windows task scheduler.

> **Note**
>
> Per ulteriori informazioni sull'importanza di applicare regolarmente patch al sistema operativo e al software installato, consulta il Modello di responsabilità condivisa.

Script di configurazione dell'host

- L'utilizzo di uno script di configurazione dell'host può modificare la sicurezza e le operazioni di un lavoratore. Una configurazione errata può causare l'instabilità o l'interruzione del lavoro del lavoratore. È responsabilità dell'utente eseguire il debug di tali errori.

Workstation

È importante proteggere le workstation con accesso a Deadline Cloud. Questo approccio aiuta a garantire che tutti i lavori che invii a Deadline Cloud non possano eseguire carichi di lavoro arbitrari fatturati a te. Account AWS

Consigliamo le seguenti best practice per proteggere le postazioni di lavoro degli artisti. Per ulteriori informazioni, consultare il Shared Responsibility Model (Modello di responsabilità condivisa).

- Proteggi tutte le credenziali permanenti che forniscono l'accesso a AWS, incluso Deadline Cloud. Per ulteriori informazioni, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente di IAM .
- Installa solo software affidabile e sicuro.
- Richiedi agli utenti di federarsi con un provider di identità per accedere AWS con credenziali temporanee.
- Utilizza autorizzazioni sicure sui file di programma del mittente di Deadline Cloud per impedirne la manomissione.
- Concedi alle persone fidate l'accesso meno privilegiato alle postazioni di lavoro degli artisti.
- Utilizza solo i mittenti e gli adattatori che ottieni tramite Deadline Cloud Monitor.
- Limita le autorizzazioni al DNS localemacOS, sostituisci i file di configurazione (attivati e `/etc/hosts` attivatiWindows) Linux e instrada le tabelle `C:\Windows\system32\etc\hosts` sulle workstation e sui sistemi operativi host dei lavoratori.
- Limita le autorizzazioni alle workstation e ai `/etc/resolve.conf` sistemi operativi host dei lavoratori.

- Aggiorna regolarmente il sistema operativo e tutto il software installato. Questo approccio include software utilizzati specificamente con Deadline Cloud, come mittenti, adattatori, agenti di lavoro, OpenJD pacchetti e altro.

Verifica l'autenticità del software scaricato

Verifica l'autenticità del software dopo aver scaricato il programma di installazione per proteggerlo dalla manomissione dei file. Questa procedura funziona per entrambi i sistemi. Windows Linux

Windows

Per verificare l'autenticità dei file scaricati, completa i seguenti passaggi.

1. Nel comando seguente, *file* sostituiscilo con il file che desideri verificare. Ad esempio, ***C:\PATH\TO\MY*DeadlineCloudSubmitter-windows-x64-installer.exe** . Inoltre, *signtool-sdk-version* sostituiscilo con la versione dell'SignToolSDK installata. Ad esempio, **10.0.22000.0**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Ad esempio, puoi verificare il file di installazione del submitter di Deadline Cloud eseguendo il seguente comando:

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-
   windows-x64-installer.exe
   ```

Linux

Per verificare l'autenticità dei file scaricati, utilizza lo strumento da riga di comando. gpg

1. Importa la OpenPGP chiave eseguendo il seguente comando:

   ```
    gpg --import --armor <<EOF
   -----BEGIN PGP PUBLIC KEY BLOCK-----

   mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
   MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
   BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
   ```

```
/shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. Determina se fidarti della OpenPGP chiave. Alcuni fattori da considerare quando si decide se considerare attendibile la chiave di cui sopra sono i seguenti:

   - La connessione Internet che hai utilizzato per ottenere la chiave GPG da questo sito Web è sicura.
   - Il dispositivo da cui accedi a questo sito Web è sicuro.
   - AWS ha adottato misure per proteggere l'hosting della chiave OpenPGP pubblica su questo sito web.

3. Se decidi di considerare attendibile la OpenPGP chiave, modifica la chiave in base all'attendibilità con un metodo gpg simile al seguente esempio:

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF

    gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.
```

```
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud example@example.com

   gpg> trust
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com

   Please decide how far you trust this user to correctly verify other users'
keys
   (by looking at passports, checking fingerprints from different sources,
etc.)

     1 = I don't know or won't say
     2 = I do NOT trust
     3 = I trust marginally
     4 = I trust fully
     5 = I trust ultimately
     m = back to the main menu

   Your decision? 5
   Do you really want to set this key to ultimate trust? (y/N) y

   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: ultimate      validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
   Please note that the shown key validity is not necessarily correct
   unless you restart the program.

   gpg> quit
```

4. Verifica il programma di installazione del mittente di Deadline Cloud

   Per verificare il programma di installazione di Deadline Cloud Submitter, completa i seguenti passaggi:

   a. Scarica il file di firma per il programma di installazione del mittente di Deadline Cloud.

      Scarica il file della firma (.sig)

b. Verifica la firma del programma di installazione del mittente di Deadline Cloud eseguendo:

```
gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
DeadlineCloudSubmitter-linux-x64-installer.run
```

5. Verifica il monitor Deadline Cloud

> **Note**
>
> Puoi verificare il download del monitor Deadline Cloud utilizzando file di firma o metodi specifici della piattaforma. Per i metodi specifici della piattaforma, consulta la Linux (Debian) scheda, la scheda Linux (RPM) o la Linux (AppImage) scheda in base al tipo di file scaricato.

Per verificare l'applicazione desktop Deadline Cloud Monitor con i file di firma, completa i seguenti passaggi:

a. Scarica il file di firma corrispondente per il tuo programma di installazione del monitor Deadline Cloud:

- Scarica il file di firma.deb
- Scarica il file di firma.rpm
- Scarica. AppImage file di firma

b. Verifica la firma:

Per .deb:

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

Per .rpm:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

Per. AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Verificate che l'output sia simile al seguente:

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Se l'output contiene la frase`Good signature from "AWS Deadline Cloud"`, significa che la firma è stata verificata con successo e che puoi eseguire lo script di installazione del monitor Deadline Cloud.

Chiavi storiche

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
```

```
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

Per verificare i pacchetti che utilizzano unLinux. AppImage binario, completa prima i passaggi 1-3 nella Linux scheda, quindi completa i passaggi seguenti.

1. Dalla AppImageUpdate pagina in poi GitHub, scarica il validate-x86_64. AppImagefile.
2. Dopo aver scaricato il file, per aggiungere i permessi di esecuzione, esegui il seguente comando.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```

3. Per aggiungere i permessi di esecuzione, esegui il comando seguente.

   ```
   chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

4. Per verificare la firma del monitor di Deadline Cloud, esegui il seguente comando.

   ```
   ./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

   Se l'output contiene la frase`Validation successful`, significa che la firma è stata verificata con successo e puoi eseguire in sicurezza lo script di installazione del monitor di Deadline Cloud.

Linux (Debian)

Per verificare i pacchetti che utilizzano un Linux file binario.deb, completate prima i passaggi 1-3 nella scheda. Linux

dpkg è lo strumento di gestione dei pacchetti principale nella maggior parte delle distribuzioni basate. debian Linux È possibile verificare il file.deb con lo strumento.

1. Scarica il file.deb del monitor Deadline Cloud:

   Scarica Deadline Cloud monitor (.deb)

2. Verifica il file.deb:

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. L'output sarà simile a:

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. Per verificare il file.deb, verificate che `GOODSIG` sia presente nell'output.

Linux (RPM)

Per verificare i pacchetti che utilizzano un Linux file binario .rpm, completate prima i passaggi 1-3 nella scheda. Linux

1. Scarica il file.rpm di Deadline Cloud monitor:

   Scarica Deadline Cloud monitor (.rpm)

2. Verifica il file.rpm:

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. L'output sarà simile a:

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. Per verificare il file.rpm, verificate che `digests signatures OK` sia presente nell'output.

Cronologia dei documenti

Per informazioni sugli aggiornamenti di AWS Deadline Cloud, consulta le note di rilascio di Deadline Cloud.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.