aws

Guía para desarrolladores

Nube de plazos

Nube de plazos: Guía para desarrolladores

Table of Contents

¿Qué es AWS Deadline Cloud?

AWS Deadline Cloud es un AWS servicio totalmente gestionado que le permite tener una granja de procesamiento escalable en funcionamiento en cuestión de minutos. Proporciona una consola de administración para gestionar los usuarios, las granjas, las colas para programar los trabajos y las flotas de trabajadores que se encargan del procesamiento.

Esta guía para desarrolladores está destinada a desarrolladores de procesos, herramientas y aplicaciones en una amplia gama de casos de uso, incluidos los siguientes:

- Los desarrolladores y directores técnicos de Pipeline pueden integrar Deadline Cloud APIs y sus funciones en sus procesos de producción personalizados.
- Los proveedores de software independientes pueden integrar Deadline Cloud en sus aplicaciones, lo que permite a los artistas y usuarios de creación de contenido digital enviar los trabajos de renderizado de Deadline Cloud sin problemas desde sus estaciones de trabajo.
- Los desarrolladores de servicios web y basados en la nube pueden integrar el renderizado de Deadline Cloud en sus plataformas, lo que permite a los clientes proporcionar recursos para ver los productos de forma virtual.

Proporcionamos herramientas que te permiten trabajar directamente en cualquier fase de tu proceso:

- Una interfaz de línea de comandos que puedes usar directamente o desde scripts.
- El AWS SDK para 11 lenguajes de programación populares.
- Una interfaz web basada en REST a la que puede llamar desde sus aplicaciones.

También puede utilizar otras Servicios de AWS en sus aplicaciones personalizadas. Por ejemplo, puede usar:

- AWS CloudFormationpara automatizar la creación y eliminación de granjas, colas y flotas.
- Amazon CloudWatch recopilará métricas para los puestos de trabajo.
- Amazon Simple Storage Service para almacenar y gestionar los activos digitales y la producción de trabajos.
- AWS IAM Identity Centerpara gestionar los usuarios y grupos de sus granjas.

Abrir descripción del puesto

Deadline Cloud utiliza la especificación Open Job Description (OpenJD) para especificar los detalles de un trabajo. OpenJD se desarrolló para definir trabajos que son transferibles entre soluciones. Se usa para definir un trabajo que es un conjunto de comandos que se ejecutan en los hosts de los trabajadores.

Puedes crear una plantilla de trabajo de OpenJD con un remitente que te proporciona Deadline Cloud, o puedes usar cualquier herramienta que desees para crear la plantilla. Después de crear la plantilla, la envías a Deadline Cloud. Si utilizas un remitente, este se encarga de enviar la plantilla. Si has creado la plantilla de otra forma, llamas a una acción de línea de comandos de Deadline Cloud o puedes usar una de ellas AWS SDKs para enviar el trabajo. De cualquier forma, Deadline Cloud añade el trabajo a la cola especificada y programa el trabajo.

Conceptos y terminología para Deadline Cloud

Para ayudarte a empezar a usar AWS Deadline Cloud, en este tema se explican algunos de sus conceptos y terminología clave.

Recursos agrícolas

Este diagrama muestra cómo funcionan juntos los recursos de la granja de Deadline Cloud.

Granja

Una granja contiene todos los demás recursos relacionados con el envío y la ejecución de trabajos. Las granjas son independientes unas de otras, lo que las hace útiles para separar los entornos de producción.

Cola

Una cola contiene los trabajos para programarlos en las flotas asociadas. Los usuarios pueden enviar los trabajos a una cola y administrar su prioridad y estado dentro de la cola. Una cola debe estar asociada a una flota con una asociación de flota-cola para que sus trabajos se ejecuten, y las colas pueden estar asociadas a varias flotas.

Flota

Una flota contiene capacidad de cómputo para ejecutar tareas. Las flotas se pueden gestionar mediante el servicio o gestionadas por el cliente. Las flotas gestionadas por el servicio se

ejecutan en Deadline Cloud e incluyen funciones integradas, como el escalado automático, las licencias y el acceso al software. Las flotas gestionadas por el cliente se ejecutan en sus propios recursos informáticos, como EC2 instancias de Amazon o servidores locales.

Realizar presupuestos

Un presupuesto establece los umbrales de gasto para su actividad laboral y le permite tomar medidas cuando se alcanzan los umbrales, como detener la programación de los trabajos.

Entorno de colas

Un entorno de colas define los scripts que se ejecutan en cada trabajador para configurar o desmantelar el entorno de carga de trabajo. Son útiles para establecer variables de entorno, instalar software y configurar el almacenamiento de activos.

Perfil de almacenamiento

Un perfil de almacenamiento es una configuración para un grupo de hosts y estaciones de trabajo que indica dónde se encuentran los datos en el sistema de archivos. Deadline Cloud usa perfiles de almacenamiento para mapear las rutas cuando se ejecutan trabajos en hosts configurados de manera diferente, como un trabajo enviado desde Windows y en Linux ejecución.

Límite

Un límite te permite realizar un seguimiento del uso de los recursos compartidos, como las licencias flotantes, y controlar cómo se asignan entre los trabajos. Los límites están asociados a las colas con asociaciones de límites de cola.

Supervisión

El monitor configura la URL de la aplicación web Deadline Cloud Monitor, lo que permite a los usuarios finales supervisar y gestionar los trabajos. Se puede acceder a él en un navegador o a través de la aplicación de escritorio Deadline Cloud Monitor.

Recursos de ejecución de trabajos

Este diagrama muestra cómo funcionan juntos los recursos de trabajo de Deadline Cloud.

Trabajo

Un trabajo es un conjunto de trabajos que un usuario envía a Deadline Cloud para programarlos y ejecutarlos con los trabajadores disponibles. Un trabajo puede renderizar una escena 3D o

ejecutar una simulación. Los trabajos se crean a partir de plantillas de trabajo reutilizables, que definen el entorno de ejecución y los procesos, así como los parámetros específicos del trabajo. Los trabajos contienen pasos y tareas que definen el trabajo que se debe realizar y se pueden configurar con las prioridades, el número máximo de trabajadores y los reintentos.

Prioridad del trabajo

La prioridad del trabajo es el orden aproximado en que Deadline Cloud procesa un trabajo en una cola. Puede establecer la prioridad de los trabajos entre 1 y 100; los trabajos con una prioridad numérica más alta generalmente se procesan primero. Los trabajos con la misma prioridad se procesan en el orden en que se reciben.

Propiedades del trabajo

Las propiedades del trabajo son ajustes que se definen al enviar un trabajo de renderizado. Algunos ejemplos incluyen el rango de fotogramas, la ruta de salida, los archivos adjuntos del trabajo, la cámara renderizable y más. Las propiedades varían en función del DCC desde el que se envía el renderizado.

Paso

Un paso forma parte de un trabajo y proporciona una plantilla para ejecutar muchas tareas que son idénticas, excepto en lo que respecta a los valores de los parámetros de la tarea. Los pasos pueden depender de otros pasos, lo que le permite crear flujos de trabajo complejos con rutas de ejecución secuenciales o paralelas. En los trabajos de renderizado, un paso suele definir el comando para renderizar un fotograma y utiliza el número de fotograma como parámetro de la tarea.

Tarea

Una tarea es la unidad de trabajo más pequeña de Deadline Cloud. Las tareas forman parte de los pasos y las ejecutan los trabajadores, lo que representa operaciones individuales que deben realizarse como parte de un trabajo. Las tareas se pueden configurar con parámetros específicos y se asignan a los trabajadores en función de sus capacidades y disponibilidad. En los trabajos de renderizado, una tarea suele renderizar un único fotograma.

Entorno de trabajo

Los trabajadores forman parte de una flota y ejecutan tareas a partir de sus trabajos. Los trabajadores se pueden configurar con capacidades específicas, como los aceleradores de GPU, la arquitectura de la CPU y el sistema operativo. En las flotas gestionadas por el servicio, los trabajadores se crean automáticamente a medida que la flota se amplía y se amplía.

instancia

Las flotas utilizan instancias como recursos de CPU. Una instancia es una instancia de EC2 rendimiento de Amazon. Deadline Cloud utiliza instancias puntuales y bajo demanda.

Instancia bajo demanda

Las instancias bajo demanda tienen un precio por segundo, no tienen un compromiso a largo plazo y no se interrumpirán.

Instancia puntual

Las instancias puntuales son una capacidad sin reservas que puede utilizar a un precio reducido, pero puede verse interrumpida por las solicitudes bajo demanda.

Espere y ahorre

La función de esperar y ahorrar permite retrasar la programación de los trabajos a un costo menor y puede interrumpirse con solicitudes puntuales y bajo demanda. Wait and Save solo está disponible en las flotas gestionadas por el servicio Deadline Cloud.

Wait and Save sirve para gestionar la ejecución de cargas de trabajo de computación visual en Deadline Cloud. AWS Consulte las condiciones AWS del servicio para obtener más información.

Sesión

Una sesión representa la secuencia de trabajo de un trabajador en un puesto de trabajo. Durante una sola sesión, a un trabajador se le pueden asignar varias tareas que ejecuta una tras otra. Las sesiones suelen incluir acciones de configuración que configuran los entornos y cargan los activos antes de ejecutar las acciones de la tarea.

Acción de sesión

Una acción de sesión representa operaciones específicas que se realizan durante una sesión, como la configuración del entorno, la ejecución de una tarea y la sincronización de activos.

Otros conceptos y terminología importantes

Explorador de uso

El explorador de uso es una función del monitor Deadline Cloud. Proporciona una estimación aproximada de sus costos y uso.

Gestor de presupuesto

El gestor de presupuestos forma parte del monitor de Deadline Cloud. Use el administrador de presupuestos para crear y administrar presupuestos. También puede usarlo para limitar las actividades y mantenerse dentro del presupuesto.

Biblioteca de clientes de Deadline Cloud

La biblioteca de clientes de código abierto incluye una interfaz de línea de comandos y una biblioteca para administrar Deadline Cloud. La funcionalidad incluye enviar paquetes de trabajos basados en la especificación Open Job Description a Deadline Cloud, descargar los resultados de los adjuntos de trabajos y monitorear su granja mediante la interfaz de línea de comandos (CLI).

Aplicación de creación de contenido digital (DCC)

Las aplicaciones de creación de contenido digital (DCCs) son productos de terceros con los que se crea contenido digital. Deadline Cloud incorpora integraciones con muchas de ellas, DCCs como Autodesk Maya, Blender y Maxon Cinema 4D, lo que te permite enviar trabajos desde el DCC y renderizarlos en flotas gestionadas por el servicio con software y licencias preconfigurados.

Adjuntos de trabajo

Los adjuntos de trabajo son una función de Deadline Cloud que permite cargar y descargar activos como parte de un trabajo, como texturas, modelos 3D y equipos de iluminación. Los adjuntos de trabajo se almacenan en Amazon S3 y evitan la necesidad de almacenamiento en red compartido.

Plantilla de trabajo

Una plantilla de trabajo define el entorno de ejecución y todos los procesos que se ejecutan como parte de un trabajo de Deadline Cloud.

Presentador de Deadline Cloud

Un remitente de Deadline Cloud es un complemento para un DCC que permite a los usuarios enviar trabajos fácilmente desde el DCC.

Punto final de licencia

Un punto final de licencia hace que las licencias basadas en el uso de Deadline Cloud para productos de terceros estén disponibles dentro de su VPC. Este modelo es de pago por uso y se le cobra por la cantidad de horas y minutos que utilice. Los puntos finales de las licencias no están conectados a las granjas y se pueden utilizar de forma independiente.

Tags

Una etiqueta es una etiqueta que se puede asignar a un AWS recurso. Cada etiqueta consta de una clave y un valor opcional definido por usted. Con las etiquetas, puede clasificar AWS los recursos de diferentes maneras, por ejemplo, por propósito, propietario o entorno.

Licencias basadas en el uso (UBL)

La licencia basada en el uso (UBL) es un modelo de licencia bajo demanda que está disponible para determinados productos de terceros. Este modelo es de pago por uso y se le cobra por la cantidad de horas y minutos que utilice.

Guía de arquitectura de Deadline Cloud

Este tema proporciona orientación y prácticas recomendadas para diseñar y crear granjas de renderizado fiables, seguras, eficientes y rentables para sus cargas de trabajo mediante Deadline Cloud. El uso de esta guía puede ayudarlo a crear cargas de trabajo estables y eficientes, lo que le permitirá centrarse en la innovación, reducir los costos y mejorar la experiencia de sus clientes.

Este contenido está dirigido a directores de tecnología (CTOs), arquitectos, desarrolladores y miembros del equipo de operaciones.

Un flujo de trabajo de end-to-end renderizado requiere soluciones en varios niveles del proceso, como la generación de empleos, el acceso a los activos y la supervisión de los trabajos. Deadline Cloud ofrece múltiples soluciones para cada capa del proceso de renderizado. Al seleccionar las opciones de Deadline Cloud en cada capa, puede diseñar un flujo de trabajo que se adapte a su caso de uso.

Para cada capa, tendrás que decidir qué enfoque es mejor para tu caso de uso. Estas no son definiciones de escenarios estrictas y no son la única forma de utilizar Deadline Cloud. En cambio, se trata de un conjunto de conceptos de alto nivel que le ayudarán a comprender cómo Deadline Cloud podría adaptarse a su negocio o flujo de trabajo. Puede separar las cargas de trabajo de Deadline Cloud en las siguientes capas: Job Source, Job Submission, Application Management, Application Licensing, Asset Access, Output Management y Worker Infrastructure Management.

En general, puede combinar mix-and-match cualquier escenario de una capa con cualquier otro escenario de otra capa, excepto las combinaciones específicas que se especifican a continuación.


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

Fuente de trabajo

La fuente de trabajo es el punto de acceso desde el que ingresarán nuevos trabajos al sistema para que Deadline Cloud los procese. A un alto nivel, hay dos fuentes principales de puestos de trabajo: la interactividad humana y los sistemas informáticos automatizados.

Flujo de trabajo interactivo

En este escenario, un artista u otro rol creativo es el principal generador del trabajo que se procesará en la granja de Deadline Cloud. Por lo general, el resultado de estos trabajos es un artefacto principal para el proyecto o el equipo más grande. Realizan su trabajo mediante software, como una herramienta de creación de contenido digital (DCC) estándar del sector. Están enviando los trabajos manualmente a la granja de Deadline Cloud y, posteriormente, visualizan los resultados para revisarlos. La estación de trabajo en sí no está gestionada por AWS.

En la mayoría de los casos, estos artistas utilizan los emisores integrados de Deadline Cloud y el monitor de Deadline Cloud en las capas de carga de trabajo, aplicación y supervisión.

Flujo de trabajo automatizado

En este escenario, un sistema programático propiedad del cliente es el principal generador de trabajos en la granja de Deadline Cloud. Podría tratarse de la generación de activos en un proceso de venta minorista, como un vídeo con tocadiscos generado a partir de un modelo 3D o escaneado. Esto podría consistir en la composición automática de gráficos de retransmisiones y tarjetas de jugadores para deportes. El tema de este escenario es que una persona no envía manualmente cada trabajo a Deadline Cloud, sino que el trabajo se genera como parte de un sistema más grande.

Con los trabajos automatizados, es menos común que se utilicen los remitentes integrados de Deadline Cloud y el monitor de Deadline Cloud. A menudo, las definiciones de los puestos se basan en aplicaciones personalizadas y redactadas por usted, y los resultados de los trabajos pasan automáticamente a un sistema de gestión de activos digitales (DAM) o a un sistema de gestión de activos multimedia (MAM) para su aprobación y distribución.

Presentación de trabajos

Los trabajos se envían a Deadline Cloud mediante OpenJobDescriptionplantillas. OpenJobDescription es una especificación abierta y flexible para definir los trabajos de procesamiento por lotes que se pueden transportar entre diferentes implementaciones de sistemas

de programación. El archivo de definición del trabajo describe los parámetros del trabajo, los pasos del trabajo, cómo se parametriza un paso en función de las entradas del trabajo, así como el script real que se ejecutará en un Worker para realizar el procesamiento. La idea del envío de la carga de trabajo es cómo se crean estas definiciones de trabajo, quién las crea y cómo se envían.

Presentador integrado con DCC

Un remitente integrado de Deadline Cloud es un software que une Deadline Cloud con un DCC o paquete de software estándar del sector. El remitente integrado determina cómo transformar los datos y la configuración de una carga de trabajo renderizada, compuesta u otra carga de trabajo en una plantilla de trabajo, algo que Deadline Cloud puede entender. Muchos de los remitentes integrados los crea y mantiene el equipo de Deadline Cloud o el creador del paquete de software, pero si aún no existe ninguno para la aplicación deseada, puedes crear y mantener tu propio remitente. El equipo de DCCs Deadline Cloud admite un conjunto limitado de ellos.

Los flujos de trabajo interactivos suelen incluir remitentes integrados, pero no siempre. En el caso de los flujos de trabajo automatizados y con plantillas, un flujo de trabajo habitual consiste en que un artista configure un trabajo de plantilla en su DCC y realice una exportación única del paquete de trabajos. Este paquete de trabajos define cómo ejecutar ese tipo de trabajo en particular en Deadline Cloud de forma parametrizada. Este paquete de trabajos se puede integrar en el escenario del flujo de trabajo automatizado con fines de automatización.

Definición de trabajo personalizada

En el caso de las aplicaciones y los flujos de trabajo personalizados, es posible controlar por completo la forma en que se crean y envían estas definiciones de trabajo a Deadline Cloud. Por ejemplo, un sitio de comercio electrónico puede pedir a los vendedores que suban modelos 3D del objeto que venden. Tras esta subida, la plataforma de comercio electrónico podría generar de forma dinámica una definición de trabajo para enviarla a Deadline Cloud y generar automáticamente una animación con un tocadiscos sobre un fondo común y utilizar una iluminación común para que coincida con los demás objetos 3D disponibles en el sitio. Durante el desarrollo de la plataforma de comercio electrónico, un desarrollador de software creaba una definición de trabajo, la integraba en la plataforma de comercio electrónico con los parámetros que eventualmente proporcionaban los vendedores y codificaba la plataforma para que enviara este trabajo durante el flujo de carga de productos de la plataforma.

Deadline Cloud ofrece varios ejemplos de definiciones de trabajo en el repositorio de ejemplos de github.

Application Management (Administración de aplicaciones)

Después de enviar un trabajo a Deadline Cloud y asignarlo a un trabajador, el script de la definición del trabajo se ejecuta en el trabajador. En la mayoría de los casos, este script invocará una aplicación para realizar el procesamiento propiamente dicho, como un renderizador, una composición, una codificación, un filtrado o cualquier otra tarea que requiera un uso intensivo de la computación. La gestión de aplicaciones consiste en garantizar que los trabajadores dispongan de la versión necesaria del software necesario.

Puede administrar las aplicaciones con cualquier sistema de administración de paquetes que desee, pero Deadline Cloud proporciona una serie de herramientas para permitir fácilmente el uso de los paquetes conda. Conda es un administrador de paquetes y un sistema de gestión de entornos de código abierto, multiplataforma e independiente del idioma.

Deadline: canal conda gestionado en la nube para flotas gestionadas por servicios (SMF)

Al utilizar flotas gestionadas por servicios, un canal conda gestionado por Deadline Cloud se configura y configura automáticamente para que lo utilice en sus puestos de trabajo. El servicio Deadline Cloud proporciona una serie de aplicaciones de DCC asociadas y las renderiza en este canal conda. Para obtener más información, consulte Crear un entorno de colas en la guía del usuario de Deadline Cloud. El servicio Deadline Cloud actualiza automáticamente estos paquetes y no requieren mantenimiento por tu parte. Este canal conda solo está disponible cuando se utilizan flotas gestionadas por el servicio y no está disponible cuando se utilizan flotas gestionadas por el cliente.

Canal conda autogestionado

Si no puede utilizar el canal conda administrado por Deadline Cloud, debe determinar cómo instalar, parchear y gestionar las aplicaciones de su flota de Deadline Cloud. Una opción es crear un canal conda que puedas configurar y mantener. Esto interoperará más estrechamente con el canal conda administrado por Deadline Cloud. Por ejemplo, puedes usar un DCC del canal conda administrado por Deadline Cloud, pero traer tu propio paquete que contenga un complemento de DCC específico. Para obtener más información sobre este proceso, consulte Crear un canal conda con S3.

Administración de aplicaciones personalizada

Para la administración de aplicaciones, el requisito de Deadline Cloud es que la aplicación esté disponible en el PATH cuando se ejecute el script de trabajo en el trabajador.

Si ya ha creado y mantenido los paquetes de Rez, puede utilizar un entorno de colas para instalar las aplicaciones desde los repositorios de Rez. Puedes encontrar un ejemplo de entorno de colas en AWS Deadline Cloud org. GitHub

Si ya gestiona las aplicaciones en flotas gestionadas por un cliente con empleados de larga trayectoria o en imágenes del sistema, no necesitará ningún entorno de colas para la gestión de las aplicaciones. Asegúrese de que la solicitud aparezca en la ruta del usuario del trabajo y envíe el trabajo.

Application licensing

Muchas cargas de trabajo que normalmente se ejecutan en Deadline Cloud requieren una licencia de software del proveedor del software. Estas aplicaciones se suelen licenciar por puesto, por CPU o por host. Es su responsabilidad asegurarse de que el uso de software de terceros en Deadline Cloud cumpla con el acuerdo de licencia de terceros. Si utilizas software de código abierto, software personalizado o software sin licencia, no es necesario configurar esta capa. Tenga en cuenta que Deadline Cloud solo admite licencias de renderizado y no admite licencias de estaciones de trabajo.

Flotas gestionadas por servicios y licencias basadas en el uso

Cuando se utilizan flotas gestionadas por el servicio de Deadline Cloud, las licencias basadas en el uso (UBL) se configuran automáticamente para el software compatible. Los trabajos que se ejecutan en flotas gestionadas por servicios tienen automáticamente variables de entorno configuradas para las aplicaciones compatibles, a fin de indicarles que utilicen los servidores de licencias de Deadline Cloud. Al usar Deadline Cloud UBL, solo se le cobrará por la cantidad de horas que use la aplicación con licencia.

Flotas gestionadas por el cliente y licencias basadas en el uso

Las licencias basadas en el uso (UBL) de Deadline Cloud también están disponibles cuando no se utilizan flotas gestionadas por el servicio. En este escenario, configurará puntos finales de licencia de Deadline Cloud que proporcionan direcciones IP en las subredes de VPC seleccionadas que proporcionan acceso a los servidores de licencias de Deadline Cloud. Tras configurar las variables

de entorno específicas del software adecuadas para sus trabajadores y configurar la conectividad de red entre los trabajadores y las direcciones IP de los puntos de conexión de las licencias, los trabajadores podrán retirar y registrar las licencias del software compatible. Las licencias se cobran por hora igual que cuando se utiliza la UBL en flotas gestionadas por servicios.

Licencias personalizadas

Puede utilizar una aplicación que no sea compatible con Deadline Cloud UBL o puede que tenga licencias preexistentes que sigan siendo válidas. En este escenario, usted es responsable de configurar la ruta de red desde sus trabajadores (gestionada por el cliente o por el servicio) hasta los servidores de licencias. Para obtener más información sobre las licencias personalizadas, consulte. Connect las flotas gestionadas por el servicio a un servidor de licencias personalizado

Acceso a los activos

Una vez que se envía un trabajo a un trabajador y se configura la aplicación, el trabajador debe estar configurado para acceder a los datos de activos necesarios para el trabajo. Pueden ser datos 3D, datos de textura, datos de animación, fotogramas de vídeo o cualquier otro tipo de datos utilizados en su trabajo.

Comience por pensar dónde están almacenados actualmente sus datos. Puede estar en el disco duro de la estación de trabajo, en una herramienta de colaboración de usuarios, en el control de código fuente, en un sistema de archivos compartido local o en la nube, en Amazon S3 o en cualquier otro lugar.

A continuación, considere qué es lo que necesita un trabajador para acceder a estos datos. ¿Estos datos solo están disponibles en su red corporativa? ¿Qué identidad o credenciales se requieren para acceder a los datos? ¿La fuente de datos está escalada para respaldar el trabajo con la cantidad de trabajadores que espera que procesen el trabajo?

Adjuntos de trabajo

El mecanismo más fácil de empezar para acceder a los activos es el de los adjuntos de trabajo de Deadline Cloud. Cuando se envía un trabajo mediante adjuntos de trabajo, los datos requeridos por el trabajo se cargan en un bucket de Amazon S3 junto con un archivo de manifiesto que especifica qué archivos requiere el trabajo. En el caso de los adjuntos de trabajo, no es necesaria una configuración complicada de redes o almacenamiento compartido. Los archivos solo se cargan una vez, por lo que las cargas posteriores se completan más rápidamente. Cuando un trabajador

termina de procesar un trabajo, los datos de salida se cargan en Amazon S3 para que el artista u otro cliente puedan descargarlos. La báscula Job Adjuntos se adapta a flotas de cualquier tamaño y es fácil y rápida de incorporar y utilizar.

Job attachments no es la mejor herramienta para todas las situaciones. Si los datos ya están guardados AWS, los adjuntos de trabajo añaden una copia adicional de los datos, incluidos el tiempo de transferencia y los costes de almacenamiento asociados. Los adjuntos de trabajo requieren que el trabajo pueda especificar completamente los datos que requiere en el momento del envío, de modo que los datos se puedan cargar.

Para usar adjuntos de trabajos, tu cola de Deadline Cloud debe tener un grupo de adjuntos de trabajos asociado y el rol de cola debe usarse para proporcionar acceso a ese grupo. De forma predeterminada, todos los remitentes integrados de Deadline Cloud admiten adjuntos de trabajo. Si no utilizas un remitente integrado en Deadline Cloud, los adjuntos de trabajo se pueden utilizar con tu software personalizado integrando la biblioteca python de Deadline Cloud.

Acceso personalizado al almacenamiento

Si no utiliza adjuntos de trabajo, es responsable de garantizar que los trabajadores tengan acceso a los datos necesarios para los trabajos. Deadline Cloud proporciona una serie de herramientas para respaldar esta tarea y facilitar la portabilidad de los trabajos. Es posible que desees utilizar una solución de almacenamiento personalizada si ya dispones de almacenamiento en red compartido para artistas y trabajadores, si prefieres utilizar un servicio externo o por alguna otra razón. LucidLink

Utilice los perfiles de almacenamiento para modelar los sistemas de archivos de su estación de trabajo y de los hosts de trabajo. Cada perfil de almacenamiento describe el diseño del sistema operativo y del sistema de archivos de una de las configuraciones del sistema. Con los perfiles de almacenamiento, cuando un artista que utiliza una estación de trabajo con Windows envía un trabajo procesado por un Linux trabajador, Deadline Cloud se asegura de que se realice un mapeo de rutas para que el trabajador pueda acceder al almacenamiento de datos que ha configurado.

Al utilizar flotas gestionadas por el servicio de Deadline Cloud, los scripts de configuración del host y los puntos finales de recursos de VPC permiten a los trabajadores montar y acceder directamente al almacenamiento compartido u otros servicios disponibles en su VPC.

Supervisión de trabajos y gestión de resultados

Una vez que los trabajos enviados a Deadline Cloud se hayan completado correctamente, una persona o un proceso descargará el resultado del trabajo para usarlo en el flujo de trabajo

empresarial fuera de Deadline Cloud. Tras un fallo en un trabajo, los registros de trabajos y la información de supervisión ayudan a diagnosticar los problemas.

Monitor Deadline Cloud

La aplicación de monitoreo Deadline Cloud está disponible en la web y para computadoras de escritorio. Esta solución es la más adecuada para los estudios que utilizan flujos de trabajo interactivos, ya que DCCs utilizan una amplia gama de archivos adjuntos de trabajo para el almacenamiento. El monitor solo es compatible con el IAM Identity Center. IAM Identity Center es un producto de Workforce Identity, no una solución de identidad del consumidor (B2C), por lo que no es adecuado para muchos escenarios B2C.

Aplicación de monitorización personalizada

Si desea personalizar la experiencia de monitoreo de sus usuarios, está creando un producto B2C o está creando un sistema altamente especializado con Deadline Cloud, opta por crear una aplicación de monitoreo personalizada. Puedes usar la API de AWS Deadline Cloud para crear esta aplicación personalizada, combinando el contexto de tu flujo de trabajo general con los conceptos de Deadline Cloud. Por ejemplo, tu producto B2C puede tener su propio concepto de proyecto que los usuarios configuran y tu aplicación puede agrupar los trabajos de Deadline Cloud en la misma interfaz.

Solución de monitorización automatizada

En algunos escenarios, no se necesita una aplicación de monitoreo dedicada para Deadline Cloud. Este escenario es común en los flujos de trabajo automatizados, en los que Deadline Cloud se utiliza para renderizar automáticamente los activos de una canalización, como gráficos de retransmisiones de deportes o noticias. En este escenario, la API y los EventBridge eventos de Deadline Cloud se utilizan para integrarse con un sistema externo de gestión de activos multimedia para aprobarlos y pasar a la siguiente fase del proceso.

Gestión de la infraestructura de los trabajadores

Las flotas de Deadline Cloud son un grupo de servidores (trabajadores) que pueden procesar los trabajos enviados a una cola de Deadline Cloud y son la infraestructura central de cualquier granja de Deadline Cloud.

Flotas gestionadas por el servicio

En una flota gestionada por servicios, Deadline Cloud asume la responsabilidad de los servidores de los trabajadores, el sistema operativo, las redes, los parches, el escalado automático y otros factores relacionados con el funcionamiento de una granja de renderizados. Tú especificas el número mínimo y máximo de trabajadores que deseas, junto con las especificaciones del sistema requeridas para tu solicitud, y Deadline Cloud se encarga del resto. Las flotas gestionadas por servicios son la única opción de flota que puede utilizar los canales conda gestionados por Deadline Cloud para gestionar fácilmente las aplicaciones de DCC del sector. Además, la UBL de Deadline Cloud se configura automáticamente con las flotas gestionadas por el servicio. Las flotas de Wait and Save permiten reducir los costes y las cargas de trabajo tolerantes a demoras solo están disponibles con flotas gestionadas por el servicio.

Flotas gestionadas por el cliente

Utiliza flotas gestionadas por el cliente cuando necesita tener más control sobre los anfitriones de los trabajadores y su entorno. Las flotas gestionadas por el cliente son las más adecuadas cuando se utiliza Deadline Cloud de forma local. Para obtener más información, consulte Cree y utilice flotas gestionadas por los clientes de Deadline Cloud.

Ejemplos de arquitecturas

Estudio de producción tradicional

El estudio de producción tradicional requiere una importante infraestructura de cómputo, almacenamiento y redes que pueda abarcar varias ubicaciones físicas para atender las cargas de trabajo de renderizado. Cada paquete de software y proveedor individual tiene requisitos únicos de hardware, software, redes y licencias que deben cumplirse al tiempo que se resuelven los conflictos de versiones, compatibilidad y recursos.

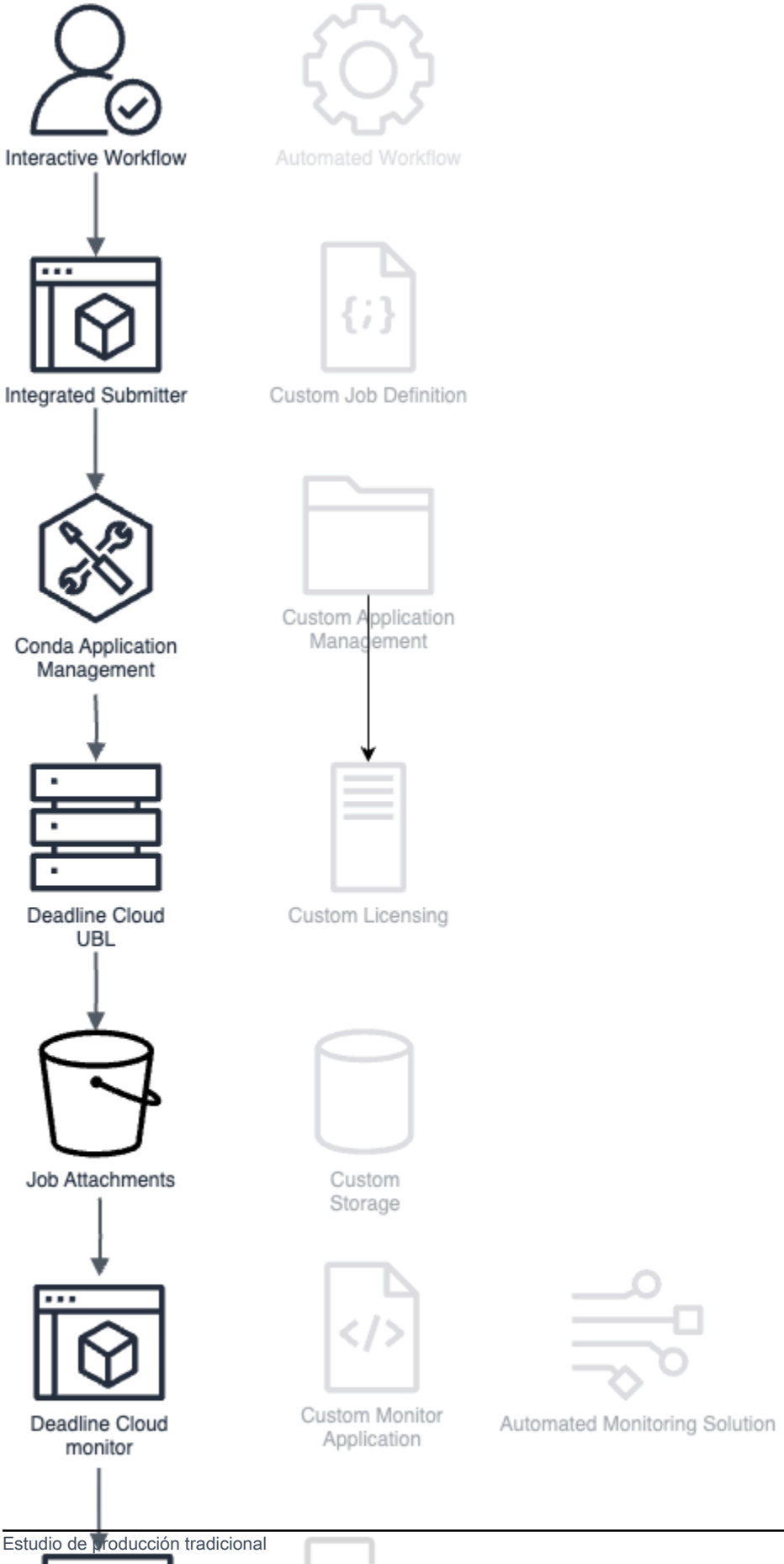
Es habitual tener requisitos de infraestructura distintos para las estaciones de trabajo de los artistas, los nodos de renderizado, el almacenamiento en red, los servidores de licencias, los sistemas de cola de trabajos, las herramientas de supervisión y la gestión de activos. Por lo general, los estudios necesitan mantener varias versiones de las herramientas de DCC, los renderizadores, los complementos y las herramientas personalizadas y, al mismo tiempo, gestionar complejos acuerdos de licencia en toda su granja de renderización. La infraestructura de su estudio se vuelve más complicada si se tienen en cuenta los entornos de desarrollo, control de calidad y producción.

Una implementación típica de Deadline Cloud con opciones de administración de servicios resuelve o reduce muchos de estos desafíos mediante:

- Envío de trabajos de flujo de trabajo interactivo a través de remitentes de DCC integrados
- Gestión de solicitudes a través de los canales conda gestionados por Deadline Cloud
- Las licencias basadas en el uso se configuran automáticamente para el software compatible
- Gestión de activos mediante adjuntos de trabajo
- Supervisión a través de la aplicación de monitorización Deadline Cloud
- Gestión de infraestructuras mediante flotas gestionadas por servicios

Con este enfoque, los artistas pueden enviar sus trabajos directamente desde sus conocidas herramientas de DCC a una granja de renderización en la nube escalable sin administrar una infraestructura compleja. El servicio gestiona automáticamente la implementación del software, las licencias, la transferencia de datos y el escalado de la infraestructura. Los artistas pueden supervisar sus trabajos a través de una interfaz web o una aplicación de escritorio, y los resultados se almacenan automáticamente en Amazon S3 para facilitar el acceso.

Con esta configuración, los estudios pueden crear entornos de desarrollo y producción en cuestión de minutos, pagar solo por el procesamiento y las licencias que utilizan y centrarse en el trabajo creativo más que en la administración de la infraestructura. El enfoque de gestión de servicios ofrece la forma más rápida de adoptar el renderizado en la nube y, al mismo tiempo, mantener los flujos de trabajo habituales para los artistas.


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

Estudio en la nube

Los estudios de animación y efectos visuales modernos están trasladando cada vez más toda su cartera a la nube, incluidas las estaciones de trabajo de los artistas. Este enfoque elimina la necesidad de una infraestructura local, permite la colaboración global y proporciona una escalabilidad perfecta tanto para el trabajo interactivo como para la renderización. Sin embargo, también presenta nuevos desafíos a la hora de administrar los recursos de la nube, garantizar el acceso de baja latencia a los datos e integrar las estaciones de trabajo basadas en la nube con los conjuntos de renderizados.

Un estudio nativo de la nube típico requiere un enfoque unificado para administrar las estaciones de trabajo en la nube, el almacenamiento compartido, la infraestructura de renderización y la implementación de software en todos estos componentes. Los enfoques tradicionales solían dar como resultado sistemas complejos gestionados manualmente que tenían dificultades para equilibrar el rendimiento, el coste y la flexibilidad.

Se puede implementar una implementación de Deadline Cloud para un estudio nativo de la nube mediante:

- Envío de trabajos de flujo de trabajo interactivo a través de remitentes de DCC integrados en estaciones de trabajo en la nube
- Gestión de aplicaciones a través de los canales conda gestionados por Deadline Cloud y nodos de renderización
- Las licencias basadas en el uso se configuran automáticamente para el software compatible
- Acceso personalizado al almacenamiento mediante el servidor FSx de Windows archivos para compartir los datos del proyecto
- Supervisión a través de la aplicación de monitorización Deadline Cloud
- Gestión de infraestructuras mediante flotas gestionadas por servicios

Este enfoque permite a los artistas trabajar en estaciones de trabajo basadas en la nube con acceso directo a un almacenamiento compartido de alto rendimiento y enviar sus trabajos sin problemas a la granja de Deadline Cloud. El estudio puede gestionar la implementación del software tanto en las estaciones de trabajo como en los nodos de renderizado utilizando los mismos canales conda, lo que garantiza la coherencia y reduce los gastos de mantenimiento.

Entre las principales ventajas de esta configuración se incluyen las siguientes:

- Colaboración global con artistas que pueden acceder a las estaciones de trabajo desde cualquier lugar
- Entornos de software uniformes en todas las estaciones de trabajo y nodos de renderizado
- Almacenamiento compartido de alto rendimiento accesible tanto para las estaciones de trabajo como para los nodos de renderizado
- Escalado flexible de los recursos informáticos interactivos y por lotes
- Administración centralizada de toda la infraestructura del estudio en la nube

La configuración del almacenamiento en este escenario suele implicar:

- FSx para el servidor de Windows archivos para los datos del proyecto, al que pueden acceder tanto las estaciones de trabajo en la nube como los trabajadores de Deadline Cloud
- Almacene perfiles en Deadline Cloud para gestionar el mapeo de rutas entre las estaciones de trabajo y los nodos de renderizado
- Montaje directo de FSx recursos compartidos en los trabajadores de Deadline Cloud mediante puntos finales de recursos de VPC y scripts de configuración de hosts

Este enfoque nativo de la nube permite a los estudios eliminar la infraestructura local, lo que permite escalarlos rápidamente para proyectos de cualquier tamaño y, al mismo tiempo, mantener los flujos de trabajo de los artistas habituales. Ofrece la flexibilidad necesaria para utilizar una combinación de recursos gestionados por el servicio y gestionados por el cliente, lo que optimiza la facilidad de administración y los requisitos de rendimiento específicos.

Al aprovechar las estaciones de trabajo en la nube junto con Deadline Cloud, los estudios pueden lograr un proceso de producción totalmente integrado y accesible a nivel mundial, que se adapta sin problemas desde pequeños equipos hasta grandes producciones.

ECommerce Automation

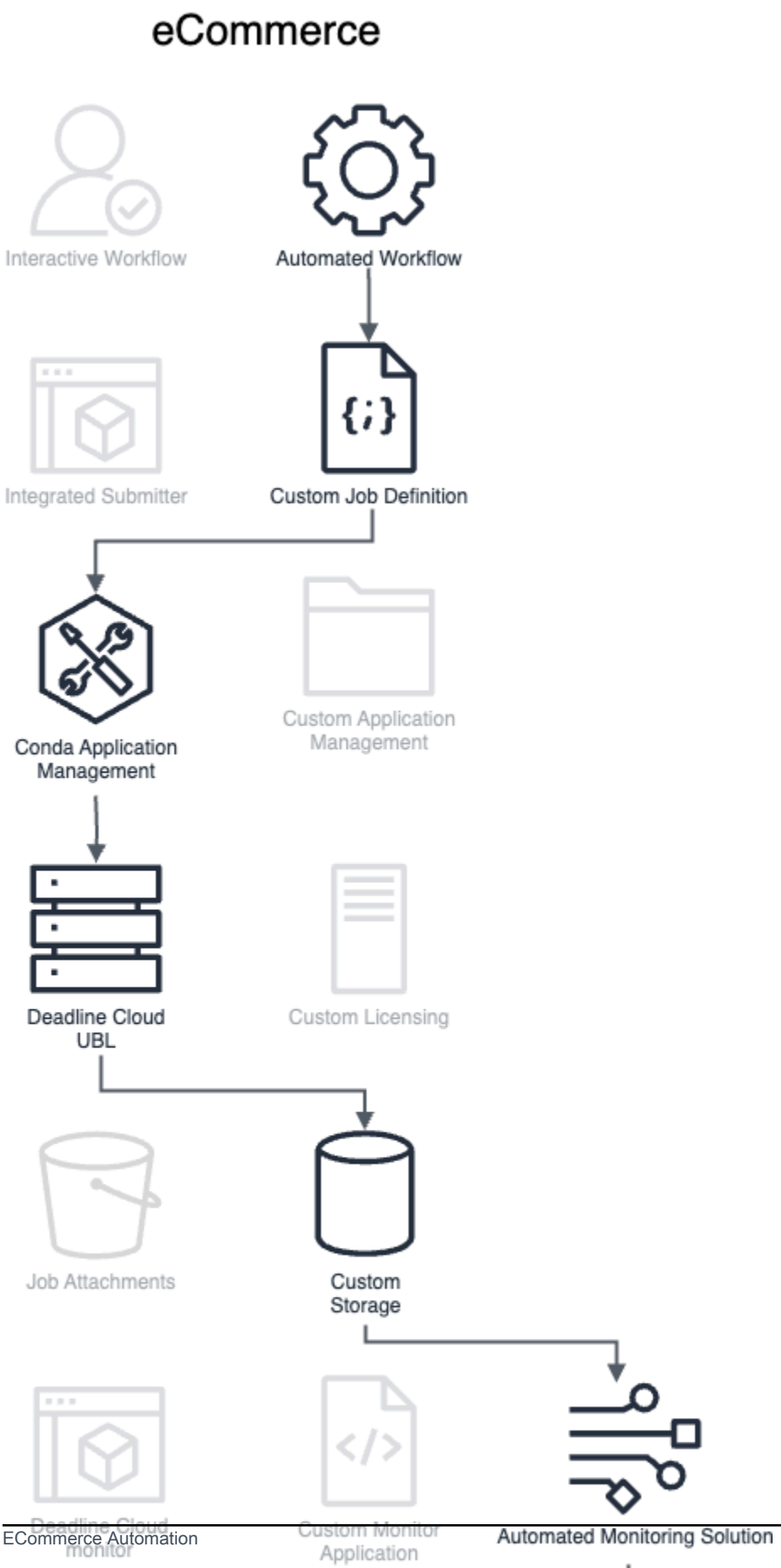
La plataforma de comercio electrónico moderna requiere la generación automática de activos a escala para ofrecer una visualización completa de los productos en millones de artículos. Los enfoques tradicionales requerían una importante inversión en infraestructura para procesar grandes volúmenes de modelos 3D y convertirlos en soportes de producto estandarizados, lo que a menudo daba como resultado sistemas insuficientemente aprovisionados que generaban retrasos en el procesamiento o sistemas sobreaprovisionados con capacidad inactiva.

Un flujo de trabajo de comercio electrónico automatizado típico debe gestionar la carga de productos, el procesamiento, la validación de modelos 3D, la gestión de los parques de renderizados, el procesamiento de la producción y la integración con los sistemas de información sobre los productos. La gestión de estos flujos de trabajo tradicionalmente requiere coordinar múltiples aplicaciones de renderizado, recursos de cómputo y procesos de procesamiento de datos, al tiempo que se garantiza una calidad uniforme y se mantiene la rentabilidad a escala.

Se puede implementar una implementación de Deadline Cloud para la automatización del comercio electrónico mediante:

- Envío automatizado de trabajos de flujo de trabajo mediante la integración de una API personalizada en la aplicación de ingestión de comercio electrónico existente
- Definiciones de trabajos personalizadas adaptadas a la visualización estandarizada de los productos
- Administración de aplicaciones a través de los canales conda administrados por Deadline Cloud
- Las licencias basadas en el uso se configuran automáticamente para el software compatible
- Integración directa de Amazon S3 para la administración de activos
- Aplicación de monitoreo personalizada integrada con los sistemas de administración de productos existentes
- Flotas gestionadas por el servicio para una escalabilidad elástica

Este enfoque permite procesar miles de productos por día y generar automáticamente visualizaciones de productos estandarizadas, como animaciones de tocadiscos. La infraestructura gestionada por el servicio se amplía automáticamente para satisfacer la demanda variable y, al mismo tiempo, mantener la rentabilidad mediante la reutilización de los trabajadores y la implementación optimizada de las aplicaciones.


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

El software de creación de contenido digital (DCC) tradicional suele requerir que los usuarios mantengan su propia infraestructura de renderizado o procesen los renderizados de forma local en su estación de trabajo, lo que conlleva importantes inversiones en hardware o largos tiempos de espera que interrumpen los flujos de trabajo creativos. Para los proveedores de software, ofrecer capacidades de renderizado en la nube solía requerir la creación y el mantenimiento de sistemas de infraestructura y facturación complejos.

Una implementación de Deadline Cloud integrada en el software B2C permite renderizar en la nube sin problemas directamente desde la interfaz familiar del usuario. Esta integración combina:

- Envío de trabajos de flujo de trabajo interactivo integrado en la aplicación DCC
- Deadline: canales conda gestionados en la nube para el despliegue de aplicaciones de renderizado
- Las licencias basadas en el uso se configuran automáticamente
- Administración de activos mediante adjuntos de trabajos con almacenamiento gestionado por el proveedor
- Supervisión personalizada integrada directamente en la interfaz DCC
- Flotas gestionadas por el servicio compartidas entre los usuarios

Este enfoque permite a los usuarios finales enviar los renderizados a la nube con un solo clic desde su software, sin tener que gestionar las cuentas, la infraestructura ni una configuración compleja. El proveedor de software mantiene un entorno multiusuario en el que:

- Los usuarios se autentican mediante sus credenciales de software existentes
- Los trabajos se redirigen automáticamente a colas dedicadas a cada usuario
- Los activos se aíslan de forma segura mediante prefijos de almacenamiento controlados por IAM
- La facturación se gestiona a través de los sistemas existentes del proveedor
- El estado del trabajo y los resultados se transmiten directamente a la aplicación del usuario

El enfoque de flota compartida garantiza un rendimiento óptimo al mantener un grupo cálido de trabajadores, minimizar los tiempos de puesta en marcha y maximizar la utilización de los recursos en toda la base de usuarios. Esta configuración permite a los proveedores de software ofrecer

la renderización en la nube como una función de producto integrada, en lugar de un servicio independiente que requiera configuraciones o cuentas adicionales.

Los usuarios finales se benefician de:

- Envío con un solo clic desde su interfaz familiar
- Pay-as-you-go precios sin administración de infraestructura
- Acelere los tiempos de inicio de los trabajos mediante una infraestructura compartida
- Descarga y organización automáticas de los renderizados finalizados
- Experiencia uniforme en todas las plataformas

Este patrón de integración permite a los proveedores de software proporcionar capacidades de renderizado de nivel empresarial a toda su base de usuarios y, al mismo tiempo, mantener una experiencia sencilla y fácil de usar para el consumidor que parece nativa de su aplicación.


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

¿Qué es una carga de trabajo de Deadline Cloud?

Con AWS Deadline Cloud, puede enviar trabajos para ejecutar sus aplicaciones en la nube y procesar datos para la producción de contenido o información crucial para su negocio. Deadline Cloud utiliza Open Job Description (OpenJD) como sintaxis para las plantillas de trabajo, una especificación diseñada para las necesidades de los procesos de procesamiento visual, pero aplicable a muchos otros casos de uso. Algunos ejemplos de cargas de trabajo incluyen la renderización de gráficos por ordenador, la simulación física y la fotogrametría.

Las cargas de trabajo van desde simples paquetes de tareas que los usuarios envían a una cola con la CLI o una GUI generada automáticamente, hasta complementos de envío integrados que generan dinámicamente un paquete de trabajos para una carga de trabajo definida por la aplicación.

Cómo surgen las cargas de trabajo de la producción

Para entender las cargas de trabajo en contextos de producción y cómo respaldarlas con Deadline Cloud, piense en cómo surgen. La producción puede implicar la creación de efectos visuales, animaciones, juegos, imágenes de catálogos de productos, reconstrucciones en 3D para el modelado de información de edificios (BIM) y mucho más. Por lo general, este contenido lo crea un equipo de especialistas artísticos o técnicos que utilizan una variedad de aplicaciones de software y secuencias de comandos personalizadas. Los miembros del equipo se transmiten datos entre sí mediante un proceso de producción. Muchas de las tareas que se llevan a cabo en el proceso implican cálculos intensivos que tardarían días si se ejecutaran en la estación de trabajo de un usuario.

Algunos ejemplos de tareas de estos procesos de producción son:

- Uso de una aplicación de fotogrametría para procesar fotografías tomadas de un set de filmación con el fin de reconstruir una malla digital texturizada.
- Realizar una simulación de partículas en una escena 3D para añadir capas de detalle a un efecto visual explosivo para un programa de televisión.
- Introduce los datos de un nivel de juego en la forma necesaria para su publicación externa y aplica los ajustes de optimización y compresión.
- Representación de un conjunto de imágenes para un catálogo de productos, incluidas las variaciones de color, fondo e iluminación.
- Ejecutar un guion desarrollado a medida en un modelo 3D para aplicar un aspecto creado a medida y aprobado por un director de cine.

Estas tareas implican ajustar muchos parámetros para obtener un resultado artístico o ajustar con precisión la calidad de la salida. A menudo, hay una GUI para seleccionar esos valores de parámetros con un botón o un menú para ejecutar el proceso localmente dentro de la aplicación. Cuando un usuario ejecuta el proceso, la aplicación y, posiblemente, el propio ordenador anfitrión no se pueden utilizar para realizar otras operaciones porque utiliza el estado de la aplicación en la memoria y puede consumir todos los recursos de CPU y memoria del ordenador anfitrión.

En muchos casos, el proceso es rápido. Durante el transcurso de la producción, la velocidad del proceso se ralentiza cuando aumentan los requisitos de calidad y complejidad. Una prueba de personaje que duró 30 segundos durante el desarrollo puede convertirse fácilmente en 3 horas si se aplica al personaje final de producción. A lo largo de esta progresión, una carga de trabajo que comenzó dentro de una interfaz gráfica de usuario puede llegar a ser demasiado grande como para caber. La migración a Deadline Cloud puede aumentar la productividad de los usuarios que ejecutan estos procesos, ya que recuperan el control total de su estación de trabajo y pueden realizar un seguimiento de más iteraciones desde el monitor de Deadline Cloud.

A la hora de desarrollar el soporte para una carga de trabajo en Deadline Cloud, hay dos niveles de soporte:

- Transferir la carga de trabajo de la estación de trabajo del usuario a una granja de Deadline Cloud sin paralelismo ni aceleración. Esto puede infrautilizar los recursos informáticos disponibles en la granja, pero la posibilidad de trasladar las operaciones largas a un sistema de procesamiento por lotes permite a los usuarios hacer más cosas con su propia estación de trabajo.
- Optimizar el paralelismo de la carga de trabajo para que utilice la escala horizontal de la granja de Deadline Cloud para completar con rapidez.

Hay veces en las que es obvio cómo hacer que una carga de trabajo se ejecute en paralelo. Por ejemplo, cada fotograma de un renderizado gráfico por ordenador se puede realizar de forma independiente. Sin embargo, es importante no quedarse atascado en este paralelismo. Por el contrario, comprenda que transferir una carga de trabajo de larga duración a Deadline Cloud ofrece beneficios significativos, incluso cuando no existe una forma obvia de dividir la carga de trabajo.

Los ingredientes de una carga de trabajo

Para especificar una carga de trabajo de Deadline Cloud, implemente un paquete de trabajos que los usuarios envíen a una cola con la CLI de Deadline Cloud. Gran parte del trabajo a la hora de crear un paquete de trabajos consiste en redactar la plantilla de trabajo, pero hay otros factores, como

la forma de proporcionar las solicitudes que requiere la carga de trabajo. Estos son los aspectos esenciales que se deben tener en cuenta al definir una carga de trabajo para Deadline Cloud:

- La aplicación que se va a ejecutar. El trabajo debe poder iniciar los procesos de la aplicación y, por lo tanto, necesita una instalación de la aplicación disponible, así como cualquier licencia que utilice la aplicación, como el acceso a un servidor de licencias flotante. Por lo general, esto forma parte de la configuración de la granja y no está integrado en el paquete de tareas en sí.
  - Configure los trabajos mediante entornos de colas
  - Connect las flotas gestionadas por el cliente a un punto final de licencia
- Definiciones de parámetros de trabajo. La experiencia del usuario al enviar el trabajo se ve afectada en gran medida por los parámetros que proporciona. Entre los parámetros de ejemplo se incluyen los archivos de datos, los directorios y la configuración de la aplicación.
  - Elementos de valores de parámetros para paquetes de trabajos
- Flujo de datos de archivos. Cuando se ejecuta un trabajo, lee la entrada de los archivos proporcionados por el usuario y, a continuación, escribe la salida como archivos nuevos. Para trabajar con los archivos adjuntos de los trabajos y las funciones de mapeo de rutas, el trabajo debe especificar las rutas de los directorios o archivos específicos para estas entradas y salidas.
  - Uso de archivos en sus trabajos
- El guion de pasos. El script de pasos ejecuta el binario de la aplicación con las opciones de línea de comandos adecuadas para aplicar los parámetros de trabajo proporcionados. También gestiona detalles como el mapeo de rutas si los archivos de datos de carga de trabajo incluyen referencias de ruta absolutas en lugar de relativas.
  - Elementos de plantillas de trabajo para paquetes de trabajos

Portabilidad de la carga de trabajo

Una carga de trabajo es portátil cuando puede ejecutarse en varios sistemas diferentes sin cambiarla cada vez que se envía un trabajo. Por ejemplo, puede ejecutarse en diferentes granjas de renderizado que tengan montados diferentes sistemas de archivos compartidos o en diferentes sistemas operativos, como Linux oWindows. Al implementar un paquete de tareas portátil, es más fácil para los usuarios ejecutar el trabajo en su granja específica o adaptarlo para otros casos de uso.

Estas son algunas maneras en las que puede hacer que su paquete de tareas sea portátil.

- Especifique completamente los archivos de datos de entrada que necesita una carga de trabajo, utilizando los parámetros del `PATH` trabajo y las referencias de activos del paquete de trabajos. Este enfoque hace que el trabajo sea transferible a granjas basadas en sistemas de archivos

compartidos y a granjas que hacen copias de los datos de entrada, como la función de adjuntar trabajos de Deadline Cloud.

- Haga que las referencias a las rutas de los archivos de entrada del trabajo sean reubicables y utilizables en diferentes sistemas operativos. Por ejemplo, cuando los usuarios envían trabajos desde Windows estaciones de trabajo para ejecutarlos en una flota. Linux
  - Utilice referencias relativas a las rutas de los archivos, de modo que si el directorio que las contiene se mueve a una ubicación diferente, las referencias seguirán resolviéndose. Algunas aplicaciones, como Blender, permiten elegir entre rutas relativas y absolutas.
  - Si no puedes usar rutas relativas, admite los metadatos de mapeo de rutas de OpenJD y traduce las rutas absolutas de acuerdo con la forma en que Deadline Cloud proporciona los archivos a la tarea.
- Implementa comandos en un trabajo mediante scripts portátiles. Python y bash son dos ejemplos de lenguajes de programación que se pueden usar de esta manera. Debería considerar la posibilidad de proporcionarlos a todos los anfitriones de trabajadores de sus flotas.
  - Utilice el intérprete de scripts binario, como `python` o`bash`, con el nombre del archivo de script como argumento. Este enfoque funciona en todos los sistemas operativosWindows, incluso en comparación con el uso de un archivo de script con el bit de ejecución activadoLinux.
  - Escribe scripts bash portátiles aplicando estas prácticas:
    - Expanda los parámetros de ruta de la plantilla entre comillas simples para gestionar las rutas con espacios y separadores de Windows rutas.
    - Cuando se ejecuteWindows, esté atento a los problemas relacionados con la traducción automática de rutas en MingW. Por ejemplo, transforma un AWS CLI comando similar `aws logs tail /aws/deadline/...` en un comando similar a un registro `aws logs tail "C:/Program Files/Git/aws/deadline/..."` y no lo guarda correctamente. Configura la variable `MSYS_NO_PATHCONV=1` para desactivar este comportamiento.
    - En la mayoría de los casos, el mismo código funciona en todos los sistemas operativos. Cuando sea necesario que el código sea diferente, utilice una `if/else` construcción para gestionar los casos.

```
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
```

  - Puede escribir scripts de Python portátiles `pathlib` para gestionar las diferencias en las rutas del sistema de archivos y evitar funciones operativas específicas. La documentación de Python incluye anotaciones para esto, por ejemplo, en la documentación de la biblioteca de señales. Linux-La compatibilidad con funciones específicas está marcada como «Disponibilidad: Linux».
- Utilice los parámetros del trabajo para especificar los requisitos de la aplicación. Utilice convenciones coherentes que el administrador de la granja pueda aplicar en los entornos de colas.
  - Por ejemplo, puede usar los `RezPackages` parámetros `CondaPackages` y/o en su trabajo, con un valor de parámetro predeterminado que muestre los nombres de los paquetes de aplicaciones y las versiones que requiere el trabajo. A continuación, puede utilizar uno de los entornos de cola de ejemplo de conda o Rez para proporcionar un entorno virtual para el trabajo.

Cómo empezar con los recursos de Deadline Cloud

Para empezar a crear soluciones personalizadas para AWS Deadline Cloud, debe configurar sus recursos. Estos incluyen una granja, al menos una cola para la granja y al menos una flota de trabajadores para atender la cola. Puede crear sus recursos mediante la consola de Deadline Cloud o puede utilizar la. AWS Command Line Interface

En este tutorial, los utilizarás AWS CloudShell para crear una granja de desarrolladores sencilla y ejecutar el agente de trabajo. A continuación, podrá enviar y ejecutar un trabajo sencillo con parámetros y adjuntos, añadir una flota gestionada por el servicio y limpiar los recursos de su granja cuando haya terminado.

En las siguientes secciones, se presentan las diferentes funciones de Deadline Cloud y cómo funcionan y funcionan juntas. Seguir estos pasos resulta útil para desarrollar y probar nuevas cargas de trabajo y personalizaciones.

Para obtener instrucciones sobre cómo configurar su granja mediante la consola, consulte Primeros pasos en la guía del usuario de Deadline Cloud.

Temas

- Cree una granja de Deadline Cloud
- Ejecute el agente de trabajo de Deadline Cloud
- Envía con Deadline Cloud
- Envíe trabajos con adjuntos de trabajo en Deadline Cloud
- Agrega una flota gestionada por servicios a tu granja de desarrolladores en Deadline Cloud
- Limpia los recursos de tu granja en Deadline Cloud

Cree una granja de Deadline Cloud

Para crear tu granja de desarrolladores y poner en cola los recursos en AWS Deadline Cloud, usa el AWS Command Line Interface (AWS CLI), como se muestra en el siguiente procedimiento. También crearás un rol AWS Identity and Access Management (IAM) y una flota gestionada por el cliente (CMF) y asociarás la flota a tu cola. A continuación, puede configurar la granja AWS CLI y confirmar que está configurada y funcionando según lo especificado.

Puedes usar esta granja para explorar las funciones de Deadline Cloud y, a continuación, desarrollar y probar nuevas cargas de trabajo, personalizaciones e integraciones de canalizaciones.

Para crear una granja

1. Abre una AWS CloudShell sesión. Utilizará la CloudShell ventana para introducir comandos AWS Command Line Interface (AWS CLI) para ejecutar los ejemplos de este tutorial. Mantén la CloudShell ventana abierta a medida que avanzas.
2. Cree un nombre para su granja y añada ese nombre a`~/.bashrc`. Esto hará que esté disponible para otras sesiones terminales.

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

3. Cree el recurso de granja y añada su ID de granja 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. Cree el recurso de cola y añada su ID de cola 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. Cree un rol de IAM para la flota. Esta función proporciona a los trabajadores anfitriones de su flota las credenciales de seguridad necesarias para ejecutar los trabajos desde su lista de espera.

```
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. Cree la flota gestionada por el cliente (CMF) y añada su ID de flota 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. Asocia la CMF a tu cola.

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

8. Instale la interfaz de línea de comandos de Deadline Cloud.

```
pip install deadline
```

9. Para establecer la granja predeterminada en el ID de granja y la cola en el ID de cola que creó anteriormente, utilice el siguiente comando.

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

10. (Opcional) Para confirmar que la granja está configurada de acuerdo con sus especificaciones, utilice los siguientes comandos:

    - Enumere todas las granjas: **`deadline farm list`**
    - Listar todas las colas de la granja predeterminada: **`deadline queue list`**
    - Enumere todas las flotas de la granja predeterminada: **`deadline fleet list`**
    - Obtenga la granja predeterminada: **`deadline farm get`**
    - Obtenga la cola predeterminada: **`deadline queue get`**
    - Obtenga todas las flotas asociadas a la cola predeterminada: **`deadline fleet get`**

Pasos a seguir a continuación

Tras crear tu granja, puedes ejecutar el agente de trabajo de Deadline Cloud en los hosts de tu flota para procesar los trabajos. Consulte Ejecute el agente de trabajo de Deadline Cloud.

Ejecute el agente de trabajo de Deadline Cloud

Para poder ejecutar los trabajos que envíes a la lista de espera de tu granja de desarrolladores, debes ejecutar el agente de trabajo de AWS Deadline Cloud en modo desarrollador en un host de trabajo.

Durante el resto de este tutorial, realizarás AWS CLI operaciones en tu granja de desarrolladores mediante dos AWS CloudShell pestañas. En la primera pestaña, puede enviar trabajos. En la segunda pestaña, puede ejecutar el agente de trabajo.

> **Note**
>
> Si deja la CloudShell sesión inactiva durante más de 20 minutos, se agotará el tiempo de espera y se detendrá al agente de trabajo. Para reiniciar el agente de trabajo, siga las instrucciones del siguiente procedimiento.

Antes de poder crear un agente de trabajo, debe configurar una granja, una cola y una flota de Deadline Cloud. Consulte Cree una granja de Deadline Cloud.

Para ejecutar el agente de trabajo en modo desarrollador

1. Con la granja aún abierta en la primera CloudShell pestaña, abre una segunda CloudShell pestaña y, a continuación, crea los `demoenv-persist` directorios `demoenv-logs` y.

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

2. Descarga e instala los paquetes de agentes de trabajo de Deadline Cloud desde PyPI:

   > **Note**
   >
   > Activado Windows, es necesario que los archivos del agente estén instalados en el directorio global de paquetes de sitios de Python. Los entornos virtuales de Python no son compatibles actualmente.

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

3. Para permitir que el agente de trabajo cree los directorios temporales para las tareas en ejecución, cree un directorio:

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

4. Ejecute el agente de trabajo de Deadline Cloud en modo desarrollador con `DEV_FARM_ID` las variables `DEV_CMF_ID` que haya agregado al`~/.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
```

A medida que el agente de trabajo inicializa y, a continuación, sondea la operación de la `UpdateWorkerSchedule` API, se muestra el siguiente resultado:

```
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. Seleccione la primera CloudShell pestaña y, a continuación, enumere los trabajadores de la flota.

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

Se muestra un resultado como el siguiente:

```
Displaying 1 of 1 workers starting at 0

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

En una configuración de producción, el agente de trabajo de Deadline Cloud requiere configurar varios usuarios y directorios de configuración como usuario administrativo en la máquina host. Puede anular esta configuración porque ejecuta los trabajos en su propia granja de desarrollo, a la que solo usted puede acceder.

Pasos a seguir a continuación

Ahora que hay un agente de trabajadores en los hosts de sus trabajadores, puede enviar los trabajos a sus trabajadores. Puede hacer lo siguiente:

- Envía con Deadline Cloudutilizando un sencillo paquete de trabajos de OpenJD.
- Envíe trabajos con adjuntos de trabajo en Deadline Cloudque comparten archivos entre estaciones de trabajo que utilizan diferentes sistemas operativos.

Envía con Deadline Cloud

Para ejecutar trabajos de Deadline Cloud en los hosts de sus trabajadores, cree y utilice un paquete de trabajos de Open Job Description (OpenJD) para configurar un trabajo. El paquete configura el trabajo, por ejemplo, especificando los archivos de entrada para un trabajo y dónde escribir el resultado del trabajo. En este tema se incluyen ejemplos de formas de configurar un paquete de trabajos.

Para poder seguir los procedimientos de esta sección, debe completar lo siguiente:

- Cree una granja de Deadline Cloud
- Ejecute el agente de trabajo de Deadline Cloud

Para usar AWS Deadline Cloud para ejecutar trabajos, utilice los siguientes procedimientos. Usa la primera AWS CloudShell pestaña para enviar trabajos a tu granja de desarrolladores. Utilice la segunda CloudShell pestaña para ver el resultado del agente obrero.

Temas

Envíe la simple_job muestra

Después de crear una granja y gestionar el agente obrero, puede enviar la simple_job muestra a Deadline Cloud.

Para enviar la simple_job muestra a Deadline Cloud

1. Elige tu primera CloudShell pestaña.
2. Descarga la muestra de GitHub.

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

3. Navegue hasta el directorio de ejemplos de paquetes de trabajos.

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

4. Envíe la simple_job muestra.

```
deadline bundle submit simple_job
```

5. Seleccione la segunda CloudShell pestaña para ver el resultado del registro sobre las llamadas`BatchGetJobEntities`, la obtención de una sesión y la ejecución de una acción de sesión.

```
...
[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**
>
> Solo se muestra el resultado del registro del agente de trabajo. Hay un registro independiente para la sesión en la que se ejecuta el trabajo.

6. Elija la primera pestaña y, a continuación, inspeccione los archivos de registro que escribe el agente de trabajo.

   a. Navegue hasta el directorio de registros del agente de trabajo y vea su contenido.

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

   b. Imprima el primer archivo de registro que cree el agente de trabajo.

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

   Este archivo contiene información sobre cómo llamó a la API de Deadline Cloud para crear un recurso de trabajadores en su flota y, después, asumió la función de flota.

   c. Imprima el resultado del archivo de registro cuando el agente obrero se una a la flota.

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

   Este registro contiene resultados sobre todas las acciones que realiza el agente trabajador, pero no contiene resultados sobre las colas desde las que ejecuta los trabajos, excepto en lo que respecta a esos recursos. IDs

   d. Imprima los archivos de registro de cada sesión en un directorio que tenga el mismo nombre que el identificador del recurso de la cola.

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

   Si el trabajo se realiza correctamente, la salida del archivo de registro será similar a la siguiente:

   ```
   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. Imprima la información sobre el trabajo.

   ```
   deadline job get
   ```

   Al enviar el trabajo, el sistema lo guarda como predeterminado para que no tenga que introducir el identificador del trabajo.

Envíe un anuncio simple_job con un parámetro

Puede enviar trabajos con parámetros. En el siguiente procedimiento, edite la simple_job plantilla para incluir un mensaje personalizado, envíe el simple_job archivo de registro de la sesión e imprima el mismo para ver el mensaje.

Para enviar el simple_job ejemplo con un parámetro

1. Seleccione la primera CloudShell pestaña y, a continuación, navegue hasta el directorio de ejemplos de paquetes de trabajos.

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

2. Imprima el contenido de la simple_job plantilla.

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

La `parameterDefinitions` sección con el `Message` parámetro debería tener el siguiente aspecto:

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

3. Envíe la simple_job muestra con un valor de parámetro y espere a que el trabajo termine de ejecutarse.

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

4. Para ver el mensaje personalizado, consulte el archivo de registro de sesión más reciente.

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

Cree un paquete de trabajos simple_file_job con E/S de archivos

Un trabajo de renderizado necesita leer la definición de la escena, renderizar una imagen a partir de ella y, a continuación, guardar esa imagen en un archivo de salida. Puede simular esta acción haciendo que el trabajo calcule el hash de la entrada en lugar de renderizar una imagen.

Para crear un paquete de trabajos simple_file_job con E/S de archivos

1. Seleccione la primera CloudShell pestaña y, a continuación, navegue hasta el directorio de ejemplos del paquete de trabajos.

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

2. Haga una copia de `simple_job` con el nuevo nombre`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Edite la plantilla de trabajo de la siguiente manera:

> **Note**
>
> Le recomendamos que utilice nano estos pasos. Si prefiere usarloVim, debe configurar su modo de pegado usando`:set paste`.

a. Abre la plantilla en un editor de texto.

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

b. Añada lo siguiente `typeobjectType`, y `dataFlowparameterDefinitions`.

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

c. Añada el siguiente comando de `bash` script al final del archivo para leer el archivo de entrada y escribir en el archivo de salida.

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

La actualización `template.yaml` debe coincidir exactamente con lo siguiente:

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

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

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

> **Note**
>
> Si desea ajustar el espaciado`template.yaml`, asegúrese de utilizar espacios en lugar de hendiduras.

d. Guarde el archivo y salga del editor de texto.

4. Proporcione los valores de los parámetros de los archivos de entrada y salida para enviar el simple_file_job.

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

5. Imprima la información sobre el trabajo.

```
deadline job get
```

- Verá un resultado como el siguiente:

```
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
```

- Aunque solo proporcionó rutas relativas, los parámetros tienen configurada la ruta completa. AWS CLI Une el directorio de trabajo actual a cualquier ruta que se proporcione como parámetro cuando las rutas tienen ese tipoPATH.
- El agente de trabajo que se encuentra en la otra ventana de la terminal recoge y ejecuta el trabajo. Esta acción crea el `hash.txt` archivo, que puede ver con el siguiente comando.

  ```
  cat hash.txt
  ```

  Este comando imprimirá un resultado similar al siguiente.

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

Siguientes pasos

Después de aprender a enviar trabajos sencillos mediante la CLI de Deadline Cloud, puede explorar:

- Envíe trabajos con adjuntos de trabajo en Deadline Cloudpara obtener información sobre cómo ejecutar trabajos en hosts que ejecutan diferentes sistemas operativos.
- Agrega una flota gestionada por servicios a tu granja de desarrolladores en Deadline Cloudpara ejecutar tus trabajos en hosts gestionados por Deadline Cloud.
- Limpia los recursos de tu granja en Deadline Cloudpara cerrar los recursos que utilizaste para este tutorial.

Envíe trabajos con adjuntos de trabajo en Deadline Cloud

Muchas granjas utilizan sistemas de archivos compartidos para compartir archivos entre los anfitriones que envían los trabajos y los que los ejecutan. Por ejemplo, en el `simple_file_job` ejemplo anterior, el sistema de archivos local se comparte entre las ventanas de AWS CloudShell

terminal, que se encuentran en la pestaña uno, donde se envía el trabajo, y en la pestaña dos, donde se ejecuta el agente de trabajo.

Un sistema de archivos compartido es ventajoso cuando la estación de trabajo remitente y los hosts de trabajo se encuentran en la misma red de área local. Si almacena los datos de forma local, cerca de las estaciones de trabajo que acceden a ellos, si utiliza una granja de servidores basada en la nube, tendrá que compartir sus sistemas de archivos a través de una VPN de alta latencia o sincronizarlos en la nube. Ninguna de estas opciones es fácil de configurar ni utilizar.

AWS Deadline Cloud ofrece una solución sencilla con archivos adjuntos de trabajo, que son similares a los archivos adjuntos de correo electrónico. Con los archivos adjuntos de trabajo, puede adjuntar datos a su trabajo. A continuación, Deadline Cloud gestiona los detalles de la transferencia y el almacenamiento de los datos de su trabajo en los depósitos de Amazon Simple Storage Service (Amazon S3).

Los flujos de trabajo de creación de contenido suelen ser iterativos, lo que significa que un usuario envía los trabajos con un pequeño subconjunto de archivos modificados. Como los buckets de Amazon S3 almacenan los adjuntos de los trabajos en un almacenamiento direccionable por contenido, el nombre de cada objeto se basa en el hash de los datos del objeto y el contenido de un árbol de directorios se almacena en un formato de archivo de manifiesto adjunto a un trabajo.

Para poder seguir los procedimientos de esta sección, debe completar lo siguiente:

- Cree una granja de Deadline Cloud
- Ejecute el agente de trabajo de Deadline Cloud

Para ejecutar trabajos con trabajos adjuntos, complete los siguientes pasos.

Temas

Añada una configuración de adjuntos de trabajos a su cola

Para habilitar los adjuntos de trabajos en su cola, añada una configuración de adjuntos de trabajos al recurso de cola de su cuenta.

Para añadir una configuración de adjuntos de trabajos a su cola

1. Seleccione la primera CloudShell pestaña y, a continuación, introduzca uno de los siguientes comandos para usar un bucket de Amazon S3 para adjuntar trabajos.
   - Si no tiene un bucket privado de Amazon S3 existente, puede crear y usar un bucket S3 nuevo.

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

   - Si ya tienes un bucket privado de Amazon S3, puedes usarlo *MY_BUCKET_NAME* sustituyéndolo por el nombre de tu bucket.

     ```
     DEV_FARM_BUCKET=MY_BUCKET_NAME
     ```

2. Después de crear o elegir su bucket de Amazon S3, añada el nombre del bucket `~/.bashrc` para que esté disponible para otras sesiones de terminal.

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

3. Cree un rol AWS Identity and Access Management (de IAM) para la cola.

   ```
   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. Actualice la cola para incluir la configuración de los adjuntos de trabajos y la función de 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. Confirme que ha actualizado la cola.

```
deadline queue get
```

Se muestra un resultado como el siguiente:

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

Enviar simple_file_job con adjuntos de trabajo

Cuando utilizas adjuntos de trabajo, los paquetes de trabajos deben proporcionar a Deadline Cloud suficiente información para determinar el flujo de datos del trabajo, por ejemplo, mediante PATH parámetros. En el caso desimple_file_job, editaste el `template.yaml` archivo para indicar a Deadline Cloud que el flujo de datos está en el archivo de entrada y en el archivo de salida.

Una vez que hayas añadido la configuración de adjuntos de trabajos a tu lista, puedes enviar el ejemplo de simple_file_job con los adjuntos de trabajo. Una vez hecho esto, podrá ver el registro y el resultado de los trabajos para confirmar que el archivo con los trabajos adjuntos funciona. simple_file_job

Para enviar el paquete de trabajos simple_file_job con los trabajos adjuntos

1. Elija la primera CloudShell pestaña y, a continuación, abra el directorio. `JobBundle-Samples`
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```

3. Envía simple_file_job a la lista de espera. Cuando se te pida que confirmes la carga, ingresa. **y**

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

4. Para ver el resultado del registro de la sesión de transferencia de datos de los archivos adjuntos al trabajo, ejecute el siguiente comando.

```
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. Enumere las acciones de la sesión que se ejecutaron dentro de la sesión.

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

Se muestra un resultado como el siguiente:

```
{
    "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 primera acción de la sesión descargó los adjuntos del trabajo de entrada, mientras que la segunda acción ejecuta la tarea como en los pasos anteriores y, a continuación, carga los adjuntos del trabajo de salida.

6. Enumere el directorio de salida.

```
ls *.txt
```

Este tipo de salida `hash.txt` existe en el directorio, pero `hash-jobattachments.txt` no existe porque el archivo de salida del trabajo aún no se ha descargado.

7. Descarga el resultado del trabajo más reciente.

```
deadline job download-output
```

8. Vea el resultado del archivo descargado.

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

Se muestra un resultado como el siguiente:

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

Entender cómo se almacenan los archivos adjuntos de trabajo en Amazon S3

Puede usar AWS Command Line Interface (AWS CLI) para cargar o descargar datos para adjuntos de trabajos, que se almacenan en depósitos de Amazon S3. Comprender cómo Deadline Cloud almacena los adjuntos de trabajo en Amazon S3 le ayudará a desarrollar integraciones de cargas de trabajo y canalizaciones.

Para inspeccionar cómo se almacenan los adjuntos de trabajo de Deadline Cloud en Amazon S3

1. Elija la primera CloudShell pestaña y, a continuación, abra el directorio de ejemplos de paquetes de trabajos.

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

2. Inspeccione las propiedades del trabajo.

   ```
   deadline job get
   ```

   Se muestra un resultado como el siguiente:

   ```
   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
```

El campo de adjuntos contiene una lista de estructuras de manifiesto que describen las rutas de datos de entrada y salida que utiliza el trabajo cuando se ejecuta. Observe `rootPath` la ruta del directorio local de la máquina que envió el trabajo. Para ver el sufijo de objeto de Amazon S3 que contiene un archivo de manifiesto, consulte la`inputManifestFile`. El archivo de manifiesto contiene metadatos para una instantánea del árbol de directorios de los datos de entrada del trabajo.

3. Imprima de forma bonita el objeto del manifiesto de Amazon S3 para ver la estructura de directorios de entrada del trabajo.

```
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 .
```

Se muestra un resultado como el siguiente:

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

4. Cree el prefijo Amazon S3 que contiene los manifiestos de los adjuntos de los trabajos de salida y enumere el objeto debajo de él.

```
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
```

No se hace referencia directamente a los adjuntos del trabajo de salida desde el recurso del trabajo, sino que se colocan en un bucket de Amazon S3 en función del recurso de la granja IDs.

5. Obtenga la clave de objeto de manifiesto más reciente para el identificador de acción de sesión específico y, a continuación, imprima de forma bonita los objetos del manifiesto.

```
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 .
```

Verás las propiedades del archivo `hash-jobattachments.txt` en el resultado, como las siguientes:

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

Tu trabajo solo tendrá un objeto de manifiesto por cada tarea que se ejecute, pero en general es posible tener más objetos por tarea ejecutada.

6. Vea la salida de almacenamiento de Amazon S3 direccionable por contenido bajo el prefijo. `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 -
```

Se muestra un resultado como el siguiente:

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

Siguientes pasos

Después de aprender a enviar trabajos con archivos adjuntos mediante la CLI de Deadline Cloud, puede explorar:

- Envía con Deadline Cloudpara aprender a ejecutar trabajos con un paquete de OpenJD en los hosts de sus trabajadores.
- Agrega una flota gestionada por servicios a tu granja de desarrolladores en Deadline Cloudpara ejecutar tus trabajos en hosts gestionados por Deadline Cloud.
- Limpia los recursos de tu granja en Deadline Cloudpara cerrar los recursos que utilizaste para este tutorial.

Agrega una flota gestionada por servicios a tu granja de desarrolladores en Deadline Cloud

AWS CloudShell no proporciona suficiente capacidad de cómputo para probar cargas de trabajo más grandes. Tampoco está configurado para funcionar con trabajos que distribuyen las tareas en varios hosts de trabajo.

En lugar de utilizarla CloudShell, puede añadir una flota gestionada por servicios (SMF) de Auto Scaling a su granja de desarrolladores. Un SMF proporciona suficiente capacidad de cómputo para cargas de trabajo más grandes y puede gestionar trabajos que necesiten distribuirse entre varios hosts de trabajo.

Antes de añadir una SMF, debe configurar una granja, una cola y una flota de Deadline Cloud. Consulte Cree una granja de Deadline Cloud.

Para añadir una flota gestionada por servicios a tu granja de desarrolladores

1. Selecciona la primera AWS CloudShell pestaña y, a continuación, crea la flota gestionada por el servicio y añade su ID de flota a. `.bashrc` Esta acción hace que esté disponible para otras sesiones de terminal.

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

 echo "DEV_SMF_ID=$(aws deadline list-fleets \
```

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

2. Asocie el SMF a su cola.

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

3. Envíelo simple_file_job a la cola. Cuando se te pida que confirmes la carga, ingresa**y**.

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

4. Confirme que el SMF funciona correctamente.

```
deadline fleet get
```

   - El trabajador puede tardar unos minutos en empezar. Repita el `deadline fleet get` comando hasta que vea que la flota está funcionando.
   - La flota gestionada `queueFleetAssociationsStatus` por el servicio será. `ACTIVE`
   - El SMF `autoScalingStatus` cambiará de a. `GROWING` `STEADY`

   Su estado será similar al siguiente:

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. Vea el registro del trabajo que envió. Este registro se guarda en un registro de Amazon CloudWatch Logs, no en el sistema de CloudShell archivos.

```
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
```

Siguientes pasos

Tras crear y probar una flota gestionada por un servicio, debe eliminar los recursos que haya creado para evitar cargos innecesarios.

- Limpia los recursos de tu granja en Deadline Cloudpara cerrar los recursos que utilizaste en este tutorial.

Limpia los recursos de tu granja en Deadline Cloud

Para desarrollar y probar nuevas cargas de trabajo e integraciones de canalizaciones, puedes seguir utilizando la granja de desarrolladores de Deadline Cloud que creaste para este tutorial. Si ya no necesitas tu granja de desarrolladores, puedes eliminar sus recursos, incluidos la granja, la flota, la cola, las funciones AWS Identity and Access Management (IAM) y los registros de Amazon CloudWatch Logs. Tras eliminar estos recursos, tendrá que volver a empezar el tutorial para poder utilizarlos. Para obtener más información, consulte Cómo empezar con los recursos de Deadline Cloud.

Para limpiar los recursos de la granja de desarrolladores

1. Elige la primera CloudShell pestaña y, a continuación, detiene todas las asociaciones de flotas que forman parte de tu lista.

```
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. Haz una lista de las asociaciones de flotas en cola.

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

Puede que tenga que volver a ejecutar el comando hasta que se muestre el resultado y`"status": "STOPPED"`, a continuación, puede continuar con el siguiente paso. Este proceso puede tardar varios minutos en completarse.

```
{
    "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. Elimine todas las asociaciones de colas y flotas de su cola.

```
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 todas las flotas asociadas a tu cola.

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

5. Elimine la cola.

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

6. Elimine la granja.

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

7. Elimina otros AWS recursos de tu granja.

   a. Elimine el rol de flota 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. Elimine la función de IAM de cola.

```
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. Elimine los grupos de CloudWatch registros de Amazon Logs. Cada cola y flota tiene su propio grupo de registros.

```
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"
```

Cree trabajos para enviarlos a Deadline Cloud

Los trabajos se envían a Deadline Cloud mediante paquetes de trabajos. Un paquete de trabajos es una colección de archivos que incluye una plantilla de trabajo de Open Job Description (OpenJD) y cualquier archivo de activos necesario para renderizar el trabajo.

La plantilla de trabajo describe cómo los trabajadores procesan los activos y acceden a ellos, y proporciona el script que ejecuta el trabajador. Los paquetes de trabajo permiten a los artistas, directores técnicos y desarrolladores de proyectos enviar fácilmente trabajos complejos a Deadline Cloud desde sus estaciones de trabajo locales o desde una granja de renderizados local. Los paquetes de tareas son especialmente útiles para los equipos que trabajan en proyectos de efectos visuales, animaciones u otros proyectos de renderización multimedia a gran escala que requieren recursos informáticos escalables y bajo demanda.

Puede crear el paquete de tareas mediante el sistema de archivos local para almacenar los archivos y un editor de texto para crear la plantilla de tareas. Después de crear el paquete, envía el trabajo a Deadline Cloud mediante la CLI de Deadline Cloud o una herramienta como un remitente de Deadline Cloud

Puedes almacenar tus activos en un sistema de archivos compartido entre tus trabajadores, o puedes usar los archivos adjuntos de trabajo de Deadline Cloud para automatizar el traslado de los activos a depósitos de S3 donde tus trabajadores puedan acceder a ellos. Los adjuntos de trabajo también ayudan a mover la salida de sus trabajos a sus estaciones de trabajo.

En las siguientes secciones se proporcionan instrucciones detalladas sobre cómo crear y enviar paquetes de trabajos a Deadline Cloud.

Temas

Plantillas Open Job Description (OpenJD) para Deadline Cloud

Un paquete de trabajos es una de las herramientas que se utilizan para definir los trabajos en AWS Deadline Cloud. Agrupan una plantilla de Open Job Description (OpenJD) con información adicional, como archivos y directorios que utilizan sus trabajos con adjuntos de trabajo. Utiliza la interfaz de línea de comandos (CLI) de Deadline Cloud para usar un paquete de trabajos para enviar trabajos para que se ejecute una cola.

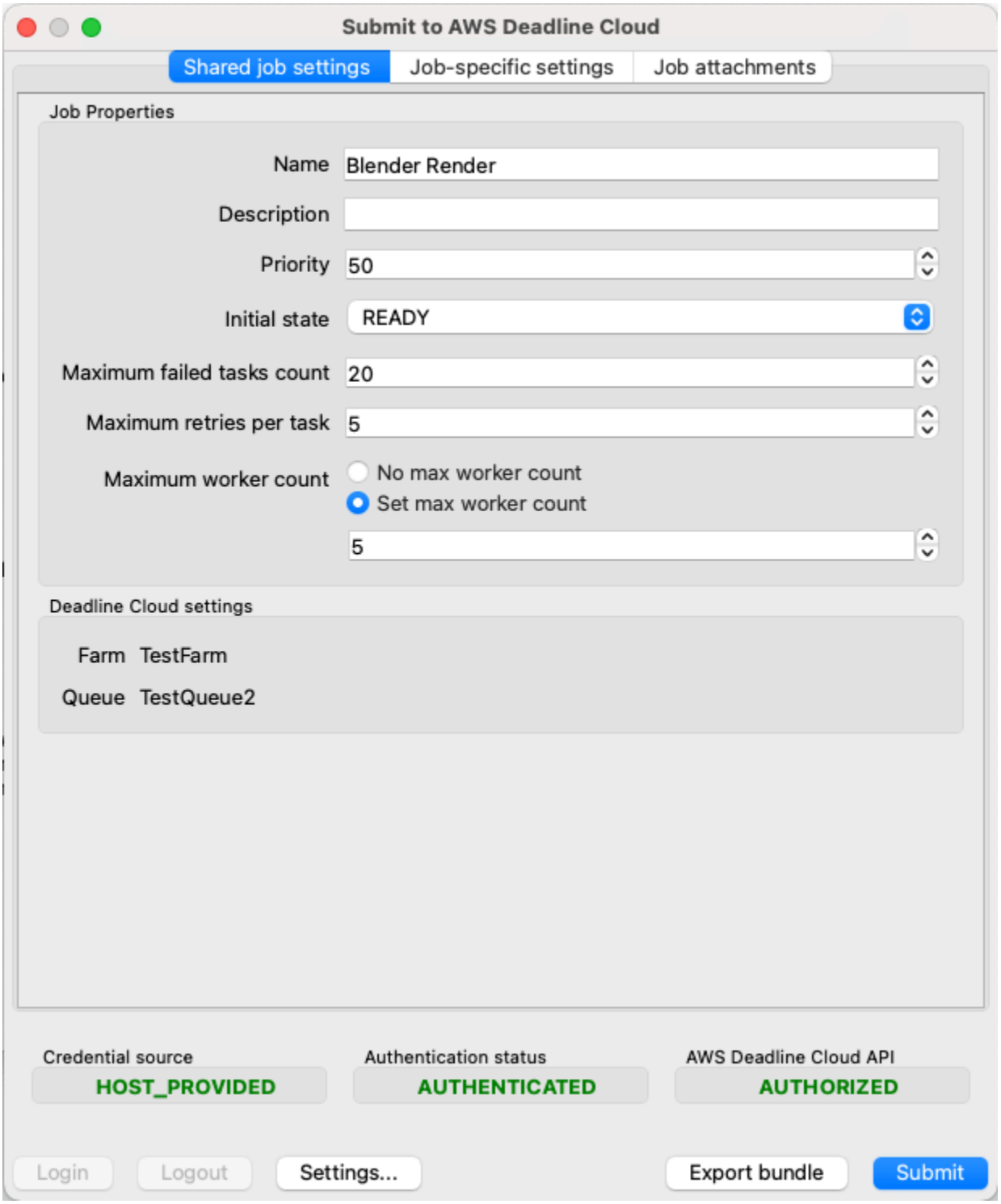
Un paquete de trabajos es una estructura de directorios que contiene una plantilla de trabajo de OpenJD, otros archivos que definen el trabajo y archivos específicos del trabajo necesarios como entrada para el trabajo. Puede especificar los archivos que definen su trabajo como archivos YAML o JSON.

El único archivo obligatorio es `template.yaml` `otemplate.json`. También puede incluir los siguientes archivos:

```
/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
```

Usa un paquete de trabajos para enviar trabajos personalizados con la CLI de Deadline Cloud y un adjunto de trabajo, o puedes usar una interfaz gráfica de envío. Por ejemplo, el siguiente es el ejemplo de Blender de GitHub. Para ejecutar el ejemplo, usa el siguiente comando en el directorio de ejemplos de Blender:

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



El panel de ajustes específicos del trabajo se genera a partir de las `userInterface` propiedades de los parámetros del trabajo definidos en la plantilla del trabajo.

Para enviar un trabajo mediante la línea de comandos, puede utilizar un comando similar al siguiente

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

O puede usar la `deadline.client.api.create_job_from_job_bundle` función en el paquete `deadline` Python.

Todos los complementos de envío de trabajos incluidos en Deadline Cloud, como el complemento Autodesk Maya, generan un paquete de trabajos para su envío y, a continuación, utilizan el paquete Python de Deadline Cloud para enviar su trabajo a Deadline Cloud. Puede ver los paquetes de trabajos enviados en el directorio del historial de trabajos de su estación de trabajo o mediante un remitente. Puede encontrar el directorio del historial de trabajos con el siguiente comando:

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

Cuando tu trabajo se ejecuta en un trabajador de Deadline Cloud, este tiene acceso a variables de entorno que le proporcionan información sobre el trabajo. Las variables de entorno son:

Nombre de variable	Disponible
DEADLINE_FARM_ID	Todas las acciones
DEADLINE_FLEET_ID	Todas las acciones
DEADLINE_WORKER_ID	Todas las acciones
DEADLINE_QUEUE_ID	Todas las acciones
DEADLINE_JOB_ID	Todas las acciones
DEADLINE_STEP_ID	Acciones de tareas
DEADLINE_SESSION_ID	Todas las acciones
DEADLINE_TASK_ID	Acciones de tareas
DEADLINE_SESSIONACTION_ID	Todas las acciones

Temas

Elementos de plantillas de trabajo para paquetes de trabajos

La plantilla de trabajo define el entorno de ejecución y los procesos que se ejecutan como parte de un trabajo de Deadline Cloud. Puede crear parámetros en una plantilla para utilizarla para crear trabajos que solo difieran en los valores de entrada, como ocurre con una función en un lenguaje de programación.

Cuando envías un trabajo a Deadline Cloud, se ejecuta en cualquier entorno de cola aplicado a la cola. Los entornos de colas se crean utilizando la especificación de entornos externos Open Job Description (OpenJD). Para obtener más información, consulte la plantilla de entorno en el repositorio de GitHub OpenJD.

Para ver una introducción a la creación de un trabajo con una plantilla de trabajo de OpenJD, consulte Introducción a la creación de un trabajo en el repositorio de GitHub OpenJD. Puede encontrar más información en Cómo se ejecutan los trabajos. Hay ejemplos de plantillas de trabajo en el `samples` directorio del GitHub repositorio de OpenJD.

Puede definir la plantilla de trabajo en formato YAML (`template.yaml`) o en formato JSON (`template.json`). Los ejemplos de esta sección se muestran en formato YAML.

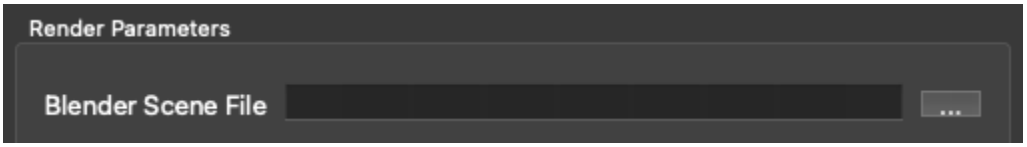
Por ejemplo, la plantilla de trabajo del `blender_render` ejemplo define un parámetro de entrada `BlenderSceneFile` como una ruta de archivo:

```
- 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` propiedad define el comportamiento de las interfaces de usuario generadas automáticamente tanto en la línea de comandos mediante el `deadline bundle gui-submit` comando como en los complementos de envío de trabajos para aplicaciones como Autodesk Maya.

En este ejemplo, el widget de interfaz de usuario para introducir un valor para el `BlenderSceneFile` parámetro es un cuadro de diálogo de selección de archivos que muestra solo los archivos. `.blend`



Para ver más ejemplos del uso del `userInteface` elemento, consulta el ejemplo gui_control_showcase en el repositorio de. deadline-cloud-samples GitHub

`dataFlow`Las propiedades `objectType` y controlan el comportamiento de los adjuntos de trabajo cuando se envía un trabajo desde un paquete de trabajos. En este caso, `objectType: FILE` y `dataFlow:IN` significan que el valor de `BlenderSceneFile` es un archivo de entrada para los trabajos adjuntos.

Por el contrario, la definición del `OutputDir` parámetro tiene `objectType: DIRECTORY` y`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.
```

Los adjuntos del trabajo utilizan el valor del `OutputDir` parámetro como directorio en el que el trabajo escribe los archivos de salida.

Para obtener más información sobre las `dataFlow` propiedades `objectType` y, consulte JobPathParameterDefinitionla especificación Open Job Description

El resto del ejemplo de plantilla de `blender_render` trabajo define el flujo de trabajo como un solo paso, en el que cada fotograma de la animación se representa como una tarea independiente:

```
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}}
```

Por ejemplo, si el valor del `Frames` parámetro es`1-10`, define 10 tareas. Cada una de las tareas tiene un valor diferente para el `Frame` parámetro. Para ejecutar una tarea:

1. Por ejemplo, se expanden todas las referencias a variables de la `data` propiedad del archivo incrustado`--render-frame 1`.
2. El contenido de la `data` propiedad se escribe en un archivo del directorio de trabajo de la sesión en el disco.
3. El onRun comando de la tarea se resuelve en `bash` *`location of embedded file`* y, a continuación, se ejecuta.

Para obtener más información sobre los archivos incrustados, las sesiones y las ubicaciones con mapas de rutas, consulte Cómo se ejecutan los trabajos en la especificación Open Job Description.

Hay más ejemplos de plantillas de trabajo en el repositorio deadline-cloud-samples/job_bundles, así como los ejemplos de plantillas que se proporcionan con la especificación Open Job Descriptions.

Fragmentación de tareas para plantillas de trabajo

La fragmentación de tareas te permite agrupar varias tareas en una sola unidad de trabajo denominada fragmento. En un trabajo de renderizado, por ejemplo, esto significa que Deadline Cloud puede enviar varios fotogramas juntos en lugar de un fotograma por cada invocación de comando. Esto reduce la sobrecarga que supone iniciar las aplicaciones para cada tarea y acorta el tiempo total de ejecución del trabajo. Para obtener más información, consulte Ejecutar varios fotogramas a la vez en la wiki de OpenJD.

OpenJD admite extensiones que añaden funciones opcionales a las plantillas de trabajo. La fragmentación de tareas se habilita añadiendo la `TASK_CHUNKING` extensión. Para utilizar la fragmentación, añada la extensión a la plantilla de trabajo y utilice el tipo de parámetro de `CHUNK[INT]` tarea. Envíe los trabajos fragmentados con el mismo `deadline bundle submit` comando. Por ejemplo, la siguiente plantilla de trabajo representa los marcos en bloques de 10:

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

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

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

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

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

En este ejemplo, Deadline Cloud divide los 100 fotogramas en fragmentos, por ejemplo `1-10` `11-20`, y así sucesivamente. La `{{Task.Param.Frame}}` variable se expande a una expresión de rango

como `1-10`. Como `rangeConstraint` está establecida en`CONTIGUOUS`, el rango siempre está en `start-end` formato. El script analiza este rango y pasa los fotogramas inicial y final a Blender usando las `-e` opciones `-s` y con`--render-anim`.

La `chunks` propiedad admite los siguientes campos:

- `defaultTaskCount`— (Obligatorio) Cuántas tareas se van a combinar en un solo bloque. El valor máximo es 150.
- `rangeConstraint`— (Obligatorio) Si`CONTIGUOUS`, un fragmento es siempre un rango contiguo, como. `1-10` Si`NONCONTIGUOUS`, un fragmento puede ser un conjunto arbitrario, como. `1,3,7-10`
- `targetRuntimeSeconds`— (Opcional) El tiempo de ejecución objetivo en segundos para cada fragmento. Deadline Cloud puede ajustar dinámicamente el tamaño de los fragmentos para acercarse a este objetivo una vez que se hayan completado algunos fragmentos.

Para ver más ejemplos de fragmentación de tareas, incluidos ejemplos básicos y de Blender con fragmentos contiguos y no contiguos, consulta los ejemplos de fragmentación de tareas en el repositorio de muestras de Deadline Cloud en. GitHub

> **Requisitos de flota gestionados por el cliente**
>
> La fragmentación de tareas requiere una versión de agente de trabajo compatible. Si utilizas flotas gestionadas por el cliente, asegúrate de que tus agentes de trabajo estén actualizados antes de enviar los trabajos con fragmentación. Las flotas gestionadas por el servicio siempre utilizan una versión de agente de trabajo compatible.

> **¿Descargando los resultados de los trabajos fragmentados**
>
> Cuando descargas el resultado de una sola tarea de un trabajo fragmentado, Deadline Cloud descarga el resultado de todo el fragmento. Por ejemplo, si los fotogramas del 1 al 10 se procesaron juntos, la descarga del resultado del fotograma 3 incluye todos los fotogramas del 1 al 10. Esta función requiere la `deadline-cloud` versión 0.53.3 o posterior.

Elementos de valores de parámetros para paquetes de trabajos

Puede usar el archivo de parámetros para establecer los valores de algunos de los parámetros del trabajo en la plantilla de trabajo o los argumentos de la solicitud de CreateJoboperación del paquete de trabajos, de modo que no necesite establecer valores al enviar un trabajo. La interfaz de usuario para el envío de trabajos le permite modificar estos valores.

Puede definir la plantilla de trabajo en formato YAML (`parameter_values.yaml`) o formato JSON (`parameter_values.json`). Los ejemplos de esta sección se muestran en formato YAML.

En YAML, el formato del archivo es:

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

Cada elemento de la `parameterValues` lista debe ser uno de los siguientes:

- Un parámetro de trabajo definido en la plantilla de trabajo.
- Un parámetro de trabajo definido en un entorno de colas para la cola a la que se envía el trabajo.
- Parámetro especial que se transfiere a la `CreateJob` operación al crear un trabajo.
  - `deadline:priority`— El valor debe ser un número entero. Se pasa a la `CreateJob` operación como parámetro de prioridad.
  - `deadline:targetTaskRunStatus`— El valor debe ser una cadena. Se pasa a la `CreateJob` operación como parámetro de targetTaskRunestado.
  - `deadline:maxFailedTasksCount`— El valor debe ser un número entero. Se pasa a la `CreateJob` operación como parámetro maxFailedTasksCount.
  - `deadline:maxRetriesPerTask`— El valor debe ser un número entero. Se pasa a la `CreateJob` operación como parámetro de maxRetriesPertarea.
  - `deadline:maxWorkercount`— El valor debe ser un número entero. Se pasa a la `CreateJob` operación como maxWorkerCountparámetro.

Una plantilla de trabajo es siempre una plantilla y no un trabajo específico que ejecutar. Un archivo de valores de parámetros permite que un paquete de trabajos actúe como plantilla si algunos

parámetros no tienen valores definidos en este archivo, o como un envío de trabajo específico si todos los parámetros tienen valores.

Por ejemplo, el ejemplo de blender_render no tiene un archivo de parámetros y su plantilla de trabajo define parámetros sin valores predeterminados. Esta plantilla debe usarse como plantilla para crear trabajos. Después de crear un trabajo con este paquete de trabajos, Deadline Cloud escribe un nuevo paquete de trabajos en el directorio del historial de trabajos.

Por ejemplo, cuando envías un trabajo con el siguiente comando:

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

El nuevo paquete de trabajos contiene un `parameter_values.yaml` archivo que contiene los parámetros especificados:

```
% 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
```

Puede crear el mismo trabajo con el siguiente comando:

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

> **Note**
>
> El paquete de trabajos que envíe se guarda en el directorio del historial de trabajos. Puede encontrar la ubicación de ese directorio con el siguiente comando:
>
> ```
> deadline config get settings.job_history_dir
> ```

Elementos de referencia de activos para paquetes de trabajos

Puedes usar los archivos adjuntos de trabajo de Deadline Cloud para transferir archivos de un lado a otro entre tu estación de trabajo y Deadline Cloud. El archivo de referencia de activos incluye los archivos y directorios de entrada, así como los directorios de salida para tus archivos adjuntos. Si no incluye todos los archivos y directorios de este archivo, puede seleccionarlos al enviar un trabajo con el `deadline bundle gui-submit` comando.

Este archivo no tiene ningún efecto si no utiliza adjuntos de trabajo.

Puede definir la plantilla de trabajo en formato YAML (`asset_references.yaml`) o en formato JSON (`asset_references.json`). Los ejemplos de esta sección se muestran en formato YAML.

En YAML, el formato del archivo es:

```
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
```

Al seleccionar el archivo de entrada o salida para cargarlo en Amazon S3, Deadline Cloud compara la ruta del archivo con las rutas que aparecen en sus perfiles de almacenamiento. Cada ubicación `SHARED` de sistema de archivos de un perfil de almacenamiento abstrae un recurso compartido de archivos de red que está montado en sus estaciones de trabajo y en los hosts de los trabajadores. Deadline Cloud carga solo los archivos que no se encuentran en uno de estos recursos compartidos de archivos.

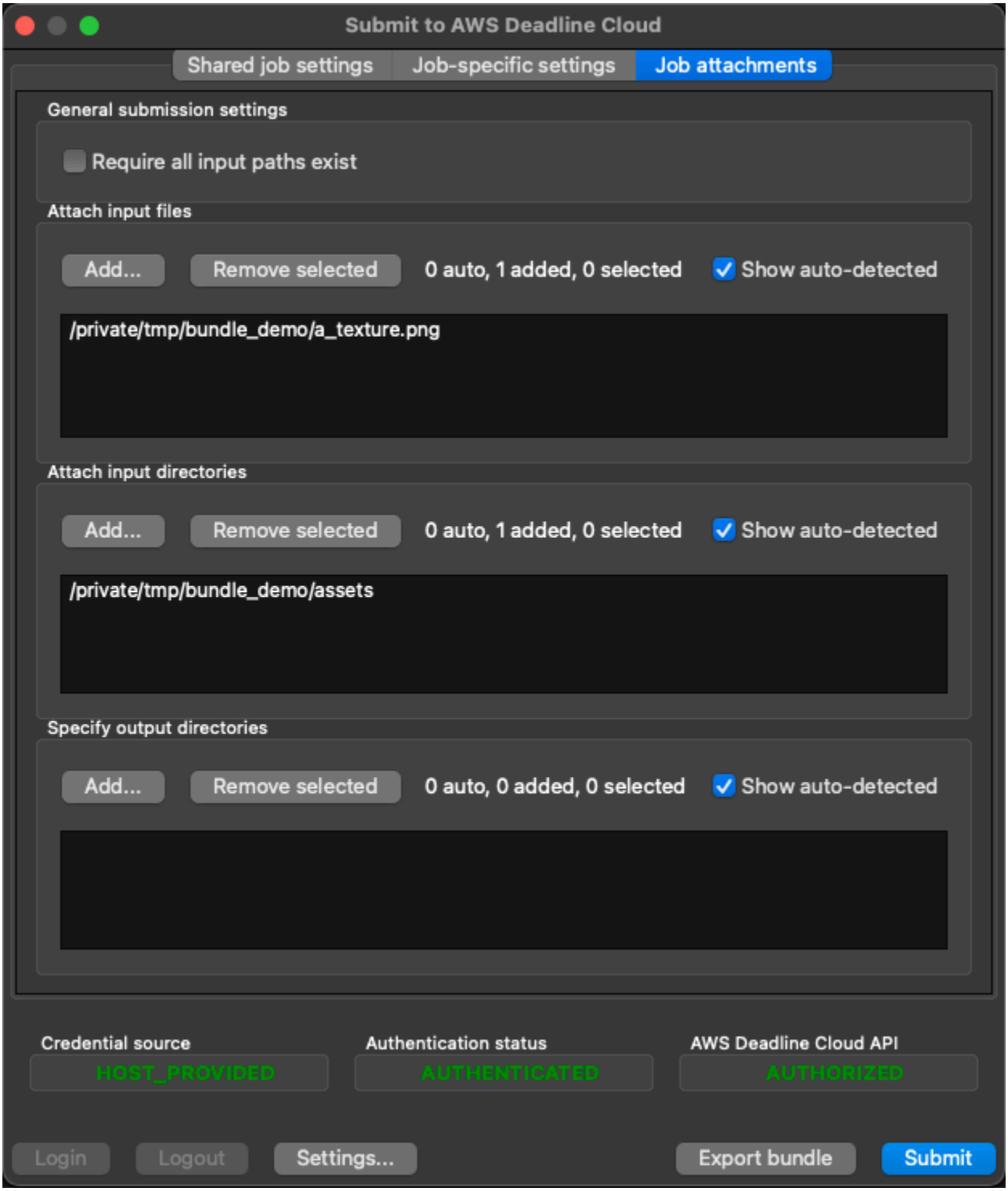
Para obtener más información sobre la creación y el uso de perfiles de almacenamiento, consulte Almacenamiento compartido en Deadline Cloud en la Guía del usuario de AWS Deadline Cloud.

Example- El archivo de referencia de activos creado por la GUI de Deadline Cloud

Usa el siguiente comando para enviar un trabajo usando el ejemplo de blender_render.

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

Añada algunos archivos adicionales al trabajo en la pestaña Adjuntos del trabajo:



Después de enviar el trabajo, puede consultar el `asset_references.yaml` archivo del paquete de trabajos en el directorio del historial de trabajos para ver los activos del archivo 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: []
```

Uso de archivos en sus trabajos

Muchos de los trabajos que envías a AWS Deadline Cloud tienen archivos de entrada y salida. Los archivos de entrada y los directorios de salida pueden estar ubicados en una combinación de sistemas de archivos compartidos y unidades locales. Los trabajos deben ubicar el contenido en esas ubicaciones. Deadline Cloud ofrece dos funciones, los archivos adjuntos de los trabajos y los perfiles de almacenamiento que funcionan en conjunto para ayudar a sus trabajos a localizar los archivos que necesitan.

Los adjuntos de trabajo ofrecen varios beneficios

- Mueva archivos entre hosts mediante Amazon S3
- Transfiera archivos desde su estación de trabajo a los hosts de los trabajadores y viceversa
- Disponible para los trabajos en colas en los que se habilita la función
- Se utiliza principalmente con flotas gestionadas por el servicio, pero también es compatible con las flotas gestionadas por el cliente.

Utilice los perfiles de almacenamiento para mapear el diseño de las ubicaciones de los sistemas de archivos compartidos en su estación de trabajo y en los hosts de los trabajadores. Este mapeo ayuda a sus trabajos a localizar los archivos y directorios compartidos cuando sus ubicaciones difieren entre la estación de trabajo y los hosts de los trabajadores, por ejemplo, en configuraciones multiplataforma con estaciones de trabajo basadas y hosts de trabajo basados en servidores de trabajo Windows basados. Linux Los adjuntos de trabajo también utilizan el mapa del perfil de almacenamiento de la configuración del sistema de archivos para identificar los archivos que necesitan transferirse de un host a otro a través de Amazon S3.

Si no utiliza adjuntos de tareas y no necesita reasignar las ubicaciones de los archivos y directorios entre las estaciones de trabajo y los hosts de los trabajadores, no necesitará modelar sus archivos compartidos con perfiles de almacenamiento.

Temas

Ejemplo de infraestructura de proyecto

Para demostrar el uso de los archivos adjuntos de trabajo y los perfiles de almacenamiento, configure un entorno de prueba con dos proyectos distintos. Puede usar la consola de Deadline Cloud para crear los recursos de prueba.

1. Si aún no lo has hecho, crea una granja de pruebas. Para crear una granja, siga el procedimiento descrito en Crear una granja.
2. Cree dos colas para los trabajos en cada uno de los dos proyectos. Para crear colas, siga el procedimiento descrito en Crear una cola.

    a. Cree la primera cola llamada. **`Q1`** Utilice la siguiente configuración, utilice los valores predeterminados para todos los demás elementos.

    - Para adjuntar trabajos, selecciona Crear un nuevo bucket de Amazon S3.
    - Seleccione Habilitar la asociación con flotas gestionadas por el cliente.
    - Para ejecutar como usuario, introduzca tanto el usuario como **`jobuser`** el grupo de POSIX.
    - Para el rol de servicio de colas, cree un nuevo rol llamado **`AssetDemoFarm-Q1-Role`**
    - Desactive la casilla de verificación del entorno de colas conda predeterminado.

    b. Cree la segunda cola llamada. **`Q2`** Utilice la siguiente configuración, utilice los valores predeterminados para todos los demás elementos.

    - Para adjuntar trabajos, selecciona Crear un nuevo bucket de Amazon S3.
    - Seleccione Habilitar la asociación con flotas gestionadas por el cliente.
    - Para ejecutar como usuario, introduzca tanto el usuario como **`jobuser`** el grupo de POSIX.
    - Para el rol de servicio de colas, cree un nuevo rol llamado **`AssetDemoFarm-Q2-Role`**

- Desactive la casilla de verificación del entorno de colas conda predeterminado.

3. Cree una flota única gestionada por el cliente que ejecute los trabajos de ambas colas. Para crear la flota, siga el procedimiento descrito en Crear una flota gestionada por el cliente. Utilice la siguiente configuración:

   - Para Nombre, utilice**`DemoFleet`**.
   - Para el tipo de flota, seleccione Gestionado por el cliente
   - Para el rol de servicio de flota, cree un nuevo rol denominado AssetDemoFarm-Fleet-Role.
   - No asocies la flota a ninguna cola.

El entorno de prueba supone que hay tres sistemas de archivos compartidos entre los hosts que utilizan recursos compartidos de archivos de red. En este ejemplo, las ubicaciones tienen los siguientes nombres:

- `FSCommon`- contiene activos de trabajo de entrada que son comunes a ambos proyectos.
- `FS1`- contiene los activos de trabajo de entrada y salida para el proyecto 1.
- `FS2`- contiene los activos de trabajo de entrada y salida para el proyecto 2.

El entorno de prueba también supone que hay tres estaciones de trabajo, de la siguiente manera:

- `WSAll`- Una estación de trabajo Linux basada que utilizan los desarrolladores para todos los proyectos. Las ubicaciones del sistema de archivos compartidos son:
  - `FSCommon`: `/shared/common`
  - `FS1`: `/shared/projects/project1`
  - `FS2`: `/shared/projects/project2`
- `WS1`- Una estación de trabajo Windows basada en el proyecto 1. Las ubicaciones del sistema de archivos compartidos son:
  - `FSCommon`: `S:\`
  - `FS1`: `Z:\`
  - `FS2`: No disponible
- `WS1`- Una estación de trabajo macOS basada en el proyecto 2. Las ubicaciones del sistema de archivos compartidos son:
  - `FSCommon`: `/Volumes/common`

- FS1: No disponible
- FS2: /Volumes/projects/project2

Por último, defina las ubicaciones del sistema de archivos compartidos para los trabajadores de su flota. Los ejemplos siguientes se refieren a esta configuración comoWorkerConfig. Las ubicaciones compartidas son:

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

No necesita configurar ningún sistema de archivos, estaciones de trabajo o trabajadores compartidos que coincidan con esta configuración. No es necesario que las ubicaciones compartidas existan para la demostración.

Perfiles de almacenamiento y mapeo de rutas

Utilice los perfiles de almacenamiento para modelar los sistemas de archivos de las estaciones de trabajo y los hosts de los trabajadores. Cada perfil de almacenamiento describe el diseño del sistema operativo y del sistema de archivos de una de las configuraciones del sistema. En este tema se describe cómo usar los perfiles de almacenamiento para modelar las configuraciones del sistema de archivos de sus hosts, de modo que Deadline Cloud pueda generar reglas de mapeo de rutas para sus trabajos, y cómo se generan esas reglas de mapeo de rutas a partir de sus perfiles de almacenamiento.

Cuando envíes un trabajo a Deadline Cloud, puedes proporcionar un ID de perfil de almacenamiento opcional para el trabajo. Este perfil de almacenamiento describe el sistema de archivos de la estación de trabajo que lo envía. Describe la configuración original del sistema de archivos que utilizan las rutas de archivos de la plantilla de trabajo.

También puede asociar un perfil de almacenamiento a una flota. El perfil de almacenamiento describe la configuración del sistema de archivos de todos los hosts trabajadores de la flota. Si tiene trabajadores con una configuración de sistema de archivos diferente, esos trabajadores deben estar asignados a una flota diferente en su granja.

Las reglas de mapeo de rutas describen cómo se deben reasignar las rutas desde la forma en que se especifican en el trabajo hasta la ubicación real de la ruta en el host de un trabajador. Deadline

Cloud compara la configuración del sistema de archivos descrita en el perfil de almacenamiento de un trabajo con el perfil de almacenamiento de la flota que ejecuta el trabajo para derivar estas reglas de mapeo de rutas.

Temas

Modele ubicaciones de sistemas de archivos compartidos con perfiles de almacenamiento

Un perfil de almacenamiento modela la configuración del sistema de archivos de una de las configuraciones de su host. Hay cuatro configuraciones de host diferentes en la infraestructura del proyecto de muestra. En este ejemplo, se crea un perfil de almacenamiento independiente para cada uno. Puede crear un perfil de almacenamiento mediante cualquiera de las siguientes opciones:

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation recurso
- Consola de AWS

Un perfil de almacenamiento se compone de una lista de ubicaciones de sistemas de archivos, cada una de las cuales indica a Deadline Cloud la ubicación y el tipo de ubicación del sistema de archivos relevantes para los trabajos enviados desde o ejecutados en un host. Un perfil de almacenamiento solo debe modelar las ubicaciones relevantes para los trabajos. Por ejemplo, la `FSCommon` ubicación compartida se encuentra en la estación de trabajo de `WS1S:\`, por lo que la ubicación correspondiente del sistema de archivos es:

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

Utilice los siguientes comandos para crear el perfil de almacenamiento para las configuraciones de las `WS1` estaciones de trabajo `WS3` y la configuración de trabajo `WorkerConfig` mediante el AWS CLIcomando in: `WS2` AWS CloudShell

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

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

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

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

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

```
  ]'
```

> **Note**
>
> Debe consultar las ubicaciones del sistema de archivos en sus perfiles de almacenamiento utilizando los mismos valores para la `name` propiedad en todos los perfiles de almacenamiento de su granja. Deadline Cloud compara los nombres para determinar si las ubicaciones de los sistemas de archivos de diferentes perfiles de almacenamiento hacen referencia a la misma ubicación al generar las reglas de mapeo de rutas.

Configure los perfiles de almacenamiento para las flotas

Puede configurar una flota para que incluya un perfil de almacenamiento que modele las ubicaciones de los sistemas de archivos de todos los trabajadores de la flota. La configuración del sistema de archivos anfitrión de todos los trabajadores de una flota debe coincidir con el perfil de almacenamiento de la flota. Los trabajadores con diferentes configuraciones de sistemas de archivos deben estar en flotas separadas.

Para establecer la configuración de su flota para usar el perfil de `WorkerConfig` almacenamiento, utilice: 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
    }
  }"
```

Configure los perfiles de almacenamiento para las colas

La configuración de una cola incluye una lista de nombres de las ubicaciones del sistema de archivos compartidos a las que deben acceder los trabajos enviados a la cola, que distinguen mayúsculas de minúsculas. Por ejemplo, los trabajos enviados a la cola Q1 requieren ubicaciones del sistema de archivos y. FSCommon FS1 Los trabajos enviados a la cola requieren ubicaciones en los sistemas de archivos Q2 y. FSCommon FS2

Para configurar las configuraciones de la cola de modo que requieran estas ubicaciones del sistema de archivos, utilice el siguiente script:

```
# 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 configuración de una cola también incluye una lista de perfiles de almacenamiento permitidos que se aplica a los trabajos enviados a esa cola y a las flotas asociadas a ella. En la lista de perfiles de almacenamiento permitidos de la cola solo se permiten los perfiles de almacenamiento que definen las ubicaciones del sistema de archivos para todas las ubicaciones de sistemas de archivos requeridas para la cola.

Se produce un error en un trabajo si lo envía con un perfil de almacenamiento que no figura en la lista de perfiles de almacenamiento permitidos para la cola. Siempre puedes enviar un trabajo sin perfil de almacenamiento a una cola. Las configuraciones de las estaciones de trabajo están

etiquetadas `WSAll` y `WS1` ambas tienen las ubicaciones de sistema de archivos requeridas (`FSCommonyFS1`) para la cola. `Q1` Deben poder enviar los trabajos a la cola. Del mismo modo, las estaciones de trabajo configuran `WSAll` y `WS2` cumplen los requisitos de cola. `Q2` Deben poder enviar trabajos a esa cola. Actualice ambas configuraciones de cola para permitir que los trabajos se envíen con estos perfiles de almacenamiento mediante el siguiente script:

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

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

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

Si agrega el perfil `WS2` de almacenamiento a la lista de perfiles de almacenamiento permitidos para la cola, se produce `Q1` un error:

```
$ 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
```

Esto se debe a que el perfil de `WS2` almacenamiento no contiene una definición para la ubicación del sistema de archivos con el nombre `FS1` que requiere la cola`Q1`.

También se produce un error al asociar una flota configurada con un perfil de almacenamiento que no está en la lista de perfiles de almacenamiento permitidos de la cola. Por ejemplo:

```
$ 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.
```

Para corregir el error, añada el perfil de almacenamiento mencionado `WorkerConfig` a la lista de perfiles de almacenamiento permitidos tanto para la cola como para la cola`Q1`. `Q2` A continuación, asocie la flota a estas colas para que los trabajadores de la flota puedan ejecutar los trabajos desde ambas colas.

```
# 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
```

Obtenga reglas de mapeo de rutas a partir de los perfiles de almacenamiento

Las reglas de mapeo de rutas describen cómo se deben reasignar las rutas desde el trabajo hasta la ubicación real de la ruta en un host de trabajo. Cuando se ejecuta una tarea en un trabajador, el perfil de almacenamiento del trabajo se compara con el perfil de almacenamiento de la flota del trabajador para obtener las reglas de mapeo de rutas de la tarea.

Deadline Cloud crea una regla de mapeo para cada una de las ubicaciones del sistema de archivos requeridas en la configuración de la cola. Por ejemplo, un trabajo enviado con el perfil de `WSAll` almacenamiento a la cola `Q1` tiene las siguientes reglas de mapeo de rutas:

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

Deadline Cloud crea reglas para las ubicaciones `FSComm` y del sistema de `FS1` archivos, pero no para la ubicación del sistema de `FS2` archivos, aunque estén definidas tanto por el `WSAll` perfil como

por el `WorkerConfig` de almacenamiento. `FS2` Esto se debe a que `Q1` la lista de ubicaciones de sistemas de archivos obligatorias de Queue sí lo es`["FSComm", "FS1"]`.

Para confirmar las reglas de mapeo de rutas disponibles para los trabajos enviados con un perfil de almacenamiento concreto, envíe un trabajo que imprima el archivo de reglas de mapeo de rutas de Open Job Description y, a continuación, lea el registro de sesión una vez finalizado el trabajo:

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

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

Si usa la CLI de Deadline Cloud para enviar trabajos, su `settings.storage_profile_id` configuración establece el perfil de almacenamiento que tendrán los trabajos enviados con la CLI. Para enviar trabajos con el perfil `WSAll` de almacenamiento, configure:

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

Para ejecutar un trabajador gestionado por el cliente como si se estuviera ejecutando en la infraestructura de ejemplo, siga el procedimiento descrito en Ejecute el agente de trabajo de la Guía del usuario de Deadline Cloud para ejecutar un trabajador. AWS CloudShell Si ha seguido esas instrucciones anteriormente, elimine primero los `~/demoenv-persist` directorios `~/demoenv-logs` y. Además, establezca los valores de las variables de `DEV_CMF_ID` entorno `DEV_FARM_ID` y las variables de entorno a las que hacen referencia las instrucciones de la siguiente manera antes de hacerlo:

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

Una vez ejecutado el trabajo, puede ver las reglas de mapeo de rutas en el archivo de registro del trabajo:

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

El registro contiene el mapeo del sistema de `FSComm` archivos `FS1` y del sistema de archivos. Reformateada para facilitar la lectura, la entrada de registro tiene el siguiente aspecto:

```
{
    "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"
        }
    ]
```

Puede enviar trabajos con diferentes perfiles de almacenamiento para ver cómo cambian las reglas de mapeo de rutas.

Usa archivos adjuntos de trabajo para compartir archivos

Usa los archivos adjuntos de trabajo para hacer que los archivos que no están en los directorios compartidos estén disponibles para tus trabajos y para capturar los archivos de salida si no están escritos en los directorios compartidos. Job attachments utiliza Amazon S3 para transferir archivos entre hosts. Los archivos se almacenan en depósitos de S3 y no es necesario cargar un archivo si su contenido no ha cambiado.

Debe usar adjuntos de trabajo cuando ejecute trabajos en flotas administradas por el servicio, ya que los hosts no comparten las ubicaciones del sistema de archivos. Los adjuntos de trabajo también son útiles con las flotas administradas por el cliente cuando los archivos de entrada o salida de un trabajo se almacenan en un sistema de archivos de red compartido, como cuando el paquete de trabajos contiene scripts de shell o Python.

Cuando envía un paquete de trabajos con la CLI de Deadline Cloud o un remitente de Deadline Cloud, los adjuntos de trabajos utilizan el perfil de almacenamiento del trabajo y las ubicaciones del sistema de archivos requeridas por la cola para identificar los archivos de entrada que no se encuentran en el host de un trabajador y que deben cargarse en Amazon S3 como parte del envío de trabajos. Estos perfiles de almacenamiento también ayudan a Deadline Cloud a identificar los archivos de salida en las ubicaciones de alojamiento de los trabajadores que deben cargarse en Amazon S3 para que estén disponibles en su estación de trabajo.

Los ejemplos de adjuntos de trabajo utilizan las configuraciones de granja, flota, colas y perfiles de almacenamiento desde Ejemplo de infraestructura de proyecto y. Perfiles de almacenamiento y mapeo de rutas Deberías revisar esas secciones antes que esta.

En los ejemplos siguientes, utiliza un paquete de trabajos de muestra como punto de partida y, a continuación, lo modifica para explorar las funciones de Job Adjunt. Los paquetes de trabajos son la mejor forma de que sus trabajos utilicen adjuntos de trabajo. Combinan una plantilla de trabajo de Open Job Description en un directorio con archivos adicionales que enumeran los archivos y directorios necesarios para los trabajos que utilizan el paquete de trabajos. Para obtener más información sobre los paquetes de trabajos, consultePlantillas Open Job Description (OpenJD) para Deadline Cloud.

Envío de archivos con un trabajo

Con Deadline Cloud, puedes permitir que los flujos de trabajo accedan a los archivos de entrada que no están disponibles en las ubicaciones de los sistemas de archivos compartidos de los anfitriones

de los trabajadores. Los adjuntos de trabajo permiten que los trabajos de renderización accedan a los archivos que se encuentran únicamente en la unidad de una estación de trabajo local o en un entorno de flota gestionado por el servicio. Al enviar un paquete de trabajos, puede incluir listas de los archivos y directorios de entrada necesarios para el trabajo. Deadline Cloud identifica estos archivos no compartidos, los carga desde la máquina local a Amazon S3 y los descarga en el host del trabajador. Agiliza el proceso de transferencia de los activos de entrada a los nodos de renderizado, lo que garantiza que todos los archivos necesarios estén accesibles para la ejecución distribuida de los trabajos.

Puede especificar los archivos de los trabajos directamente en el paquete de trabajos, utilizar los parámetros de la plantilla de trabajo que proporcione mediante variables de entorno o un script y utilizar el `assets_references` archivo del trabajo. Puede utilizar uno de estos métodos o una combinación de los tres. Puede especificar un perfil de almacenamiento para el paquete del trabajo de modo que solo cargue los archivos que se hayan modificado en la estación de trabajo local.

En esta sección, se utiliza un ejemplo de paquete de trabajos GitHub para demostrar cómo Deadline Cloud identifica los archivos del trabajo que va a cargar, cómo se organizan esos archivos en Amazon S3 y cómo se ponen a disposición de los anfitriones de los trabajadores que procesan sus trabajos.

Temas

- Cómo carga Deadline Cloud los archivos a Amazon S3
- Cómo elige Deadline Cloud los archivos que desea cargar
- ¿Cómo encuentran los trabajos los archivos de entrada adjuntos a los trabajos?

Cómo carga Deadline Cloud los archivos a Amazon S3

En este ejemplo, se muestra cómo Deadline Cloud carga archivos desde su estación de trabajo o host de trabajo a Amazon S3 para poder compartirlos. Utiliza un paquete de trabajos de muestra GitHub y la CLI de Deadline Cloud para enviar los trabajos.

Comience por clonar el GitHubrepositorio de muestras de Deadline Cloud en su AWS CloudShellentorno y, a continuación, copie el paquete de `job_attachments_devguide` trabajos en su directorio principal:

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

Instale la CLI de Deadline Cloud para enviar paquetes de trabajos:

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

El paquete de `job_attachments_devguide` tareas consta de un solo paso con una tarea que ejecuta un script de shell en bash cuya ubicación en el sistema de archivos se transmite como parámetro del trabajo. La definición del parámetro de trabajo es:

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

El `IN` valor de la `dataFlow` propiedad indica a los adjuntos del trabajo que el valor del `ScriptFile` parámetro es una entrada para el trabajo. El valor de la `default` propiedad es una ubicación relativa al directorio del paquete de tareas, pero también puede ser una ruta absoluta. Esta definición de parámetro declara el `script.sh` archivo del directorio del paquete de trabajos como un archivo de entrada necesario para que se ejecute el trabajo.

A continuación, asegúrese de que la CLI de Deadline Cloud no tenga un perfil de almacenamiento configurado y, a continuación, envíe el trabajo a la colaQ1:

```
# 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/
```

El resultado de la CLI de Deadline Cloud después de ejecutar este comando tiene el siguiente aspecto:

```
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
```

Al enviar el trabajo, Deadline Cloud primero crea un hash del `script.sh` archivo y, a continuación, lo carga en Amazon S3.

Deadline Cloud trata el depósito de S3 como almacenamiento direccionable por contenido. Los archivos se cargan en objetos de S3. El nombre del objeto se deriva de un hash del contenido del archivo. Si dos archivos tienen el mismo contenido, tienen el mismo valor hash independientemente de dónde estén ubicados los archivos o de su nombre. Este almacenamiento de contenido direccionable permite a Deadline Cloud evitar cargar un archivo si ya está disponible.

Puede usar la AWS CLI para ver los objetos que se cargaron en 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
```

Se cargaron dos objetos en S3:

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— El contenido `descript.sh`. El valor `87cb19095dd5d78fcaf56384ef0e6241` de la clave del objeto es el

hash del contenido del archivo y la extensión `xxh128` indica que el valor hash se calculó como un xxhash de 128 bits.

- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— El objeto del manifiesto para el envío del trabajo. Los valores `<farm-id>` y `<guid>` son el identificador de la granja, el identificador de la cola y un valor hexadecimal aleatorio. `<queue-id>` El valor `a1d221c7fd97b08175b3872a37428e8c` de este ejemplo es un valor hash calculado a partir de la cadena`/home/cloudshell-user/job_attachments_devguide`, el directorio en el que se encuentra`script.sh`.

El objeto de manifiesto contiene la información de los archivos de entrada de una ruta raíz específica que se cargaron en S3 como parte del envío del trabajo. Descargue este archivo de manifiesto (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`). Su contenido es similar al de:

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

Esto indica que el archivo `script.sh` se ha cargado y el hash del contenido de ese archivo lo es`87cb19095dd5d78fcaf56384ef0e6241`. Este valor hash coincide con el valor del nombre del objeto`DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`. Deadline Cloud lo utiliza para saber qué objeto descargar para el contenido de este archivo.

El esquema completo de este archivo está disponible en GitHub.

Al utilizar la CreateJob operación, puede establecer la ubicación de los objetos del manifiesto. Puedes usar la GetJoboperación para ver la ubicación:

```
{
```

```
    "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"
            }
        ]
    },
    ...
}
```

Cómo elige Deadline Cloud los archivos que desea cargar

Los archivos y directorios que job attachments considera para cargar en Amazon S3 como entradas para su trabajo son:

- Los valores de todos los parámetros `PATH` de trabajo de tipo definido en la plantilla de trabajo del paquete de trabajos con un `dataFlow` valor de `IN` o`INOUT`.
- Los archivos y directorios que aparecen como entradas en el archivo de referencias de activos del paquete de trabajos.

Si envía un trabajo sin perfil de almacenamiento, se cargarán todos los archivos que desee cargar. Si envía un trabajo con un perfil de almacenamiento, los archivos no se cargan en Amazon S3 si se encuentran en ubicaciones de sistema de archivos `SHARED` tipo perfil de almacenamiento que también son ubicaciones de sistema de archivos obligatorias para la cola. Se espera que estas ubicaciones estén disponibles en los hosts de trabajo que ejecutan el trabajo, por lo que no es necesario cargarlas en S3.

En este ejemplo, crea ubicaciones de sistemas de `SHARED` archivos `WSAll` en su CloudShell entorno de AWS y, a continuación, añade archivos a esas ubicaciones de sistemas de archivos. Utilice el siguiente 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
```

A continuación, añada un archivo de referencias de activos al paquete de trabajos que incluya todos los archivos que creó como entradas para el trabajo. Utilice el siguiente 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
```

A continuación, configure la CLI de Deadline Cloud para enviar los trabajos con el perfil de `WSAll` almacenamiento y, a continuación, envíe el paquete de trabajos:

```

# 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 carga dos archivos a Amazon S3 cuando envías el trabajo. Puede descargar los objetos del manifiesto del trabajo desde S3 para ver los archivos cargados:

```
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
```

En este ejemplo, hay un único archivo de manifiesto con el siguiente contenido:

```
{
    "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
}
```

Usa la GetJob operación del manifiesto para comprobar que `rootPath` es «/».

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

La ruta raíz del conjunto de archivos de entrada es siempre la subruta común más larga de esos archivos. Si el trabajo se envió desde, Windows en cambio, y hay archivos de entrada que no tienen una subruta común porque se encuentran en unidades diferentes, verá una ruta raíz independiente en cada unidad. Las rutas de un manifiesto siempre son relativas a la ruta raíz del manifiesto, por lo que los archivos de entrada que se cargaron son los siguientes:

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— El archivo de script del paquete de tareas.

- `/shared/projects/project2/file.txt`— El archivo en una ubicación del sistema de `SHARED` archivos del perfil `WSAll` de almacenamiento que no figura en la lista de ubicaciones de sistemas de archivos obligatorias para la cola`Q1`.

Los archivos en las ubicaciones del sistema de archivos `FSCommon` (`/shared/common/file.txt`) y `FS1` (`/shared/projects/project1/file.txt`) no están en la lista. Esto se debe a que esas ubicaciones del sistema de archivos están `SHARED` en el perfil de `WSAll` almacenamiento y ambas están en la lista de ubicaciones obligatorias del sistema de archivos en cola`Q1`.

Con la GetStorageProfileForQueue operación, puede ver las ubicaciones de los sistemas de archivos consideradas `SHARED` para un trabajo que se envía con un perfil de almacenamiento determinado. Para consultar el perfil de almacenamiento `WSAll` de la cola, `Q1` utilice el siguiente 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
```

¿Cómo encuentran los trabajos los archivos de entrada adjuntos a los trabajos?

Para que un trabajo utilice los archivos que Deadline Cloud carga en Amazon S3 mediante adjuntos de trabajo, su trabajo necesita esos archivos disponibles a través del sistema de archivos de los hosts de los trabajadores. Cuando una sesión de tu trabajo se ejecuta en el host de un trabajador, Deadline Cloud descarga los archivos de entrada del trabajo en un directorio temporal de la unidad local del anfitrión del trabajador y añade reglas de mapeo de rutas para cada una de las rutas raíz del trabajo a la ubicación del sistema de archivos en la unidad local.

Para este ejemplo, inicie el agente de trabajo de Deadline Cloud en una CloudShell pestaña de AWS. Deje que los trabajos enviados anteriormente terminen de ejecutarse y, a continuación, elimine los registros de trabajos del directorio de registros:

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

El siguiente script modifica el paquete de trabajos para mostrar todos los archivos del directorio de trabajo temporal de la sesión y el contenido del archivo de reglas de mapeo de rutas y, a continuación, envía un trabajo con el paquete modificado:

```
# 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/
```

Puede consultar el registro de la ejecución de la tarea después de que la haya ejecutado el trabajador de su AWS CloudShell entorno:

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

El registro muestra que lo primero que ocurre en la sesión es que los dos archivos de entrada del trabajo se descargan al trabajador:

```
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.
```

El siguiente es el resultado de `script.sh` run by the job:

- Los archivos de entrada que se cargaron al enviar el trabajo se encuentran en un directorio cuyo nombre comienza por «assetroot» en el directorio temporal de la sesión.
- Las rutas de los archivos de entrada se han reubicado en relación con el directorio «assetroot» en lugar de en relación con la ruta raíz del manifiesto de entrada () del trabajo. `"/"`
- El archivo de reglas de mapeo de rutas contiene una regla adicional que se reasigna `"/"` a la ruta absoluta del directorio «assetroot».

Por ejemplo:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
```

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

> **Note**
>
> Si el trabajo que envías tiene varios manifiestos con diferentes rutas raíz, habrá un directorio con el nombre «assetroot» diferente para cada una de las rutas raíz.

Si necesita hacer referencia a la ubicación del sistema de archivos reubicado de uno de sus archivos de entrada, directorios o ubicaciones del sistema de archivos, puede procesar el archivo de reglas de mapeo de rutas en su trabajo y realizar la reasignación usted mismo, o agregar un parámetro de trabajo de PATH tipo a la plantilla de trabajo en su paquete de trabajos y pasar el valor que necesita reasignar como valor de ese parámetro. Por ejemplo, el ejemplo siguiente modifica el paquete de trabajos para que tenga uno de estos parámetros de trabajo y, a continuación, envía un trabajo con la ubicación del sistema de archivos `/shared/projects/project2` como valor:

```
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
```

El archivo de registro de la ejecución de este trabajo contiene el resultado:

```
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
```

Obtener los archivos de salida de un trabajo

En este ejemplo, se muestra cómo Deadline Cloud identifica los archivos de salida que generan sus trabajos, decide si los carga en Amazon S3 y cómo puede colocarlos en su estación de trabajo.

En este ejemplo, utilice el `job_attachments_devguide_output` paquete de `job_attachments_devguide` trabajos en lugar del paquete de trabajos. Comience por hacer una copia del paquete en su AWS CloudShell entorno a partir de su clon del GitHub repositorio de muestras de Deadline Cloud:

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

La diferencia importante entre este paquete de trabajos y el paquete de `job_attachments_devguide` trabajos es la adición de un nuevo parámetro de trabajo en la plantilla de trabajo:

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

La `dataFlow` propiedad del parámetro tiene el valor`OUT`. Deadline Cloud utiliza el valor de los parámetros del `dataFlow` trabajo con un valor `IN`OUT igual `OUT` o como resultados de su trabajo. Si la ubicación del sistema de archivos transferida como valor a este tipo de parámetros de trabajo se reasigna a una ubicación del sistema de archivos local del trabajador que ejecuta el trabajo, Deadline Cloud buscará nuevos archivos en la ubicación y los cargará en Amazon S3 como resultados del trabajo.

Para ver cómo funciona, primero inicia el agente de trabajadores de Deadline Cloud en una AWS CloudShell pestaña. Deje que los trabajos enviados anteriormente terminen de ejecutarse. A continuación, elimine los registros de trabajos del directorio de registros:

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

A continuación, envíe un trabajo con este paquete de trabajos. Después de que el trabajador CloudShell ejecute sus ejecuciones, observe los registros:

```
# 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
```

El registro muestra que se detectó un archivo como salida y se cargó en 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.
```

El registro también muestra que Deadline Cloud creó un nuevo objeto de manifiesto en el bucket de Amazon S3 configurado para que lo usen los adjuntos de trabajos en colaQ1. El nombre del objeto de manifiesto se deriva de la granja, la cola, el trabajo, el paso, la tarea, la marca de tiempo y los `sessionaction` identificadores de la tarea que generó el resultado. Descarga este archivo de manifiesto para ver dónde ha colocado Deadline Cloud los archivos de salida para esta tarea:

```
# 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 .
```

El manifiesto tiene este aspecto:

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

Esto muestra que el contenido del archivo de salida se guarda en Amazon S3 de la misma manera que se guardan los archivos de entrada del trabajo. Al igual que los archivos de entrada, el archivo de salida se almacena en S3 con un nombre de objeto que contiene el hash del archivo y el prefijo`DeadlineCloud/Data`.

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

Puede descargar el resultado de un trabajo a su estación de trabajo mediante el monitor de Deadline Cloud o la CLI de Deadline Cloud:

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

El valor del parámetro del `OutputDir` trabajo en el trabajo enviado es`./output_dir`, por lo que los resultados se descargan a un directorio llamado `output_dir` dentro del directorio del paquete de trabajos. Si especificó una ruta absoluta o una ubicación relativa diferente como valor`OutputDir`, los archivos de salida se descargarán en esa ubicación.

```
$ 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)
```

Utilizar archivos de un paso en un paso dependiente

Este ejemplo muestra cómo un paso de un trabajo puede acceder a los resultados de un paso del que depende en el mismo trabajo.

Para que los resultados de un paso estén disponibles para otro, Deadline Cloud añade acciones adicionales a una sesión para descargar esos resultados antes de ejecutar tareas en la sesión. Para indicarle de qué pasos debe descargar los resultados, debe declarar esos pasos como dependencias del paso que debe utilizar los resultados.

Utilice el paquete de `job_attachments_devguide_output` tareas para este ejemplo. Comience por hacer una copia en su AWS CloudShell entorno desde su clon del GitHub repositorio de muestras de Deadline Cloud. Modifíquelo para añadir un paso dependiente que solo se ejecute después del paso existente y utilice el resultado de ese paso:

```
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
```

El trabajo creado con este paquete de trabajos modificado se ejecuta en dos sesiones independientes, una para la tarea del paso «Paso» y otra para la tarea del paso «DependentStep».

Primero, inicie el agente de trabajo de Deadline Cloud en una CloudShell pestaña. Deje que los trabajos enviados anteriormente terminen de ejecutarse y, a continuación, elimine los registros de trabajos del directorio de registros:

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

A continuación, envíe un trabajo con el paquete de `job_attachments_devguide_output` trabajos modificado. Espere a que termine de ejecutarse en el trabajador de su CloudShell entorno. Observe los registros de las dos sesiones:

```
# 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-*
```

En el registro de sesiones de la tarea del paso mencionado`DependentStep`, se ejecutan dos acciones de descarga independientes:

```
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 primera acción descarga el `script.sh` archivo utilizado por el paso denominado «Paso». La segunda acción descarga los resultados de ese paso. Deadline Cloud determina qué archivos descargar utilizando el manifiesto de salida generado en ese paso como manifiesto de entrada.

Al final del mismo registro, puedes ver el resultado del paso denominado "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
```

Cree límites de recursos para los trabajos

Los trabajos enviados a Deadline Cloud pueden depender de los recursos que se comparten entre varios trabajos. Por ejemplo, una granja puede tener más trabajadores que las licencias flotantes para un recurso específico. O bien, es posible que un servidor de archivos compartido solo pueda entregar datos a un número limitado de trabajadores al mismo tiempo. En algunos casos, uno o más trabajos pueden ocupar todos estos recursos y provocar errores debido a que los recursos no están disponibles cuando comienzan a trabajar nuevos trabajadores.

Para ayudar a solucionar este problema, puedes usar límites para estos recursos limitados. Deadline Cloud tiene en cuenta la disponibilidad de los recursos limitados y utiliza esa información

para garantizar que los recursos estén disponibles a medida que se van incorporando nuevos trabajadores, de forma que los trabajos tengan menos probabilidades de fracasar debido a la falta de recursos.

Se establecen límites para toda la granja. Los trabajos enviados a una cola solo pueden adquirir los límites asociados a la cola. Si especificas un límite para un trabajo que no está asociado a la cola, el trabajo no es compatible y no se ejecutará.

Para usar un límite, debes

- Crea un límite
- Asocia un límite y una cola
- Envíe un trabajo que requiera límites

> **Note**
>
> Si ejecuta un trabajo que tiene recursos limitados en una cola que no está asociada a un límite, ese trabajo puede consumir todos los recursos. Si tiene un recurso restringido, asegúrese de que todos los pasos de los trabajos de las colas que utilizan el recurso estén asociados a un límite.

En el caso de los límites definidos en una granja, asociados a una cola y especificados en un trabajo, puede ocurrir una de estas cuatro cosas:

- Si crea un límite, lo asocia a una cola y especifica el límite en la plantilla de un trabajo, el trabajo se ejecuta y utiliza solo los recursos definidos en el límite.
- Si crea un límite, lo especifica en una plantilla de trabajo, pero no lo asocia a una cola, el trabajo se marcará como incompatible y no se ejecutará.
- Si crea un límite, no lo asocia a una cola ni especifica el límite en la plantilla de un trabajo, el trabajo se ejecuta pero no utiliza el límite.
- Si no utiliza ningún límite, el trabajo se ejecuta.

Si asocias un límite a varias colas, las colas comparten los recursos limitados por el límite. Por ejemplo, si crea un límite de 100 y una cola utiliza 60 recursos, las demás colas solo pueden utilizar 40 recursos. Cuando se libera un recurso, una tarea de cualquier cola lo puede ocupar.

Deadline Cloud proporciona dos AWS CloudFormation métricas para ayudarte a supervisar los recursos que proporciona un límite. Puede supervisar la cantidad actual de recursos en uso y la cantidad máxima de recursos disponibles dentro del límite. Para obtener más información, consulta las métricas del límite de recursos en la Guía para desarrolladores de Deadline Cloud.

Aplicas un límite a un paso de trabajo en una plantilla de trabajo. Al especificar la cantidad requerida (nombre) de un límite en la `amounts` sección `hostRequirements` de un paso y `amountRequirementName` se asocia un límite con el mismo nombre a la cola de trabajos, las tareas programadas para este paso están restringidas por el límite del recurso.

Si un paso requiere un recurso limitado por un límite alcanzado, más trabajadores no se encargarán de las tareas de ese paso.

Puede aplicar más de un límite a un paso del trabajo. Por ejemplo, si el paso utiliza dos licencias de software diferentes, puede aplicar un límite diferente para cada licencia. Si un paso requiere dos límites y se alcanza el límite de uno de los recursos, más trabajadores no se encargarán de las tareas de ese paso hasta que los recursos estén disponibles.

Detener y eliminar los límites

Al detener o eliminar la asociación entre una cola y un límite, un trabajo que utilice el límite deja de programar las tareas a partir de los pasos que requieren este límite y bloquea la creación de nuevas sesiones para un paso.

Las tareas que están preparadas permanecen listas y las tareas se reanudan automáticamente cuando la asociación entre la cola y el límite vuelve a activarse. No es necesario volver a poner en cola ningún trabajo.

Al detener o eliminar la asociación entre una cola y un límite, tiene dos opciones para detener la ejecución de tareas:

- Detener y cancelar tareas: los trabajadores con sesiones en las que se ha alcanzado el límite cancelan todas las tareas.
- Detener y terminar la ejecución de las tareas: los trabajadores con sesiones que han alcanzado el límite completan sus tareas.

Al eliminar un límite mediante la consola, los trabajadores primero dejan de ejecutar las tareas inmediatamente o, finalmente, cuando las terminan. Cuando se elimina la asociación, ocurre lo siguiente:

- Los pasos que requieren el límite están marcados como no compatibles.
- Se cancela todo el trabajo que contiene esos pasos, incluidos los pasos que no requieren el límite.
- El trabajo está marcado como no compatible.

Si la cola asociada al límite tiene una flota asociada con una capacidad de flota que coincide con la cantidad requerida (nombre del límite), esa flota seguirá procesando los trabajos con el límite especificado.

Crea un límite

Puede crear un límite mediante la consola de Deadline Cloud o la CreateLimit operación en la API de Deadline Cloud. Los límites se definen para una granja, pero se asocian a las colas. Tras crear un límite, puede asociarlo a una o más colas.

Para crear un límite

1. En el panel de la consola de Deadline Cloud (consola de Deadline Cloud), selecciona la granja para la que quieres crear una cola.
2. Elige la granja a la que quieres añadir el límite, selecciona la pestaña Límites y, a continuación, selecciona Crear límite.
3. Proporcione los detalles del límite. El nombre del importe obligatorio es el nombre utilizado en la plantilla de trabajo para identificar el límite. Debe empezar por el prefijo **`amount.`** seguido del nombre del importe. El nombre del importe requerido debe ser exclusivo en las colas asociadas al límite.
4. Si elige Establecer una cantidad máxima, esa es la cantidad total de recursos permitidos por este límite. Si eliges Sin cantidad máxima, el uso de los recursos no está limitado. Incluso cuando el uso de los recursos no está limitado, la CloudWatch métrica de `CurrentCount` Amazon se emite para que puedas realizar un seguimiento del uso. Para obtener más información, consulta CloudWatchlas métricas en la Guía para desarrolladores de Deadline Cloud.
5. Si ya conoces las colas que deberían usar el límite, puedes elegirlas ahora. No necesitas asociar una cola para crear un límite.
6. Selecciona Crear límite.

Asocia un límite y una cola

Tras crear un límite, puede asociar una o más colas al límite. Solo las colas asociadas a un límite utilizan los valores especificados en el límite.

Para crear una asociación con una cola, utilice la consola de Deadline Cloud o la CreateQueueLimitAssociation operación de la API de Deadline Cloud.

Para asociar una cola a un límite

1. En el panel de la consola de Deadline Cloud (consola de Deadline Cloud), selecciona la granja a la que quieres asociar un límite a una cola.
2. Selecciona la pestaña Límites, elige el límite al que quieres asociar una cola y, a continuación, selecciona Editar límite.
3. En la sección Asociar colas, elija las colas que desee asociar al límite.
4. Seleccione Save changes (Guardar cambios).

Envíe un trabajo que requiera límites

Para aplicar un límite, debe especificarlo como un requisito de anfitrión para el trabajo o paso del trabajo. Si no especificas un límite en un paso y ese paso usa un recurso asociado, el uso del paso no se descuenta del límite cuando se programan los trabajos.

Algunos remitentes de Deadline Cloud te permiten establecer un requisito de anfitrión. Puede especificar el nombre del requisito de cantidad del límite en el remitente para aplicar el límite.

Si el remitente no admite la adición de requisitos de anfitrión, también puedes aplicar un límite editando la plantilla de trabajo correspondiente al trabajo.

Para aplicar un límite a un paso de trabajo del paquete de trabajos

1. Abra la plantilla de trabajo del trabajo mediante un editor de texto. La plantilla de trabajo se encuentra en el directorio del paquete de trabajos del trabajo. Para obtener más información, consulte Paquetes de trabajos en la Guía para desarrolladores de Deadline Cloud.
2. Busca la definición del paso al que quieres aplicar el límite.
3. Añada lo siguiente a la definición del paso. `amount.name`Sustitúyalo por el nombre del importe obligatorio de tu límite. Para un uso normal, debe establecer el `min` valor en 1.

YAML

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

JSON

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

Puede añadir varios límites a un paso de trabajo de la siguiente manera. Sustituya *amount.name_1* y *amount.name_2* por los nombres de los requisitos de importe de sus límites.

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. Guarde los cambios en la plantilla de trabajo.

Cómo enviar un trabajo a Deadline Cloud

Hay muchas formas diferentes de enviar trabajos a AWS Deadline Cloud. En esta sección, se describen algunas de las formas en las que puedes enviar trabajos utilizando las herramientas que ofrece Deadline Cloud o creando tus propias herramientas personalizadas para tus cargas de trabajo.

- Desde un terminal: para cuando estés desarrollando un paquete de trabajos por primera vez o cuando los usuarios que envíen un trabajo se sientan cómodos usando la línea de comandos
- Desde un script: para personalizar y automatizar las cargas de trabajo
- Desde una aplicación: para cuando el trabajo del usuario está en una aplicación o cuando el contexto de una aplicación es importante.

Los ejemplos siguientes utilizan la biblioteca de `deadline` Python y la herramienta de línea de `deadline` comandos. Ambas están disponibles PyPiy alojadas en GitHub.

Temas

- Envía un trabajo a Deadline Cloud desde una terminal
- Envía un trabajo a Deadline Cloud mediante un script
- Envíe un trabajo dentro de una solicitud

Envía un trabajo a Deadline Cloud desde una terminal

Con solo un paquete de trabajos y la CLI de Deadline Cloud, usted o sus usuarios más técnicos pueden repetir rápidamente la redacción de paquetes de trabajos para probar el envío de un trabajo. Usa el siguiente comando para enviar un paquete de trabajos:

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

Si envía un paquete de trabajos con parámetros que no tienen valores predeterminados en el paquete, puede especificarlos con la `--parameter` opción`-p`/.

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

Para obtener una lista completa de las opciones disponibles, ejecute el comando help:

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

Envíe un trabajo a Deadline Cloud mediante una interfaz gráfica

La CLI de Deadline Cloud también incluye una interfaz gráfica de usuario que permite a los usuarios ver los parámetros que deben proporcionar antes de enviar un trabajo. Si sus usuarios prefieren no interactuar con la línea de comandos, puede escribir un atajo de escritorio que abra un cuadro de diálogo para enviar un paquete de tareas específico:

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

Utilice la `--browse` opción can para que el usuario pueda seleccionar un paquete de trabajos:

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

Para obtener una lista completa de las opciones disponibles, ejecute el comando help:

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

Envía un trabajo a Deadline Cloud mediante un script

Para automatizar el envío de trabajos a Deadline Cloud, puedes programarlos con herramientas como bash, Powershell y archivos por lotes.

Puedes añadir funciones como rellenar los parámetros del trabajo a partir de variables de entorno u otras aplicaciones. También puede enviar varios trabajos seguidos o programar la creación de un paquete de trabajos para enviarlos.

Enviar un trabajo con Python

Deadline Cloud también tiene una biblioteca Python de código abierto para interactuar con el servicio. El código fuente está disponible en GitHub.

La biblioteca está disponible en pypi a través de pip (). `pip install deadline` Es la misma biblioteca que utiliza la herramienta CLI de 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)
```

Para crear un diálogo como el `deadline bundle gui-submit` comando, puede utilizar la `show_job_bundle_submitter` función de `deadline.client.ui.job_bundle_submitter`.

En el siguiente ejemplo, se inicia una aplicación de Qt y se muestra el remitente del paquete de tareas:

```
# 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)
```

Para crear su propio diálogo, puede usar la `SubmitJobToDeadlineDialog` clase en `deadline.client.ui.dialogs.submit_job_to_deadline_dialog`. Puede transferir valores, incrustar su propia pestaña específica para el trabajo y determinar cómo se crea (o transfiere) el paquete de trabajos.

Envíe un trabajo dentro de una solicitud

Para facilitar a los usuarios el envío de trabajos, puede utilizar los tiempos de ejecución de secuencias de comandos o los sistemas de complementos que proporciona una aplicación. Los usuarios disponen de una interfaz familiar y se pueden crear potentes herramientas que les ayuden a enviar una carga de trabajo.

Inserte paquetes de trabajos en una aplicación

En este ejemplo, se muestra el envío de los paquetes de trabajos que usted pone a disposición en la solicitud.

Para dar a un usuario acceso a estos paquetes de trabajos, cree un script incrustado en un elemento del menú que inicie la CLI de Deadline Cloud.

El siguiente script permite al usuario seleccionar el paquete de trabajos:

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

Para utilizar en su lugar un paquete de tareas específico en un elemento del menú, utilice lo siguiente:

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

Esto abre un cuadro de diálogo en el que el usuario puede modificar los parámetros, las entradas y las salidas del trabajo y, a continuación, enviar el trabajo. Puede disponer de diferentes elementos de menú para distintos paquetes de trabajo para que un usuario los envíe en una solicitud.

Si el trabajo que envía con un paquete de trabajos contiene parámetros y referencias de activos similares en todas las solicitudes, puede rellenar los valores predeterminados del paquete de trabajos subyacente.

Obtenga información de una solicitud

Para extraer información de una aplicación para que los usuarios no tengan que añadirla manualmente a la presentación, puedes integrar Deadline Cloud con la aplicación para que tus

usuarios puedan enviar los trabajos mediante una interfaz familiar sin necesidad de salir de la aplicación ni utilizar herramientas de línea de comandos.

Si su aplicación tiene un tiempo de ejecución de secuencias de comandos compatible con Python y pyside/pyqt, puede usar los componentes de la interfaz gráfica de usuario de la biblioteca de clientes de Deadline Cloud para crear una interfaz de usuario. Para ver un ejemplo, consulta la integración de Deadline Cloud para Maya en. GitHub

La biblioteca de clientes de Deadline Cloud proporciona operaciones que hacen lo siguiente para ayudarlo a brindar una experiencia de usuario sólida e integrada:

- Extraiga los parámetros del entorno de colas, los parámetros de los trabajos y las referencias a los activos desde las variables de entorno y mediante una llamada al SDK de la aplicación.
- Establezca los parámetros en el paquete de tareas. Para evitar modificar el paquete original, debe hacer una copia del paquete y enviar la copia.

Si utiliza el `deadline bundle gui-submit` comando para enviar el paquete de tareas, debe utilizar los `asset_references.yaml` archivos `parameter_values.yaml` y mediante programación para pasar la información de la aplicación. Para obtener más información sobre estos archivos, consulte. Plantillas Open Job Description (OpenJD) para Deadline Cloud

Si necesita controles más complejos que los que ofrece OpenJD, necesita abstraer el trabajo del usuario o quiere que la integración se adapte al estilo visual de la aplicación, puede escribir su propio cuadro de diálogo que llame a la biblioteca de clientes de Deadline Cloud para enviar el trabajo.

Programe trabajos en Deadline Cloud

Después de crear un trabajo, AWS Deadline Cloud lo programa para que se procese en una o más de las flotas asociadas a una cola. La flota que procesa una tarea en particular se elige en función de la configuración de programación, las capacidades configuradas para la flota y los requisitos de hospedaje de un paso específico.

Las siguientes secciones proporcionan detalles del proceso de programación de un trabajo.

Configuraciones de programación

Puede configurar la forma en que Deadline Cloud programa los trabajos de una cola estableciendo una configuración de programación en la cola. La configuración de programación controla cómo se distribuyen los trabajadores entre los trabajos.

Puede establecer la configuración de la programación mediante la consola de Deadline Cloud o llamando al CreateQueuequirófano UpdateQueue APIs.

Hay tres configuraciones de programación disponibles:

- Prioridad, first-in-first-out (`priorityFifo`): programa primero el trabajo de mayor prioridad y el primero enviado (predeterminado).
- Prioridad, equilibrada (`priorityBalanced`): distribuye a los trabajadores de manera uniforme entre los trabajos de mayor prioridad.
- Ponderado, equilibrado (`weightedBalanced`): utiliza una fórmula ponderada para determinar cómo se distribuyen los trabajadores entre los puestos de trabajo.

En todas las configuraciones de programación, las tareas en curso se completan antes de que se tome una nueva decisión de programación. Si cambias la configuración de la programación mientras las tareas están en ejecución, el cambio solo se aplica cuando se asignen los siguientes trabajadores. Las tareas en ejecución no se interrumpen ni se reasignan.

Prioridad, first-in-first-out

La prioridad, first-in-first-out (`priorityFifo`) es la configuración de programación predeterminada para las colas nuevas. Deadline Cloud asigna primero a los trabajadores el trabajo de mayor prioridad. Cuando varios trabajos comparten la misma prioridad, el trabajo más antiguo (presentado más temprano) recibe primero a todos los trabajadores disponibles.

Utilice el FIFO prioritario cuando desee ordenar los trabajos de forma estricta. Esta configuración es adecuada cuando los trabajos deben completarse de uno en uno en el orden en que se enviaron, por ejemplo, en etapas de procesamiento secuenciales o en el procesamiento por lotes, donde cada trabajo debe finalizar antes de que comience el siguiente.

Esta configuración no tiene parámetros adicionales.

Prioridad, equilibrada

Priority, balanced (`priorityBalanced`) distribuye a los trabajadores de manera uniforme entre todos los puestos de trabajo con el nivel de prioridad más alto. Cuando solo existe un trabajo con la máxima prioridad, Deadline Cloud asigna a todos los trabajadores a ese trabajo. Cuando varios trabajos comparten la máxima prioridad, los trabajadores se dividen equitativamente entre ellos. Si los trabajadores no se pueden dividir en partes iguales, los trabajadores adicionales se distribuyen entre los trabajos de mayor prioridad.

Utilice el equilibrio de prioridades cuando varios artistas o usuarios envíen trabajos con la misma prioridad y cada usuario necesite comentarios inmediatos. Esta configuración garantiza que ningún trabajo monopolice a todos los trabajadores disponibles, de modo que se asignen trabajadores a todos los usuarios poco después de enviarlos.

Si a un trabajo le quedan menos tareas que la proporción de trabajadores que le corresponde, los trabajadores sobrantes se redistribuyen a otros trabajos con el mismo nivel de prioridad. Si todos los puestos de trabajo con la máxima prioridad se asignan en su totalidad, los trabajadores sobrantes se desplazan en cascada a los puestos del siguiente nivel de máxima prioridad.

Esta configuración tiene el siguiente parámetro:

`renderingTaskBuffer`

> Controla la adherencia de los trabajadores. Un trabajador cambia de su trabajo actual a otro con la misma prioridad solo si la diferencia en la representación de las tareas supera el `renderingTaskBuffer` valor. Un valor más alto mantiene a los trabajadores en sus puestos de trabajo actuales durante más tiempo, lo que reduce el cambio de contexto. El valor predeterminado es 1.

Ponderado y equilibrado

Weighted, balanced (`weightedBalanced`) utiliza una fórmula para calcular el peso de cada trabajo. Deadline Cloud asigna primero a los trabajadores el trabajo con mayor peso. Si varios trabajos tienen el mismo peso, los trabajadores se distribuyen entre ellos.

Utilice una ponderación equilibrada cuando necesite un control pormenorizado sobre la distribución de los trabajadores en los distintos trabajos, con diferentes prioridades, tasas de error y tiempos de entrega. Esta configuración es adecuada para entornos de granjas de renderizados complejas en los que desee ajustar el equilibrio entre la prioridad del trabajo, la antigüedad del trabajo, la gestión de errores y la dedicación de los trabajadores.

El peso de cada trabajo se calcula de la siguiente manera:

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

El `renderingTaskBuffer` componente se aplica solo si el trabajador está trabajando actualmente en el trabajo. Por lo general, `renderingTaskWeight` se establece en un valor negativo para que los trabajos con trabajadores asignados reciban una menor importancia, lo que hace que los demás trabajos ocupen el primer lugar de la lista. También `errorWeight` suele ser negativo, por lo que se pierde la prioridad de los trabajos con errores. Puede utilizar las anulaciones de programación para los trabajos de prioridad mínima y máxima.

Esta configuración tiene los siguientes parámetros:

`priorityWeight`

El peso que se aplica a la prioridad de un trabajo. Un valor positivo significa que los trabajos de mayor prioridad se programan primero. El valor predeterminado es `100.0`. Rango: 2 `0` a. `10000`

`errorWeight`

El peso aplicado al recuento de errores de un trabajo. Un valor negativo significa que los trabajos sin errores se programan primero. El valor predeterminado es `-10.0`. Rango: 2 `-10000` a`10000`.

`submissionTimeWeight`

El peso aplicado al tiempo de presentación de un trabajo (en segundos). Un valor positivo significa que los trabajos presentados anteriormente se programan primero. El valor predeterminado es `3.0`. Rango: 2 `0` a`10000`.

`renderingTaskWeight`

El peso aplicado al número de tareas que se están procesando actualmente para un trabajo. Un valor negativo significa que los próximos trabajos con menos trabajadores están programados. El valor predeterminado es `-100.0`. Rango: 2 `-10000` a`10000`.

`renderingTaskBuffer`

El número de tareas de renderizado antes de que surta efecto el peso de la tarea de renderizado. Un valor positivo mantiene a los trabajadores en sus puestos de trabajo actuales. El valor predeterminado es `1`. Rango: 2 `0` a`1000`.

`maxPriorityOverride`

Opcional. Si se establece en`alwaysScheduleFirst`, los trabajos con la máxima prioridad (100) siempre se programan antes que los demás trabajos, independientemente de la fórmula ponderada. Cuando varios trabajos tienen la máxima prioridad, los empates se rompen mediante la fórmula ponderada estándar. Cuando no existe la anulación, los trabajos de máxima prioridad utilizan la fórmula ponderada estándar sin ningún tratamiento especial.

`minPriorityOverride`

Opcional. Si se establece en`alwaysScheduleLast`, los trabajos con la prioridad mínima (0) siempre se programan después de otros trabajos, independientemente de la fórmula ponderada. Cuando varios trabajos tienen la prioridad mínima, los empates se rompen mediante la fórmula ponderada estándar. Cuando no existe la anulación, los trabajos de prioridad mínima utilizan la fórmula ponderada estándar sin ningún tratamiento especial.

Determine la compatibilidad de la flota

Tras crear un trabajo, Deadline Cloud compara los requisitos de alojamiento para cada paso del trabajo con las capacidades de las flotas asociadas a la cola a la que se envió el trabajo. Si una flota cumple con los requisitos de hospedaje, el trabajo pasa a manos del `READY` estado.

Si algún paso del trabajo tiene requisitos que una flota asociada a la cola no puede cumplir, el estado del paso se establece en. `NOT_COMPATIBLE` Además, el resto de los pasos del trabajo se cancelan.

Las capacidades de una flota se establecen a nivel de flota. Incluso si un trabajador de una flota cumple con los requisitos del trabajo, no se le asignarán tareas del trabajo si su flota no cumple con los requisitos del trabajo.

La siguiente plantilla de trabajo tiene un paso que especifica los requisitos de anfitrión para el paso:

```
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
```

Este trabajo se puede programar para una flota con las siguientes capacidades:

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

Este trabajo no se puede programar para una flota con ninguna de las siguientes capacidades:

```
{
    "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.
```

Escalado de flotas

Cuando se asigna un trabajo a una flota compatible gestionada por un servicio, la flota se escala automáticamente. El número de trabajadores de la flota cambia en función del número de tareas disponibles para la flota.

Cuando se asigna un trabajo a una flota gestionada por el cliente, es posible que ya existan trabajadores o que se puedan crear mediante el escalado automático basado en eventos. Para obtener más información, consulte Uso EventBridge para gestionar eventos de autoescalado en la Guía del usuario de Amazon EC2 Auto Scaling.

Sesiones

Las tareas de un trabajo se dividen en una o más sesiones. Los trabajadores dirigen las sesiones para configurar el entorno, ejecutar las tareas y, a continuación, desmantelar el entorno. Cada sesión se compone de una o más acciones que el trabajador debe realizar.

A medida que un trabajador completa las acciones de la sección, se le pueden enviar acciones de sesión adicionales. El trabajador reutiliza los entornos existentes y los adjuntos de trabajo en la sesión para completar las tareas de manera más eficiente.

En el caso de los trabajadores de flotas gestionados por el servicio, los directorios de las sesiones se eliminan una vez finalizada la sesión, pero los demás directorios se conservan entre sesiones. Este comportamiento le permite implementar estrategias de almacenamiento en caché para los datos que se pueden reutilizar en varias sesiones. Para almacenar en caché los datos entre sesiones, guárdelos en el directorio principal del usuario que ejecuta el trabajo. Por ejemplo, los paquetes conda se almacenan en caché en el directorio principal del usuario del trabajo, en `C:\Users\job-user\.conda-pkgs` on Windows workers y `/home/job-user/.conda-pkgs` on Linux workers. Estos datos permanecen disponibles hasta que el trabajador deje de trabajar.

El remitente crea los adjuntos de trabajo y los utilizas como parte de tu paquete de trabajos CLI de Deadline Cloud. También puede crear adjuntos de trabajo mediante la `--attachments` opción del `create-job` AWS CLI comando. Los entornos se definen en dos lugares: los entornos de cola adjuntos a una cola específica y los entornos de tareas y escalones definidos en la plantilla de trabajos.

Hay cuatro tipos de acciones de sesión:

- `syncInputJobAttachments`— Descarga los archivos adjuntos al trabajo de entrada para el trabajador.

- `envEnter`— Realiza las `onEnter` acciones de un entorno.
- `taskRun`— Realiza las `onRun` acciones de una tarea.
- `envExit`— Realiza las `onExit` acciones para un entorno.

La siguiente plantilla de trabajo tiene un entorno escalonado. Cuenta con una `onEnter` definición para configurar el entorno escalonado, una `onRun` definición que define la tarea que se va a ejecutar y una `onExit` definición para desmantelar el entorno escalonado. Las sesiones creadas para este trabajo incluirán una `envEnter` acción, una o más `taskRun` acciones y, a continuación, una `envExit` acción.

```
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 }}
```

Canalización de las acciones de la sesión

La canalización de acciones de sesión permite a un programador preasignar varias acciones de sesión a un trabajador. A continuación, el trabajador puede ejecutar estas acciones de forma secuencial, lo que reduce o elimina el tiempo de inactividad entre tareas.

Para crear una asignación inicial, el planificador crea una sesión con una tarea, el trabajador completa la tarea y, a continuación, el planificador analiza la duración de la tarea para determinar las futuras asignaciones.

Para que el programador sea efectivo, existen reglas de duración de las tareas. Para las tareas de menos de un minuto, el programador utiliza un patrón de crecimiento de 2 potencias. Por ejemplo, para una tarea de 1 segundo, el planificador asigna 2 tareas nuevas, luego 4 y luego 8. Para las tareas de más de un minuto, el planificador asigna solo una nueva tarea y la canalización permanece desactivada.

Para calcular el tamaño de la canalización, el programador hace lo siguiente:

- Utiliza la duración media de las tareas completadas
- Su objetivo es mantener ocupado al trabajador durante un minuto
- Considera solo las tareas de la misma sesión
- No comparte los datos de duración entre los trabajadores

Con la canalización de las acciones de la sesión, los trabajadores comienzan nuevas tareas de forma inmediata y no hay tiempo de espera entre las solicitudes del programador. También proporciona una mayor eficiencia de los trabajadores y una mejor distribución de las tareas para los procesos de larga duración.

Además, si hay disponible un nuevo trabajo de mayor prioridad, el trabajador terminará todo el trabajo que se le asignó anteriormente antes de que finalice la sesión actual y se le asigne una nueva sesión de un trabajo de mayor prioridad.

Dependencias escalonadas

Deadline Cloud permite definir las dependencias entre los pasos, de modo que un paso espere a que se complete otro paso antes de empezar. Puedes definir más de una dependencia para un paso. Un paso con una dependencia no se programa hasta que todas sus dependencias estén completas.

Si la plantilla de trabajo define una dependencia circular, el trabajo se rechaza y su estado se establece en. `CREATE_FAILED`

La siguiente plantilla de trabajo crea un trabajo en dos pasos. `StepB`depende de`StepA`. `StepB`solo se ejecuta después de que `StepA` se complete correctamente.

Una vez creado el trabajo, `StepA` se encuentra en el `READY` estado y `StepB` se encuentra en el `PENDING` estado. Una vez `StepA` finalizado, `StepB` pasa al `READY` estado. Si `StepA` falla o `StepA` se cancela, `StepB` pasa al `CANCELED` estado.

Puede establecer una dependencia en varios pasos. Por ejemplo, si `StepC` depende de ambos `StepA` pasos`StepB`, `StepC` no empezará hasta que finalicen los otros dos pasos.

Las dependencias escalonadas tienen las siguientes restricciones:

- Dependencias por paso: un paso puede depender de un máximo de otros 128 pasos.
- Consumidores por paso: un máximo de otros 32 pasos pueden depender de un solo paso.

```
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!
```

Modificar un trabajo en Deadline Cloud

Puede usar los siguientes `update` comandos AWS Command Line Interface (AWS CLI) para modificar la configuración de un trabajo o para establecer el estado objetivo de un trabajo, paso o tarea:

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

En los siguientes ejemplos de `update` comandos, sustituya cada uno *`user input placeholder`* por su propia información.

Example— Volver a poner en cola un trabajo

Todas las tareas del trabajo cambian al `READY` estado, a menos que haya dependencias entre pasos. Los pasos con dependencias cambian a uno `READY` o a `PENDING` medida que se restauran.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status PENDING
```

Example— Cancelar un trabajo

Todas las tareas del trabajo que no tienen el estado `SUCCEEDED` o `FAILED` están marcadas`CANCELED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example— Marcar un trabajo fallido

Todas las tareas del trabajo que tienen ese estado `SUCCEEDED` permanecen sin cambios. Todas las demás tareas están marcadas`FAILED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example— Marcar un trabajo como exitoso

Todas las tareas del trabajo se trasladan al `SUCCEEDED` estado.

```
aws deadline update-job \
--farm-id farmID \
```

```
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example— Suspender un trabajo

Las tareas del trabajo en el `FAILED` estado `SUCCEEDEDCANCELED`, o no cambian. Todas las demás tareas están marcadas`SUSPENDED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUSPENDED
```

Example— Cambiar la prioridad de un trabajo

Actualiza la prioridad de un trabajo en una cola para cambiar el orden en que está programado. Por lo general, los trabajos de mayor prioridad se programan primero.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example— Cambiar el número de tareas fallidas permitidas

Actualiza el número máximo de tareas fallidas que puede tener el trabajo antes de que se cancelen las tareas restantes.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example— Cambia el número de reintentos de tareas permitidos

Actualiza el número máximo de reintentos de una tarea antes de que se produzca un error en la tarea. Una tarea que ha alcanzado el número máximo de reintentos no se puede volver a poner en cola hasta que se aumente este valor.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example— Archivar un trabajo

Actualiza el estado del ciclo de vida del trabajo a`ARCHIVED`. Los trabajos archivados no se pueden programar ni modificar. Solo puede archivar un trabajo que se encuentre en el `SUSPENDED` estado `FAILEDCANCELED`,`SUCCEEDED`, o.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example— Cambiar el nombre de un trabajo

Actualiza el nombre para mostrar de un trabajo. El nombre del trabajo puede tener hasta 128 caracteres.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example— Cambiar la descripción de un trabajo

Actualiza la descripción de un trabajo. La descripción puede tener una longitud máxima de 2048 caracteres. Para eliminar la descripción existente, pase una cadena vacía.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example— Volver a poner en cola un paso

Todas las tareas del paso cambian al `READY` estado, a menos que haya dependencias entre pasos. Las tareas de los pasos con dependencias cambian a uno `READY` o `PENDING` varios pasos y la tarea se restaura.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example— Cancelar un paso

Todas las tareas del paso que no tienen el estado `SUCCEEDED` o `FAILED` están marcadas`CANCELED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status CANCELED
```

Example— Marcar un paso como fallido

Todas las tareas del paso que tienen ese estado `SUCCEEDED` permanecen sin cambios. Todas las demás tareas están marcadas`FAILED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example— Marcar un paso como exitoso

Todas las tareas del paso están marcadas`SUCCEEDED`.

```
aws deadline update-step \
```

```
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example— Suspender un paso

Las tareas del paso en el `FAILED` estado `SUCCEEDEDCANCELED`, o no cambian. Todas las demás tareas están marcadas`SUSPENDED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example— Cambiar el estado de una tarea

Al utilizar el comando CLI de `update-task` Deadline Cloud, la tarea cambia al estado especificado.

```
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
```

Cree y utilice flotas gestionadas por los clientes de Deadline Cloud

Cuando crea una flota gestionada por el cliente (CMF), tiene el control total sobre su proceso de procesamiento. Usted define el entorno de red y software para cada trabajador. Deadline Cloud actúa como repositorio y programador de sus trabajos.

Un trabajador puede ser una instancia de Amazon Elastic Compute Cloud (Amazon EC2), un trabajador de una instalación de ubicación conjunta o un trabajador local. Cada trabajador debe ejecutar el agente de trabajo de Deadline Cloud. Todos los trabajadores deben tener acceso al punto final del servicio Deadline Cloud.

En los temas siguientes, se muestra cómo crear una CMF básica mediante instancias de Amazon EC2.

Temas

Cree una flota gestionada por el cliente

Para crear una flota gestionada por el cliente (CMF), complete los siguientes pasos.

Deadline Cloud console

Para usar la consola de Deadline Cloud para crear una flota gestionada por el cliente

1. Abre la consola de Deadline Cloud.
2. Selecciona Farms. Aparece una lista de las granjas disponibles.
3. Seleccione el nombre de la granja en la que desea trabajar.
4. Selecciona la pestaña Flotas y, a continuación, selecciona Crear flota.
5. Introduce un nombre para tu flota.
6. (Opcional) Introduzca una descripción para su flota.
7. Seleccione Gestionado por el cliente para el tipo de flota.
8. Seleccione el acceso al servicio de su flota.

    a. Te recomendamos que utilices la opción Crear y usar una nueva función de servicio para cada flota para controlar los permisos de forma más detallada. Esta opción está seleccionada de forma predeterminada.

    b. También puede usar un rol de servicio existente seleccionando Elegir un rol de servicio.
9. Revisa tus selecciones y, a continuación, selecciona Siguiente.
10. Seleccione un sistema operativo para su flota. Todos los trabajadores de una flota deben tener un sistema operativo común.
11. Seleccione la arquitectura de la CPU del host.
12. Seleccione las capacidades de hardware de memoria y vCPU mínimas y máximas para satisfacer las demandas de carga de trabajo de sus flotas.
13. Seleccione un tipo de Auto Scaling. Para obtener más información, consulte Uso EventBridge para gestionar eventos de Auto Scaling.

    - Sin escalado: está creando una flota local y quiere excluirse de Deadline Cloud Auto Scaling.
    - Recomendaciones de escalado: está creando una flota de Amazon Elastic Compute Cloud (Amazon EC2).
14. (Opcional) Seleccione la flecha para expandir la sección Agregar capacidades.
15. (Opcional) Selecciona la casilla Añadir capacidad de GPU (opcional) y, a continuación, introduce el mínimo y el máximo GPUs y la memoria.
16. Revisa tus selecciones y, a continuación, selecciona Siguiente.
17. (Opcional) Defina las capacidades de trabajo personalizadas y, a continuación, seleccione Siguiente.
18. En el menú desplegable, selecciona una o más colas para asociarlas a la flota.

> **Note**
>
> Recomendamos asociar una flota únicamente a las colas que estén todas en el mismo límite de confianza. Esta recomendación garantiza un límite de seguridad sólido entre los trabajos en ejecución del mismo trabajador.

19. Revise las asociaciones de colas y, a continuación, seleccione Siguiente.
20. (Opcional) Para el entorno de colas de Conda predeterminado, crearemos un entorno para su cola en el que se instalarán los paquetes de conda solicitados por los trabajos.

> **Note**
>
> El entorno de colas conda se utiliza para instalar los paquetes conda solicitados por los trabajos. Normalmente, deberías desmarcar el entorno de colas conda en las colas asociadas, CMFs ya que no CMFs tendrás instalados los comandos conda necesarios de forma predeterminada.

21. (Opcional) Agrega etiquetas a tu CMF. Para obtener más información, consulte Etiquetar AWS los recursos.
22. Revisa la configuración de tu flota y realiza los cambios necesarios y, a continuación, selecciona Crear flota.
23. Selecciona la pestaña Flotas y, a continuación, anota el ID de la flota.

AWS CLI

Para utilizarla AWS CLI para crear una flota gestionada por el cliente

1. Abra un terminal.
2. Crea `fleet-trust-policy.json` en un editor nuevo.
   a. Agrega la siguiente política de IAM y reemplaza el *`ITALICIZED`* texto por tu ID de AWS cuenta y tu ID de granja de 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. Guarde los cambios.

3. Cree `fleet-policy.json`.

   a. Añada la siguiente política de 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. Guarde los cambios.

4. Añada una función de IAM para que la usen los trabajadores de su flota.

```
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. Cree `create-fleet-request.json`.

a. Añada la siguiente política de IAM y sustituya el texto en cursiva por los valores de su CMF.

> **Note**
>
> Puede encontrarlo en. *ROLE_ARN* `create-cmf-fleet.json`
> Para el*OS_FAMILY*, debe elegir uno `delinux`, `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. Guarde los cambios.

6. Crea tu flota.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

Instalación y configuración del host de trabajo

Un anfitrión de trabajo se refiere a una máquina host que ejecuta un servidor de Deadline Cloud. En esta sección se explica cómo configurar el host de trabajo y configurarlo para sus necesidades específicas. Cada host de trabajo ejecuta un programa denominado agente de trabajo. El agente obrero es responsable de:

- Gestionar el ciclo de vida del trabajador.
- Sincronizar el trabajo asignado, su progreso y sus resultados.
- Supervisión del trabajo en ejecución.
- Reenviar los registros a los destinos configurados.

Le recomendamos que utilice el agente de trabajo de Deadline Cloud proporcionado. El agente de trabajo es de código abierto y te recomendamos que solicites funciones, pero también puedes desarrollarlo y personalizarlo para que se adapte a tus necesidades.

Para completar las tareas de las siguientes secciones, necesitará lo siguiente:

Linux

- Una instancia Linux basada en Amazon Elastic Compute Cloud (Amazon EC2). Recomendamos Amazon Linux 2023.
- `sudo`privilegios
- Python 3.9 o superior

Windows

- Una instancia Windows basada en Amazon Elastic Compute Cloud (Amazon EC2). RecomendamosWindows Server 2022.
- Acceso de administrador al anfitrión del trabajador
- Python 3.9 o superior instalado para todos los usuarios

Crear y configurar un entorno virtual de Python

Puede crear un entorno virtual de Python Linux si ha instalado Python 3.9 o superior y lo ha colocado en suPATH.

> **Note**
>
> SíWindows, los archivos del agente deben estar instalados en el directorio global de paquetes de sitios de Python. Los entornos virtuales de Python no son compatibles actualmente.

Para crear y activar un entorno virtual de Python

1. Abra una terminal como `root` usuario (o utilice`sudo`/`su`).
2. Cree y active un entorno virtual de Python.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Instale el agente de trabajo de Deadline Cloud

Una vez que hayas configurado tu Python y creado un entorno virtualLinux, instala los paquetes Python del agente de trabajo de Deadline Cloud.

Para instalar los paquetes de Python del agente de trabajo

Linux

1. Abra una terminal como `root` usuario (o utilice`sudo`/`su`).
2. Descarga e instala los paquetes de agentes de trabajo de Deadline Cloud desde PyPI:

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. Abre una línea de comandos o PowerShell una terminal de administrador.

2. Descarga e instala los paquetes de agentes de trabajo de Deadline Cloud desde PyPI:

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

Si su anfitrión Windows de trabajo requiere nombres de ruta largos (más de 250 caracteres), debe habilitar los nombres de ruta largos de la siguiente manera:

Para habilitar rutas largas para los hosts Windows de trabajo

1. Asegúrese de que la clave de registro de ruta larga esté habilitada. Para obtener más información, consulte Configuración del registro para habilitar las rutas de registro en el sitio web de Microsoft.
2. Instale el Windows SDK para aplicaciones C++ x86 de escritorio. Para obtener más información, consulte el WindowsSDK en el Centro de Windows desarrolladores.
3. Abra la ubicación de instalación de Python en su entorno donde está instalado el agente de trabajo. El valor predeterminado es `C:\Program Files\Python311`. Hay un archivo ejecutable llamado`pythonservice.exe`.
4. Cree un nuevo archivo llamado `pythonservice.exe.manifest` en la misma ubicación. Añada lo siguiente:

```
<?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. Abra una línea de comandos y ejecute el siguiente comando en la ubicación del archivo de manifiesto que creó:

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

Debería ver una salida similar a esta:

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

El trabajador ahora puede acceder a rutas largas. Para limpiarlo, elimine el `pythonservice.exe.manifest` archivo y desinstale el SDK.

Configura el agente de trabajo de Deadline Cloud

Puede configurar los ajustes del agente de trabajo de Deadline Cloud de tres maneras. Le recomendamos que utilice la configuración del sistema operativo ejecutando la `install-deadline-worker` herramienta.

El agente de trabajo no admite la ejecución como usuario de dominio en Windows. Para ejecutar un trabajo como usuario de dominio, puede especificar una cuenta de usuario de dominio al configurar un usuario de cola para ejecutar trabajos. Para obtener más información, consulta el paso 7 de las colas de Deadline Cloud de la Guía del usuario de AWS Deadline Cloud.

Argumentos de línea de comandos: puede especificar argumentos al ejecutar el agente de trabajo de Deadline Cloud desde la línea de comandos. Algunos ajustes de configuración no están disponibles a través de los argumentos de la línea de comandos. Para ver todos los argumentos de la línea de comandos disponibles, introduzca`deadline-worker-agent --help`.

Variables de entorno: puede configurar el agente de trabajo de Deadline Cloud configurando la variable de entorno que comience por`DEADLINE_WORKER_`. Por ejemplo, para ver todos los argumentos de la línea de comandos disponibles, puede utilizar `export DEADLINE_WORKER_VERBOSE=true` para configurar el resultado del agente de trabajo en un formato detallado. Para obtener más ejemplos e información, consulte `/etc/amazon/deadline/worker.toml.example` en Linux o `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` en. Windows

Archivo de configuración: al instalar el agente de trabajo, se crea un archivo de configuración ubicado `/etc/amazon/deadline/worker.toml` en Linux o `C:\ProgramData\Amazon\Deadline\Config\worker.toml` onWindows. El agente de trabajo carga este archivo de configuración cuando se inicia. Puede usar el archivo de configuración de ejemplo (`/etc/amazon/deadline/worker.toml.example`activado Linux o `C:\ProgramData\Amazon\Deadline\Config`

`\worker.toml.example` activadoWindows) para adaptar el archivo de configuración del agente de trabajo predeterminado a sus necesidades específicas.

Por último, le recomendamos que habilite el apagado automático del agente de trabajo una vez que el software se haya implementado y funcione según lo esperado. Esto permite que la flota de trabajadores se amplíe cuando sea necesario y se cierre cuando finalice un trabajo. El escalado automático ayuda a garantizar que solo utilice los recursos necesarios. Para permitir que una instancia iniciada por el grupo de autoescalado se cierre, debe agregarla `shutdown_on_stop=true` al archivo `worker.toml` de configuración.

Para habilitar el apagado automático

Como **`root`** usuario:

- Instale el agente de trabajo con los parámetros**`--allow-shutdown`**.

  Linux

  Escriba:

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

  Windows

  Escriba:

  ```
  install-deadline-worker ^
    --farm-id FARM_ID ^
    --fleet-id FLEET_ID ^
    --region REGION ^
    --allow-shutdown
  ```

Cree usuarios y grupos de trabajo

En esta sección se describe la relación de usuario y grupo necesaria entre el usuario agente y la `jobRunAsUser` definida en sus colas.

El agente de trabajo de Deadline Cloud debe funcionar como un usuario dedicado a un agente específico en el host. Debe configurar la `jobRunAsUser` propiedad de las colas de Deadline Cloud para que los trabajadores ejecuten las tareas de cola como un usuario y un grupo específicos del sistema operativo. Esta configuración significa que puedes controlar los permisos del sistema de archivos compartidos que tienen tus trabajos. También proporciona un importante límite de seguridad entre sus trabajos y el usuario del agente de trabajo.

Linuxusuarios y grupos del trabajo

Para configurar un usuario de agente de trabajo local`jobRunAsUser`, asegúrese de cumplir los siguientes requisitos. Si utiliza un módulo de autenticación conectable (PAM) de Linux, como Active Directory o LDAP, el procedimiento puede ser diferente.

El usuario del agente de trabajo y el `jobRunAsUser` grupo compartido se configuran al instalar el agente de trabajo. Los valores predeterminados son `deadline-worker-agent` y`deadline-job-users`, pero puede cambiarlos al instalar el agente de trabajo.

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

Los comandos se deben ejecutar como usuario root.

- Cada uno `jobRunAsUser` debe tener un grupo principal coincidente. Al crear un usuario con el `adduser` comando, normalmente se crea un grupo principal coincidente.

  ```
  adduser -r -m jobRunAsUser
  ```

- El grupo principal de `jobRunAsUser` es un grupo secundario para el usuario del agente de trabajo. El grupo compartido permite al agente de trabajo poner los archivos a disposición del trabajo mientras se ejecuta.

  ```
  usermod -a -G jobRunAsUser deadline-worker-agent
  ```

- `jobRunAsUser`Debe ser miembro del grupo de trabajos compartidos.

  ```
  usermod -a -G deadline-job-users jobRunAsUser
  ```

- No `jobRunAsUser` debe pertenecer al grupo principal del usuario del agente de trabajo. Los archivos confidenciales escritos por el agente de trabajo son propiedad del grupo principal del

agente. Si a `jobRunAsUser` forma parte de este grupo, los trabajos que se estén ejecutando en el trabajador pueden acceder a los archivos del agente trabajador.

- El valor predeterminado Región de AWS debe coincidir con la región de la granja a la que pertenece el trabajador. Esto debe aplicarse a todas `jobRunAsUser` las cuentas del trabajador.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

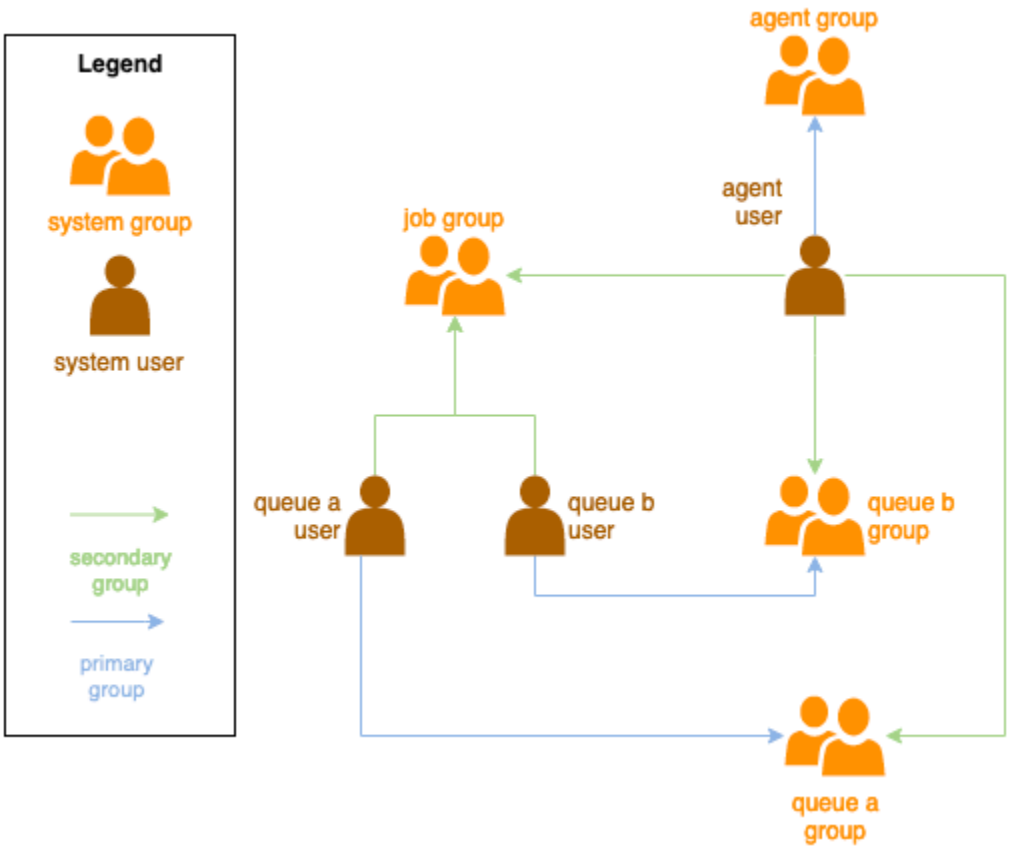
- El usuario del agente de trabajo debe poder ejecutar `sudo` comandos como`jobRunAsUser`. Ejecute el siguiente comando para abrir un editor y crear una nueva regla de sudoers:

```
visudo -f /etc/sudoers.d/deadline-worker-job-user
```

Agregue lo siguiente al archivo:

```
# 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
```

El siguiente diagrama ilustra la relación entre el usuario agente y los `jobRunAsUser` usuarios y grupos de las colas asociadas a la flota.



Usuarios de Windows

Para utilizar un Windows usuario como`jobRunAsUser`, debe cumplir los siguientes requisitos:

- Deben existir todos `jobRunAsUser` los usuarios de la cola.
- Sus contraseñas deben coincidir con el valor del secreto especificado en el campo de la `JobRunAsUser` cola. Para obtener instrucciones, consulta el paso 7 de las colas de Deadline Cloud de la Guía del usuario de AWS Deadline Cloud.
- El agente-usuario debe poder iniciar sesión como esos usuarios.

Proteger el host de su trabajador

Al configurar su servidor de trabajo, siga las mejores prácticas de seguridad para proteger la información confidencial y mantener los controles de acceso adecuados.

Configurar los permisos de las carpetas de registro

El agente de trabajo escribe archivos de registro que pueden contener información confidencial de los scripts de configuración del host y de la ejecución de tareas. El `install-deadline-worker` comando crea el directorio de registro con permisos seguros. Si necesita crear el directorio manualmente antes de la instalación, utilice los siguientes procedimientos para que coincidan con los permisos que utilizan las flotas gestionadas por el servicio:

Linux

Para configurar los permisos del directorio de registro en Linux

1. Cree el directorio de registro:

```
sudo mkdir -p /var/log/amazon/deadline
```

2. Defina el propietario y el grupo como el usuario del agente de trabajo:

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. Establezca los permisos en 750:

```
sudo chmod -R 750 /var/log/amazon/deadline
```

Estos permisos garantizan que solo el agente de trabajo, el usuario y el grupo puedan acceder a los archivos de registro, lo que impide que los usuarios del trabajo y otros usuarios no autorizados lean información potencialmente confidencial.

Windows

Para configurar los permisos del directorio de registro en Windows

1. Abra un PowerShell terminal de administrador.
2. Cree el directorio de registro:

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. Configure la configuración restringida ACLs para permitir que solo el usuario del agente de trabajo y los administradores:

```
$acl = Get-Acl "$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
```

Estos comandos restringen el acceso al directorio de registro únicamente al usuario del agente de trabajo y al grupo de administradores, lo que impide que los usuarios del trabajo y otros usuarios no autorizados lean información potencialmente confidencial.

Gestione el acceso a los secretos de los usuarios del Windows trabajo

Al configurar una cola con un Windows`jobRunAsUser`, debe especificar un secreto de AWS Secrets Manager. Se espera que el valor de este secreto sea un objeto codificado en JSON del siguiente formato:

```
{
    "password": "JOB_USER_PASSWORD"
}
```

Para que los trabajadores puedan ejecutar los trabajos tal y como está configurada la cola`jobRunAsUser`, la función de IAM de la flota debe tener permisos para obtener el valor del secreto. Si el secreto se cifra con una clave de KMS gestionada por el cliente, la función de IAM de la flota también debe tener permisos para descifrarlo mediante la clave de KMS.

Se recomienda encarecidamente seguir el principio del mínimo privilegio para estos secretos. Esto significa que el acceso para obtener el valor secreto de una cola → → debería ser: `jobRunAsUser` `windows` `passwordArn`

- se otorga a un rol de flota cuando se crea una asociación de cola y flota entre la flota y la cola

- se revoca de un rol de flota cuando se elimina una asociación de cola y flota entre la flota y la cola

Además, el secreto de AWS Secrets Manager que contiene la `jobRunAsUser` contraseña debe eliminarse cuando ya no se utilice.

Conceda acceso a una contraseña secreta

Las flotas de Deadline Cloud necesitan acceder a la `jobRunAsUser` contraseña almacenada en la contraseña secreta de la cola cuando se asocian la cola y la flota. Recomendamos utilizar la política de recursos de AWS Secrets Manager para conceder acceso a las funciones de la flota. Si sigues estrictamente esta directriz, es más fácil determinar qué roles de flota tienen acceso al secreto.

Para conceder acceso al secreto

1. Abre la consola de AWS Secret Manager para acceder al secreto.
2. En la sección Permisos de recursos, añada una declaración de política del siguiente formato:

   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 el acceso a una contraseña secreta

Cuando una flota ya no necesite acceder a una cola, elimina el acceso a la contraseña secreta de la cola. `jobRunAsUser` Recomendamos utilizar la política de recursos de AWS Secrets Manager para conceder acceso a las funciones de la flota. Si sigues estrictamente esta directriz, es más fácil determinar qué roles de flota tienen acceso al secreto.

Para revocar el acceso al secreto

1. Abre la consola de AWS Secret Manager para acceder al secreto.
2. En la sección Permisos de recursos, elimine la declaración de política del formulario:

   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"
    }
  ]
}
```

Instalar y configurar el software necesario para los trabajos

Después de configurar el agente de trabajo de Deadline Cloud, puede preparar el anfitrión del trabajador con cualquier software necesario para ejecutar los trabajos.

Cuando envías un trabajo a una cola con un asociado`jobRunAsUser`, el trabajo se ejecuta como ese usuario. Cuando se envía un trabajo con comandos que no son una ruta absoluta, ese comando debe encontrarse en la PATH de ese usuario.

En Linux, puede especificar el valor PATH para un usuario en una de las siguientes opciones:

- su `~/.bashrc` o `~/.bash_profile`
- archivos de configuración del sistema, como `/etc/profile.d/*` y `/etc/profile`
- scripts de inicio de shell:`/etc/bashrc`.

En Windows, puede especificar el PATH para un usuario en una de las siguientes opciones:

- sus variables de entorno específicas del usuario
- las variables de entorno de todo el sistema

Instale adaptadores para herramientas de creación de contenido digital

Deadline Cloud proporciona OpenJobDescription adaptadores para usar aplicaciones populares de creación de contenido digital (DCC). Para utilizar estos adaptadores en una flota gestionada por el cliente, debe instalar el software DCC y los adaptadores de la aplicación. A continuación, asegúrese de que los programas ejecutables del software estén disponibles en la ruta de búsqueda del sistema (por ejemplo, en la variable de entorno). PATH

Para instalar adaptadores DCC en una flota gestionada por el cliente

1. Abra el terminal A.
    a. En Linux, abre un terminal como `root` usuario (o usa`sudo`/`su`)
    b. ActivadoWindows, abra una línea de comandos o una PowerShell terminal de administrador.
2. Instale los paquetes de adaptadores de Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

Configuración de AWS credenciales

La fase inicial del ciclo de vida del trabajador es el arranque. En esta fase, el software de agente obrero crea un trabajador en la flota y obtiene AWS las credenciales del rol de la flota para continuar con las operaciones.

AWS credentials for Amazon EC2

Para crear un rol de IAM para Amazon EC2 con permisos de host de trabajadores de Deadline Cloud

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, selecciona Roles en el panel de navegación y, a continuación, selecciona Crear rol.
3. Seleccione el servicio de AWS .
4. Seleccione EC2 como servicio o caso de uso y, a continuación, seleccione Siguiente.
5. Para conceder los permisos necesarios, adjunte la política `AWSDeadlineCloud-WorkerHost` AWS gestionada.

On-premises AWS credentials

Sus trabajadores locales utilizan las credenciales para acceder a Deadline Cloud. Para un acceso más seguro, te recomendamos que utilices IAM Roles Anywhere para autenticar a tus trabajadores. Para obtener más información, consulte IAM Roles Anywhere.

Para realizar las pruebas, puede utilizar las claves de acceso de los usuarios de IAM como credenciales. AWS Le recomendamos que establezca una fecha de caducidad para el usuario de IAM mediante la inclusión de una política interna restrictiva.

> **Important**
>
> Preste atención a las siguientes advertencias:
>
> - NO utilices las credenciales raíz de tu cuenta para acceder AWS a los recursos. Estas credenciales proporcionan acceso ilimitado a la cuenta y son difíciles de revocar.
> - NO incluya claves de acceso literales ni información sobre credenciales en sus archivos de aplicación. Si lo hace, puede crear un riesgo de exposición accidental de sus credenciales si, por ejemplo, carga el proyecto en un repositorio público.
> - NO incluya archivos que contengan credenciales en el área del proyecto.
> - Proteja sus claves de acceso. No proporcione sus claves de acceso a terceros no autorizados, ni siquiera para que le ayuden a buscar sus identificadores de cuenta. De este modo, podrías dar a alguien acceso permanente a tu cuenta.

- Ten en cuenta que todas las credenciales almacenadas en el archivo de AWS credenciales compartidas se guardan en texto plano.

Para obtener más información, consulte las prácticas recomendadas para administrar las claves de AWS acceso en la Referencia AWS general.

Creación de un usuario de IAM

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Usuarios y, a continuación, seleccione Crear usuario.
3. Asigne un nombre al usuario. Desactive la casilla de verificación Proporcionar acceso de usuario al y, a continuación Consola de administración de AWS, seleccione Siguiente.
4. Seleccione Asociar políticas directamente.
5. En la lista de políticas de permisos, elija la AWSDeadlineCloud-WorkerHostpolítica y, a continuación, elija Siguiente.
6. Revisa los detalles del usuario y, a continuación, selecciona Crear usuario.

Restricción del acceso del usuario a un período limitado

Todas las claves de acceso de usuario de IAM que cree son credenciales a largo plazo. Para garantizar que estas credenciales caduquen en caso de que se usen de forma incorrecta, puede establecer un límite de tiempo para estas credenciales creando una política insertada que especifique una fecha a partir de la cual las claves dejarán de ser válidas.

1. Abra el usuario de IAM que acaba de crear. En la pestaña Permisos, selecciona Añadir permisos y, a continuación, selecciona Crear política integrada.
2. En el editor JSON, especifique los siguientes permisos. Para usar esta política, sustituye el valor de la `aws:CurrentTime` marca de tiempo de la política de ejemplo por tu propia fecha y hora.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Creación de una clave de acceso

1. En la página de detalles del usuario, selecciona la pestaña Credenciales de seguridad. En la sección Claves de acceso, haga clic en Crear clave de acceso.
2. Indique que desea utilizar la clave para Otros, seleccione Siguiente y, a continuación, seleccione Crear clave de acceso.
3. En la página Recuperar claves de acceso, selecciona Mostrar para ver el valor de la clave de acceso secreta de tu usuario. Puede copiar las credenciales o descargar un archivo .csv.

Guarde las claves de acceso del usuario

- Guarde las claves de acceso de los usuarios en el archivo de AWS credenciales del usuario agente en el sistema host de trabajo:
  - SíLinux, el archivo se encuentra en `~/.aws/credentials`
  - SíWindows, el archivo se encuentra en `%USERPROFILE\.aws\credentials`

  Sustituya las siguientes teclas:

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> Cuando ya no necesite este usuario de IAM, le recomendamos que lo quite para adaptarlo a las mejores prácticas AWS de seguridad. Le recomendamos que exija a sus usuarios humanos que utilicen credenciales temporales para AWS IAM Identity Centeracceder AWS.

Flujo de datos de host de trabajadores para flotas gestionadas por el cliente

En este tema se describen las conexiones de red que los anfitriones de trabajadores de AWS Deadline Cloud (Deadline Cloud) establecen durante la operación, incluidos los puntos finales contactados, los protocolos utilizados y los datos transmitidos. Esta información se aplica a los trabajadores de flotas gestionadas por el cliente (CMF), incluidas las instancias de Amazon Elastic Compute Cloud (Amazon EC2) y los trabajadores locales. Utilice esta información para configurar las reglas de firewall, crear puntos finales de VPC, realizar auditorías de seguridad o planificar políticas de red para los hosts de sus trabajadores. Para obtener información sobre las redes de flotas gestionadas por servicios, consulte. Inter-network privacidad del tráfico

Todas las comunicaciones de los trabajadores son únicamente salientes. Los anfitriones de los trabajadores inician todas las conexiones; no es necesario permitir ninguna conexión entrante. Todas las conexiones utilizan HTTPS (TLS 1.2 o posterior) a través del puerto 443.

En este tema se incluyen las siguientes secciones:

- the section called “Puntos finales y protocolos”
- the section called “Operaciones de API utilizadas por los trabajadores”
- the section called “Otros datos transmitidos”
- the section called “Opciones de conectividad privada”

Puntos finales y protocolos

En la siguiente tabla se enumeran los puntos finales AWS de servicio a los que se conectan los hosts de trabajo durante la operación. Para obtener la lista completa de los puntos de enlace regionales de cada servicio, consulte los puntos de enlace y las cuotas del servicio en la AWS Referencia general.

Referencia del punto final del servidor trabajador

AWS servicio	Punto de conexión	Puerto/protocolo	Finalidad	Obligatorio
Deadline Cloud (programación)	`scheduling.deadline. [Region].amazonaws.com`	443/ HTTPS	Registro de trabajadores, sondeo de tareas, actualizaciones de estado, intercambio de credenciales, recuperación de entidades laborales. Consulte the section called “Operaciones de API utilizadas por los trabajadores”.	Siempre
Amazon CloudWatch Logs (CloudWatch registros)	`logs.[Region].amazonas.com`	443/ HTTPS	Entrega de un agente de trabajo y un registro de sesión.	Siempre
Amazon Simple Storage Service (Amazon S3)	`s3.[Region].amazonaws.com`	443/ HTTPS	Carga y descarga de archivos adjuntos de trabajo.	Si usa adjuntos de trabajo

Si sus trabajos utilizan otros AWS servicios, es posible que también necesite permitir las conexiones salientes a esos puntos finales del servicio.

Operaciones de API utilizadas por los trabajadores

Todas las siguientes operaciones de API utilizan el `scheduling.deadline.[Region].amazonaws.com` punto final. Para ver los esquemas completos de solicitud y respuesta de cada operación, consulta la referencia de la API de Deadline Cloud.

Fase de Bootstrap

Cuando comienza un anfitrión obrero, el agente obrero se registra en la flota. Las credenciales de arranque requieren los permisos de la política `AWSDeadlineCloud-WorkerHost` AWS gestionada o permisos personalizados equivalentes. La fase de arranque utiliza las siguientes operaciones de API:

- `CreateWorker`— Registra al trabajador en la flota. Envía el nombre del host y las direcciones IP. Recibe un identificador de trabajador.
- `AssumeFleetRoleForWorker`— Obtiene las credenciales de rol de la flota. Recibe AWS las credenciales temporales que el agente trabajador utiliza para las operaciones posteriores.

Fase operativa

Tras el arranque, el agente obrero sondea para las sesiones de trabajo y procesa. El rol de flota requiere los permisos de la política `AWSDeadlineCloud-FleetWorker` AWS gestionada, o permisos personalizados equivalentes, y utiliza las siguientes operaciones de API:

- `UpdateWorker`— Actualiza el estado del trabajador, por ejemplo, `STOPPED` durante el cierre.
- `UpdateWorkerSchedule`— Encuestas para las asignaciones de trabajo. Envía actualizaciones del estado de las acciones de la sesión, que incluyen el estado de finalización, el porcentaje de progreso, el mensaje de progreso y los códigos hash del manifiesto de salida. Recibe las sesiones asignadas (identificador de trabajo, identificador de cola, acciones de sesión, configuración del registro), las solicitudes de cancelación, el estado del trabajador deseado y el intervalo de actualización.
- `BatchGetJobEntity`— Obtiene los detalles del trabajo asignado. Envía los identificadores de la entidad del trabajo. Recibe los detalles del trabajo, los detalles del entorno y los detalles adjuntos al trabajo.
- `AssumeFleetRoleForWorker`— Actualiza periódicamente las credenciales de los roles de la flota.
- `AssumeQueueRoleForWorker`— Obtiene las credenciales de los roles de cola correspondientes a una cola específica. El trabajador usa estas credenciales para acceder a los adjuntos de trabajo en Amazon S3.

Otros datos transmitidos

Además de las operaciones de la API de programación de Deadline Cloud, los anfitriones de los trabajadores transmiten los siguientes datos a otros AWS servicios:

Datos de registro

El agente de trabajo envía los registros del agente de trabajo y los registros de sesión (stdout y stderr de los procesos de trabajo) a los CloudWatch registros mediante la operación de la API. `PutLogEvents`

Adjuntos de trabajo

Los trabajadores transfieren los archivos de entrada y salida a través de Amazon S3 mediante `GetObject` operaciones de `PutObject` API. El trabajador utiliza las credenciales del rol de cola obtenidas mediante `AssumeQueueRoleForWorker` este acceso.

Telemetría (opcional)

El agente de trabajo envía métricas operativas, como informes de fallos. Puede optar por no participar en la recopilación de datos telemétricos. Para obtener más información, consulte cancelación de la suscripción.

Opciones de conectividad privada

Puede usarlo AWS PrivateLink para mantener el tráfico entre los hosts de los trabajadores de CMF y Deadline Cloud dentro de su VPC, sin tener que atravesar la Internet pública. En el caso de los trabajadores locales, puedes combinarla AWS PrivateLink con AWS Direct Connect (Direct Connect) o con una conexión VPN. Para obtener más información, consulte Acceso AWS Deadline Cloud utilizando un punto final de interfaz (AWS PrivateLink).

Pruebe la configuración de su host de trabajo

Una vez que haya instalado el agente trabajador, instalado el software necesario para procesar sus trabajos y configurar las AWS credenciales del agente trabajador, debe comprobar que la instalación puede procesar sus trabajos antes de crear un AMI para su flota. Debe probar lo siguiente:

- El agente de trabajo de Deadline Cloud está configurado correctamente para funcionar como un servicio del sistema.

- Que el trabajador sondee la cola de trabajo asociada.
- Que el trabajador procese correctamente los trabajos enviados a la cola asociada a la flota.

Una vez que haya probado la configuración y pueda procesar correctamente los trabajos representativos, puede utilizar el trabajador configurado para crear un AMI para EC2 los trabajadores de Amazon o como modelo para sus trabajadores locales.

> **Note**
>
> Si está probando la configuración del servidor de trabajo de una flota de autoescalado, es posible que tenga dificultades para probar a su trabajador en las siguientes situaciones:
>
> - Si no hay ningún trabajo en la cola, Deadline Cloud detiene al agente trabajador poco después de que el trabajador comience.
> - Si el agente trabajador está configurado para apagar el host cuando se detiene, el agente apaga la máquina si no hay trabajo en la cola.
>
> Para evitar estos problemas, utilice una flota provisional que no escale automáticamente para configurar y probar a sus trabajadores. Después de probar el anfitrión del trabajador, asegúrate de configurar el ID de flota correcto antes de preparar un AMI.

Para probar la configuración de su host de trabajo

1. Ejecute el agente de trabajo iniciando el servicio del sistema operativo.

   Linux

   Desde un shell raíz, ejecute el siguiente comando:

   ```
   systemctl start deadline-worker
   ```

   Windows

   Desde la línea de comandos de un administrador o PowerShell terminal, introduzca el siguiente comando:

   ```
   sc.exe start DeadlineWorker
   ```

2. Supervise al trabajador para asegurarse de que comienza y sondea si hay trabajo.

   Linux

   Desde un shell raíz, ejecute el siguiente comando:

   ```
   systemctl status deadline-worker
   ```

   El comando debería devolver una respuesta como:

   ```
   Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
   ```

   Si la respuesta no tiene ese aspecto, inspeccione el archivo de registro con el siguiente comando:

   ```
   tail -n 25 /var/log/amazon/deadline/worker-agent.log
   ```

   Windows

   Desde la línea de comandos del administrador o PowerShell terminal, introduzca el siguiente comando:

   ```
   sc.exe query DeadlineWorker
   ```

   El comando debería devolver una respuesta como:

   ```
   STATE   : 4 RUNNING
   ```

   Si la respuesta no lo contiene`RUNNING`, inspeccione el archivo de registro del trabajador. Abra y administre PowerShell solicite y ejecute el siguiente comando:

   ```
   Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
   agent.log
   ```

3. Envíe los trabajos a la cola asociada a su flota. Los trabajos deben ser representativos de los trabajos que procesa la flota.
4. Supervise el progreso del trabajo mediante el monitor de Deadline Cloud o la CLI. Si se produce un error en un trabajo, compruebe los registros de sesión y de trabajo.

5. Actualice la configuración del host de trabajo según sea necesario hasta que los trabajos se completen correctamente.
6. Cuando los trabajos de prueba sean satisfactorios, puede detener al trabajador:

   Linux

   Desde un shell raíz, ejecute el siguiente comando:

   ```
   systemctl stop deadline-worker
   ```

   Windows

   Desde la línea de comandos de un administrador o PowerShell terminal, introduzca el siguiente comando:

   ```
   sc.exe stop DeadlineWorker
   ```

Crea un Amazon Machine Image

Para crear un Amazon Machine Image (AMI) para usarlo en una flota gestionada por el cliente (CMF EC2) de Amazon Elastic Compute Cloud (Amazon), complete las tareas de esta sección. Debes crear una EC2 instancia de Amazon antes de continuar. Para obtener más información, consulta Lance your instance en la Guía del EC2 usuario de Amazon para instancias de Linux.

> **Important**
>
> Cómo crear una AMI crea una instantánea de los volúmenes adjuntos a la EC2 instancia de Amazon. Todo el software instalado en la instancia se conserva, por lo que las instancias se reutilizan cuando se lanzan instancias desde AMI. Recomendamos adoptar una estrategia de aplicación de parches y actualizar periódicamente cualquier nueva AMI con un software actualizado antes de aplicarlo a su flota.

Prepara la EC2 instancia de Amazon

Antes de crear un AMI, debe eliminar el estado del trabajador. El estado obrero persiste entre el lanzamiento del agente obrero. Si este estado persiste en AMI, entonces todas las instancias que se lancen desde él compartirán el mismo estado.

También le recomendamos que elimine todos los archivos de registro existentes. Los archivos de registro pueden permanecer en una EC2 instancia de Amazon cuando prepare la AMI. La eliminación de estos archivos minimiza la confusión a la hora de diagnosticar un posible problema en las flotas de trabajadores que utilizan la AMI.

También debes habilitar el servicio del sistema de agentes de trabajo para que el agente de trabajo de Deadline Cloud se lance cuando EC2 se inicie Amazon.

Por último, le recomendamos que active el apagado automático del agente de trabajo. Esto permite que la flota de trabajadores se amplíe cuando sea necesario y se cierre cuando finalice el trabajo de renderizado. Este escalado automático ayuda a garantizar que solo utilice los recursos según sea necesario.

Para preparar la EC2 instancia de Amazon

1. Abre la EC2 consola de Amazon.
2. Lanza una EC2 instancia de Amazon. Para obtener más información, consulte Lance your instance.
3. Configure el host para que se conecte a su proveedor de identidad (IdP) y, a continuación, monte cualquier sistema de archivos compartido que necesite.
4. Sigue los tutoriales paraInstale el agente de trabajo de Deadline Cloud, luegoConfigure el agente de trabajo, y. Cree usuarios y grupos de trabajo
5. Si está preparando un AMI basado en Amazon Linux 2023 para ejecutar un software compatible con la plataforma de referencia VFX, es necesario actualizar varios requisitos. Para obtener más información, consulte la compatibilidad de la plataforma de referencia de efectos visuales en la Guía del usuario de AWS Deadline Cloud.
6. Abra un terminal.
    a. En Linux, abre un terminal como `root` usuario (o usa`sudo`/`su`)
    b. Activado Windows, abra una línea de comandos o una PowerShell terminal de administrador.

7. Asegúrese de que el servicio de trabajo no se esté ejecutando y esté configurado para iniciarse al arrancar:

   a. En Linux, ejecute

   ```
   systemctl stop deadline-worker
   systemctl enable deadline-worker
   ```

   b. Activado Windows, ejecuta

   ```
   sc.exe stop DeadlineWorker
   sc.exe config DeadlineWorker start= auto
   ```

8. Elimine el estado del trabajador.

   a. En Linux, ejecute

   ```
   rm -rf /var/lib/deadline/*
   ```

   b. Activado Windows, ejecuta

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
   ```

9. Elimine los archivos de registro.

   a. En Linux, ejecute

   ```
   rm -rf /var/log/amazon/deadline/*
   ```

   b. Activado Windows, ejecuta

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
   ```

10. Activado Windows, se recomienda ejecutar la aplicación Amazon EC2 Launch Settings que se encuentra en el menú Inicio para completar la preparación final del host y el cierre de la instancia.

> **Note**
>
> DEBE elegir Apagar sin Sysprep y nunca apagar con Sysprep. Si se cierra con Sysprep, todos los usuarios locales quedarán inutilizables. Para obtener más información,

consulte la sección Antes de empezar del tema Crear una AMI personalizada de la Guía del usuario para instancias de Windows.

Cree el AMI

Para construir el AMI

1. Abre la EC2 consola de Amazon.
2. Selecciona Instances en el panel de navegación y, a continuación, selecciona tu instancia.
3. Seleccione Estado de la instancia y, a continuación, Detenga la instancia.
4. Una vez detenida la instancia, selecciona Acciones.
5. Selecciona Imagen y plantillas y, a continuación, Crear imagen.
6. Ingresa un nombre de imagen.
7. (Opcional) Introduce una descripción para la imagen.
8. Elija Crear imagen.

Cree una infraestructura de flota con un grupo de Amazon EC2 Auto Scaling

En esta sección se explica cómo crear una flota de Amazon EC2 Auto Scaling.

Utilice la plantilla CloudFormation YAML que aparece a continuación para crear un grupo de Amazon EC2 Auto Scaling (Auto Scaling), una Amazon Virtual Private Cloud (Amazon VPC) con dos subredes, un perfil de instancia y un rol de acceso a la instancia. Son necesarios para lanzar la instancia mediante Auto Scaling en las subredes.

Deberías revisar y actualizar la lista de tipos de instancias para adaptarla a tus necesidades de renderización.

Para obtener una explicación completa de los recursos y parámetros utilizados en la plantilla CloudFormation YAML, consulta la referencia sobre los tipos de recursos de Deadline Cloud en la Guía del AWS CloudFormation usuario.

Para crear una flota de Amazon EC2 Auto Scaling

1. Utilice el siguiente ejemplo para crear una CloudFormation plantilla que defina `FarmID` los `FleetID` `AMIId` parámetros y. Guarde la plantilla en un `.YAML` archivo de su equipo local.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
```

```
    InternetGatewayId: !Ref deadlineIGW
deadlinePublicRouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref deadlineVPC
deadlinePublicRoute:
  Type: 'AWS::EC2::Route'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref deadlineIGW
  DependsOn:
    - deadlineIGW
    - deadlineVPCGatewayAttachment
deadlinePublicSubnet0:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.16.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
        - a
deadlineSubnetRouteTableAssociation0:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    SubnetId: !Ref deadlinePublicSubnet0
deadlinePublicSubnet1:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.20.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
        - c
deadlineSubnetRouteTableAssociation1:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    SubnetId: !Ref deadlinePublicSubnet1
deadlineInstanceAccessAccessRole:
  Type: 'AWS::IAM::Role'
```

```
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
        - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
```

```
      MetadataOptions:
        HttpTokens: required
        HttpEndpoint: enabled

deadlineAutoScalingGroup:
  Type: 'AWS::AutoScaling::AutoScalingGroup'
  Properties:
    AutoScalingGroupName: !Join
      - ''
      - - deadline-ASG-autoscalable-
        - !Ref FleetId
    MinSize: 0
    MaxSize: 10
    VPCZoneIdentifier:
      - !Ref deadlinePublicSubnet0
      - !Ref deadlinePublicSubnet1
    NewInstancesProtectedFromScaleIn: true
    MixedInstancesPolicy:
      InstancesDistribution:
        OnDemandBaseCapacity: 0
        OnDemandPercentageAboveBaseCapacity: 0
        SpotAllocationStrategy: capacity-optimized
        OnDemandAllocationStrategy: lowest-price
      LaunchTemplate:
        LaunchTemplateSpecification:
          LaunchTemplateId: !Ref deadlineLaunchTemplate
          Version: !GetAtt
            - deadlineLaunchTemplate
            - LatestVersionNumber
        Overrides:
          - InstanceType: m5.large
          - InstanceType: m5d.large
          - InstanceType: m5a.large
          - InstanceType: m5ad.large
          - InstanceType: m5n.large
          - InstanceType: m5dn.large
          - InstanceType: m4.large
          - InstanceType: m3.large
          - InstanceType: r5.large
          - InstanceType: r5d.large
          - InstanceType: r5a.large
          - InstanceType: r5ad.large
          - InstanceType: r5n.large
          - InstanceType: r5dn.large
```

```
            - InstanceType: r4.large
      MetricsCollection:
        - Granularity: 1Minute
          Metrics:
            - GroupMinSize
            - GroupMaxSize
            - GroupDesiredCapacity
            - GroupInServiceInstances
            - GroupTotalInstances
            - GroupInServiceCapacity
            - GroupTotalCapacity
```

2. Abre la CloudFormation consola en https://console.aws.amazon.com/cloudformation.

   Usa la CloudFormation consola para crear una pila siguiendo las instrucciones para cargar el archivo de plantilla que has creado. Para obtener más información, consulte Crear una pila en la CloudFormation consola en la Guía del AWS CloudFormation usuario.

> **Note**
>
> - Las credenciales del rol de IAM asociadas a la instancia Amazon EC2 del trabajador están disponibles para todos los procesos que se ejecutan en ese trabajador, incluidos los trabajos. El trabajador debe tener el mínimo de privilegios para operar: y `deadline:CreateWorker` `deadline:AssumeFleetRoleForWorker.`
> - El agente de trabajo obtiene las credenciales para la función de cola y las configura para que las utilice en la ejecución de trabajos. El rol del perfil de instancia de Amazon EC2 no debe incluir los permisos que necesitan sus trabajos.

Amplíe automáticamente su flota de Amazon EC2 con la función de recomendación de escalado de Deadline Cloud

Deadline Cloud aprovecha un grupo de Amazon EC2 Auto Scaling (Auto Scaling) para escalar automáticamente la flota gestionada por el cliente (CMF) de Amazon EC2. Debe configurar el modo de flota e implementar la infraestructura requerida en su cuenta para que su flota se escale automáticamente. La infraestructura que implementaste funcionará en todas las flotas, por lo que solo tendrás que configurarla una vez.

El flujo de trabajo básico consiste en configurar el modo de flota para que se escale automáticamente y, a continuación, Deadline Cloud enviará un EventBridge evento para esa flota cada vez que cambie el tamaño de la flota recomendado (un evento contiene el identificador de la flota, el tamaño de la flota recomendado y otros metadatos). Dispondrá de una EventBridge regla para filtrar los eventos relevantes y dispondrá de una Lambda para consumirlos. La Lambda se integrará con Amazon EC2 Auto Scaling `AutoScalingGroup` para escalar automáticamente la flota de Amazon EC2.

Configure el modo de flota en **EVENT_BASED_AUTO_SCALING**

Configura tu modo de flota para`EVENT_BASED_AUTO_SCALING`. Para ello, puede utilizar la consola o utilizar la AWS CLI para llamar directamente a la `UpdateFleet` API `CreateFleet` o. Una vez configurado el modo, Deadline Cloud comienza a enviar EventBridge eventos cada vez que cambia el tamaño de flota recomendado.

- Ejemplo de `UpdateFleet` comando:

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- Ejemplo de `CreateFleet` comando:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

El siguiente es un ejemplo del `configuration.json` uso en los comandos CLI anteriores (`--configuration file://configuration.json`).

- Para activar Auto Scaling en su flota, debe configurar el modo en`EVENT_BASED_AUTO_SCALING`.
- Estos `workerCapabilities` son los valores predeterminados que se asignaron al CMF cuando lo creó. Puede cambiar estos valores si necesita aumentar los recursos disponibles para su CMF.

Después de configurar el modo de flota, Deadline Cloud comienza a emitir eventos de recomendación sobre el tamaño de la flota para esa flota.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

Implemente la pila de Auto Scaling mediante la CloudFormation plantilla

Puede configurar una EventBridge regla para filtrar los eventos, una Lambda para consumir los eventos y controlar Auto Scaling y una cola SQS para almacenar los eventos sin procesar. Utilice la siguiente CloudFormation plantilla para implementar todo en una pila. Una vez que hayas desplegado los recursos correctamente, puedes enviar un trabajo y la flota se ampliará automáticamente.

```
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
```

Realice un chequeo del estado de la flota

Después de crear tu flota, debes crear un chequeo de estado personalizado para asegurarte de que tu flota se mantenga en buen estado y libre de atascos, a fin de evitar costes innecesarios. Consulte Cómo implementar una revisión del estado de la flota en GitHub Deadline Cloud. Un chequeo de estado puede reducir el riesgo de que un cambio accidental en su configuraciónAmazon Machine Image, plantilla de lanzamiento o red pase desapercibido.

Configurar y usar flotas gestionadas por el servicio Deadline Cloud

Una flota gestionada por servicios (SMF) es un conjunto de trabajadores gestionados por Deadline Cloud. Una SMF elimina la necesidad de gestionar la ampliación de la flota para procesar las demandas o reduce el tamaño de la flota una vez finalizada la tarea.

Cuando se asocia un SMF a una cola mediante el entorno de colas predeterminado de Conda, Deadline Cloud configura a los trabajadores de la flota con el paquete de software adecuado. Para ver las aplicaciones de socios compatibles, consulte el entorno de colas conda predeterminado en la Guía del usuario de Deadline Cloud.AWS

En la mayoría de los casos, no es necesario cambiar un SMF para procesar las cargas de trabajo. Sin embargo, algunas situaciones pueden requerir que realice cambios en sus flotas.

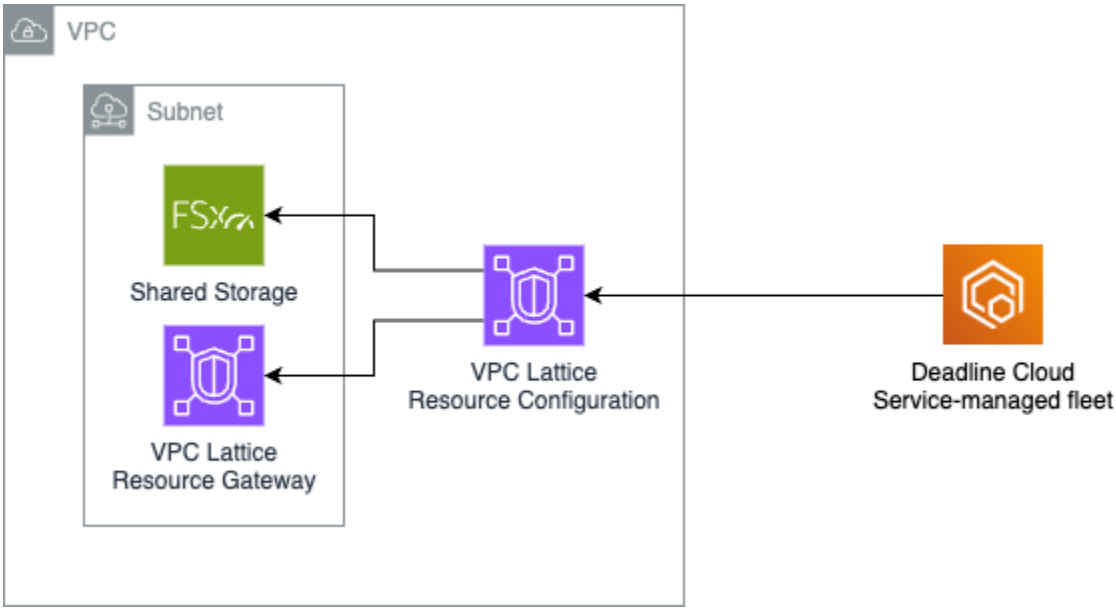
> **Note**
>
> Para instalar software personalizado en los trabajadores mediante scripts de configuración del host, consulteEjecute scripts de configuración del host con privilegios de administrador.

Temas

- Conecte los recursos de VPC a su SMF con puntos finales de recursos de VPC
- Utilice adjuntos de trabajo con flotas gestionadas por el servicio

Conecte los recursos de VPC a su SMF con puntos finales de recursos de VPC

Con los puntos de enlace de recursos de Amazon VPC para flotas gestionadas por servicios (SMF) de Deadline Cloud, puede conectar sus recursos de VPC, como sistemas de archivos de red (NFS), servidores de licencias y bases de datos, con sus trabajadores de Deadline Cloud. Esta función le permite aprovechar la plataforma totalmente gestionada de Deadline Cloud y, al mismo tiempo, integrarse con su infraestructura existente dentro de una VPC.



> **Tip**
>
> Para ver una CloudFormation plantilla de referencia que configura un FSx clúster de Amazon y lo conecta a una flota gestionada por un servicio, consulta smf_vpc_fsx en el repositorio de muestras de Deadline Cloud en. GitHub

Cómo funcionan los puntos finales de recursos de VPC

Los puntos finales de recursos de VPC utilizan VPC Lattice para crear una conexión segura entre los trabajadores de SMF de Deadline Cloud y los recursos de su VPC. La conexión es unidireccional, lo que significa que los trabajadores pueden establecer una conexión con los recursos de la VPC y transferir datos de un lado a otro, pero los recursos de la VPC no pueden establecer una conexión con un trabajador.

Cuando conectas un recurso de VPC a una flota gestionada por el servicio de Deadline Cloud, tus trabajadores pueden acceder a los recursos de tu VPC mediante un nombre de dominio privado. Además, el tráfico fluye de los trabajadores a sus recursos de VPC a través de VPC Lattice, y los recursos de su VPC ven el tráfico que proviene de la puerta de enlace de recursos de VPC Lattice.

Para obtener más información, consulte la guía del usuario de VPC Lattice.

Requisitos previos

Antes de conectar los recursos de VPC a su flota gestionada por el servicio de Deadline Cloud, asegúrese de disponer de lo siguiente:

- Una AWS cuenta con una VPC que contiene los recursos que desea conectar.
- Permisos de IAM para crear y administrar los recursos de VPC Lattice.
- Una granja de Deadline Cloud con al menos una flota gestionada por servicios.
- Recursos de VPC que desea hacer accesibles (NFSFSx, servidores de licencias, etc.).

Configurar un punto final de recursos de VPC

Para configurar un punto final de recursos de VPC, debe crear recursos en VPC Lattice y AWS RAM, a continuación, conectar esos recursos a su flota en Deadline Cloud. Para configurar un punto final de recursos de VPC para su SMF, complete los siguientes pasos.

1. Para crear una puerta de enlace de recursos en VPC Lattice, consulte Crear una puerta de enlace de recursos en la guía del usuario de VPC Lattice.
2. Para crear una configuración de recursos en VPC Lattice, consulte Crear una configuración de recursos en la guía del usuario de VPC Lattice.
3. Para compartir el recurso con su flota de Deadline Cloud, cree un recurso compartido en. AWS RAM Consulte Crear un recurso compartido para obtener instrucciones.

   Al crear un recurso compartido, en el caso de Directores, selecciona Principal de servicio en el menú desplegable y, a continuación, ingresa. **`fleets.deadline.amazonaws.com`**
4. Para conectar la configuración de recursos con su flota de Deadline Cloud, complete los siguientes pasos.

   a. Si aún no lo ha hecho, abra la consola de Deadline Cloud.

   b. En el panel de navegación, selecciona Granjas y, a continuación, selecciona tu granja.

   c. Seleccione la pestaña Flotas y, a continuación, seleccione su flota.

   d. Elija la pestaña Configurations (Configuraciones).

   e. En Puntos finales de recursos de VPC, elija Editar.

   f. Seleccione la configuración de recursos que creó y, a continuación, elija Guardar cambios.

Acceso a los recursos de su VPC

Tras conectar el recurso de VPC a la flota, los trabajadores pueden acceder a él mediante un nombre de dominio privado con el siguiente formato: `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Este dominio es privado y solo los trabajadores pueden acceder a él (no desde Internet ni desde su estación de trabajo).

Para montar o configurar el acceso al recurso de VPC en sus trabajadores, utilice un script de configuración del host. Los scripts de configuración del host se ejecutan con privilegios de administrador cuando los trabajadores comienzan a trabajar, lo que te permite montar sistemas de archivos, configurar los ajustes de red o realizar otras tareas de configuración.

Autenticación y seguridad

Para los recursos que requieren autenticación, almacene las credenciales de forma segura en AWS Secrets Manager, acceda a los secretos de los scripts de configuración o de trabajo del host e implemente los permisos de sistema de archivos adecuados para controlar el acceso. Tenga en cuenta las implicaciones de seguridad al compartir recursos entre varias flotas. Por ejemplo, si dos flotas están conectadas al mismo almacenamiento compartido, los trabajos que se ejecutan en una flota podrían acceder a los activos creados en la otra flota.

Consideraciones técnicas

Cuando utilice puntos de enlace de recursos de VPC, tenga en cuenta lo siguiente:

- Las conexiones solo se pueden iniciar desde los trabajadores a los recursos de la VPC, no desde los recursos de la VPC a los trabajadores.
- Una vez establecida, la conexión persiste hasta que se restablezca, incluso si la configuración de recursos está desconectada.
- La conexión VPC Lattice gestiona las conexiones entre las zonas de disponibilidad automáticamente sin cargos adicionales. La puerta de enlace de recursos debe compartir una zona de disponibilidad con el recurso de VPC, por lo que recomendamos configurar la puerta de enlace de recursos para que abarque todas las zonas de disponibilidad.
- El tráfico que pasa por el punto final de recursos de la VPC utiliza la traducción de direcciones de red (NAT), que no es compatible con todos los casos de uso. Por ejemplo, Microsoft Active Directory (AD) no puede conectarse a través de NAT.

Para obtener más información sobre las cuotas de VPC Lattice, consulte Cuotas para VPC Lattice.

Resolución de problemas

Si tiene problemas con los puntos finales de recursos de VPC, compruebe lo siguiente.

- Si recibes un mensaje de error como «mount.nfs: acceso denegado por el servidor durante el montaje», es posible que tengas que actualizar la configuración de cliente de tu volumen de NFS.
- Compruebe la configuración de sus recursos realizando pruebas desde una instancia de Amazon EC2 o AWS CloudShell en su VPC.
- Pruebe su conexión a Deadline Cloud con sencillos trabajos de CLI. Para obtener más información, consulte los ejemplos de Deadline Cloud en GitHub.
- Compruebe la configuración del grupo de seguridad de la puerta de enlace de recursos si se producen errores de conexión.
- Habilite los registros de acceso de la VPC para monitorear las conexiones.

Utilice adjuntos de trabajo con flotas gestionadas por el servicio

Los adjuntos de trabajo transfieren archivos entre tu estación de trabajo y los trabajadores de Deadline Cloud mediante Amazon Simple Storage Service (Amazon S3). Puede usar los archivos adjuntos de trabajo solos o junto con el almacenamiento compartido para adjuntar datos auxiliares a trabajos que no se compartan con otros trabajos, como scripts de trabajo, archivos de configuración o activos de proyectos almacenados localmente.

Para obtener información sobre cómo funcionan los adjuntos de trabajo, consulte Adjuntos de trabajos en la Guía del usuario de Deadline Cloud. Para obtener más información sobre cómo especificar los archivos de entrada y salida en los paquetes de trabajos, consulteUsa archivos adjuntos de trabajo para compartir archivos.

Elija un modo de sistema de archivos

Al enviar un trabajo con archivos adjuntos, puede elegir cómo los trabajadores cargan los archivos desde Amazon S3 configurando la `fileSystem` propiedad:

- COPIADO (predeterminado): descarga todos los archivos al disco local antes de que comiencen las tareas. Es ideal cuando cada tarea necesita la mayoría de los archivos de entrada.

- VIRTUAL: monta un sistema de archivos virtual que descarga archivos a pedido. Ideal cuando las tareas solo necesitan un subconjunto de archivos de entrada. Disponible solo para trabajadores SMF de Linux.

> **Important**
>
> El almacenamiento en caché en modo VIRTUAL puede aumentar el consumo de memoria y no está optimizado para todas las cargas de trabajo. Le recomendamos que pruebe su carga de trabajo antes de ejecutar los trabajos de producción.

Para obtener información detallada sobre la configuración del modo de sistema de archivos, consulte Sistema de archivos virtual en la Guía del usuario de Deadline Cloud.

Optimice el rendimiento de las transferencias

La velocidad de sincronización de los archivos de Amazon S3 con los trabajadores de SMF depende de la configuración del volumen de Amazon Elastic Block Store (Amazon EBS) de su flota. De forma predeterminada, los trabajadores de SMF utilizan los volúmenes gp3 de Amazon EBS con una configuración de rendimiento básica. Para cargas de trabajo con archivos de entrada grandes o muchos archivos pequeños, puede mejorar las velocidades de transferencia aumentando el rendimiento de Amazon EBS y la configuración de IOPS. Puede actualizar esta configuración mediante (). AWS Command Line Interface AWS CLI

Rendimiento (MiB/s)

: La velocidad a la que se pueden leer o escribir datos en el volumen. El valor predeterminado es 125 MiB/s, maximum is 1,000 MiB/s para los volúmenes gp3. Incremento para transferencias de archivos secuenciales de gran tamaño.

IOPS

: Operaciones de entrada/salida por segundo. El valor predeterminado es de 3000 IOPS y el máximo de 16 000 IOPS para los volúmenes gp3. Aumente al transferir muchos archivos pequeños.

> **Note**
>
> El aumento del rendimiento y las IOPS de Amazon EBS aumenta el coste de la flota. Para obtener información sobre precios, consulte los precios de Deadline Cloud.

Para actualizar la configuración de Amazon EBS en una flota existente mediante el AWS CLI

- Use el siguiente comando:

```
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"}
    }
  }'
```

Descargue los resultados de los trabajos

Cuando termine su trabajo, descargue los archivos de salida mediante la CLI de Deadline Cloud o el monitor de AWS Deadline Cloud (monitor de Deadline Cloud):

```
deadline job download-output --job-id job-0123456789abcdef0
```

Para descargar automáticamente los resultados a medida que se completan los trabajos, consulte Descargas automáticas en la Guía del usuario de Deadline Cloud.

Implemente y configure software personalizado en los trabajadores

AWS Deadline Cloud ofrece varios métodos para implementar y configurar software, complementos y herramientas personalizados para sus trabajadores. El método que elija depende de sus requisitos, como si necesita privilegios de administrador, con qué frecuencia cambia el software y si el software debe estar disponible para todos los trabajos o solo para trabajos específicos.

Elija un método de despliegue

Utilice la siguiente tabla para elegir el método de despliegue adecuado para su caso de uso.

Criterios	Entorno de colas	Script de configuración del host	Paquete conda personalizado
Se requieren privilegios de administrador	No	Sí	No
Cuando se ejecuta	Inicio de sesión	Inicio de un trabajador	Inicio de sesión
Alcance	Por cola o trabajo	Todos los trabajadores de la flota	Por cola o trabajo
Se puede controlar mediante el envío de un trabajo	Sí	No	Sí
Complejidad de la configuración	Bajo	Medio	Alto
Lo mejor para	Plugins, scripts y variables de entorno simples	Controladores de sistema, Docker, soportes de almacenamiento	Aplicaciones complejas con dependencias

Guía de decisión rápida:

- ¿Necesita privilegios de administrador o root? Utilice un script de configuración del host.
- ¿Un simple plugin o script sin derechos de administrador? Usa un entorno de colas.
- ¿Aplicación compleja con necesidades de control de versiones? Cree un paquete conda personalizado.

Configure los trabajos mediante entornos de colas

AWS Deadline Cloud utiliza entornos de colas para configurar el software de sus trabajadores. Un entorno le permite realizar tareas que requieren mucho tiempo, como la configuración y el desmontaje, de una sola vez para todas las tareas de una sesión. Define las acciones que debe ejecutar un trabajador al iniciar o detener una sesión. Puede configurar un entorno para una cola, los trabajos que se ejecutan en la cola y los pasos individuales de un trabajo.

Los entornos se definen como entornos de colas o entornos de trabajo. Cree entornos de colas con la consola de Deadline Cloud o con la fecha límite: CreateQueueEnvironment opere y defina los entornos de trabajo en las plantillas de trabajo de los trabajos que envíe. Siguen la especificación Open Job Description (OpenJD) para los entornos. Para obtener más información, consultehttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment <Environment>la especificación de OpenJD sobre. GitHub

Además de una `name` y`description`, cada entorno contiene dos campos que definen el entorno del host. Son los siguientes:

- `script`— La acción que se lleva a cabo cuando este entorno se ejecuta en un trabajador.
- `variables`— Un conjunto de name/value pares de variables de entorno que se establecen al entrar en el entorno.

Debe establecer al menos uno de los siguientes valores: `script` o`variables`.

Puede definir más de un entorno en su plantilla de trabajo. Cada entorno se aplica en el orden en que aparecen en la plantilla. Puede usarlo para ayudar a gestionar la complejidad de sus entornos.

El entorno de colas predeterminado de Deadline Cloud usa el administrador de paquetes conda para cargar el software en el entorno, pero puedes usar otros administradores de paquetes. El entorno predeterminado define dos parámetros para especificar el software que se debe cargar.

Estas variables las configuran los remitentes proporcionados por Deadline Cloud, aunque puedes configurarlas en tus propios scripts y aplicaciones que utilizan el entorno predeterminado. Son los siguientes:

- `CondaPackages`— Una lista separada por espacios de los paquetes conda que coinciden con las especificaciones que se deben instalar para el trabajo. Por ejemplo, el remitente de Blender añadiría fotogramas `blender=3.6` para renderizar en Blender 3.6.
- `CondaChannels`— Una lista de canales conda separados por espacios desde los que instalar paquetes. En el caso de las flotas gestionadas por servicios, los paquetes se instalan desde el canal. `deadline-cloud` Puede añadir otros canales.

Controle el entorno de trabajo con los entornos de colas de OpenJD

Puede definir entornos personalizados para sus trabajos de renderizado mediante entornos de cola. Un entorno de colas es una plantilla que controla las variables de entorno, las asignaciones de archivos y otros ajustes de los trabajos que se ejecutan en una cola específica. Le permite adaptar el entorno de ejecución de los trabajos enviados a una cola en función de los requisitos de sus cargas de trabajo. AWS Deadline Cloud ofrece tres niveles anidados en los que puede aplicar entornos de Open Job Description (OpenJD): cola, trabajo y paso. Al definir los entornos de colas, puedes garantizar un rendimiento uniforme y optimizado para los distintos tipos de trabajos, agilizar la asignación de recursos y simplificar la gestión de las colas.

El entorno de colas es una plantilla que se adjunta a una cola de la AWS cuenta desde la consola de AWS administración o mediante la. AWS CLI Puede crear un entorno para una cola o puede crear varios entornos de cola que se apliquen para crear el entorno de ejecución. Este enfoque le permite crear y probar un entorno por pasos para garantizar que funcione correctamente en sus trabajos.

Los entornos de trabajos y pasos se definen en la plantilla de trabajo que se utiliza para crear un trabajo en la cola. La sintaxis de OpenJD es la misma en estos distintos tipos de entornos. En esta sección, las mostraremos dentro de las plantillas de trabajo.

Temas

- Establezca las variables de entorno en un entorno de colas
- Establezca la ruta en un entorno de cola
- Ejecute un proceso daemon en segundo plano desde el entorno de colas

Establezca las variables de entorno en un entorno de colas

Muchas aplicaciones y marcos utilizan variables de entorno para controlar la configuración de las funciones, los niveles de registro y la configuración de la pantalla. Puede usar los entornos Open Job Description (OpenJD) para establecer variables de entorno que herede cada comando de tarea dentro de su ámbito.

Ámbito de la variable de entorno

AWS Deadline Cloud aplica variables de entorno de los entornos de colas que se adjuntan a una cola. Dentro de una plantilla de trabajo, también puede definir variables de entorno a nivel de trabajo y paso mediante entornos de OpenJD. Las variables definidas en un ámbito más limitado anulan las variables con el mismo nombre de un ámbito más amplio.

- Entorno de colas: plantilla que se adjunta a una cola en Deadline Cloud. Las variables se aplican a todos los trabajos enviados a la cola. Puede configurar las variables con un `variables` mapa para valores fijos o utilizar scripts para valores dinámicos.
- Entorno laboral: se define `jobEnvironments` en una plantilla de trabajo. Las variables se aplican a todos los pasos y tareas del trabajo. Una variable de nivel de trabajo anula una variable de nivel de cola con el mismo nombre.
- Entorno escalonado: se define en una plantilla de trabajo. `stepEnvironments` Las variables se aplican solo a las tareas de ese paso. Una variable de nivel de paso anula una variable de nivel de trabajo o de cola con el mismo nombre.

Establecer variables en un entorno de colas

Puede configurar las variables de entorno en un entorno de colas mediante un `variables` mapa para valores fijos o mediante una `script` `onEnter` acción para valores dinámicos.

La siguiente plantilla de entorno de colas utiliza un `variables` mapa para establecer la `QT_QPA_PLATFORM` variable`offscreen`, lo que permite que las aplicaciones que utilizan Qt Framework se ejecuten en hosts de trabajo sin una pantalla interactiva.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Para valores dinámicos, como modificar `PATH` o activar entornos virtuales, utilice un script que imprima las líneas en formato `openjd_env: VAR=value` stdout. El `openjd_env:` prefijo es obligatorio. El uso de `echoexport`, u otros mecanismos de shell sin el prefijo no propaga las variables a los trabajos y tareas.

La siguiente plantilla de entorno de colas establece la `QT_QPA_PLATFORM` variable mediante un script.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

Para adjuntar un entorno de colas a su cola, utilice la consola de Deadline Cloud o la. AWS CLI Para obtener más información, consulte Crear un entorno de colas en la Guía del usuario de AWS Deadline Cloud. El siguiente AWS CLI comando crea un entorno de colas a partir de un archivo de plantilla.

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

Para ver ejemplos más complejos, como la creación y activación de entornos virtuales de conda, consulte los ejemplos del entorno de colas de Deadline Cloud en. GitHub

Establecer variables en una plantilla de trabajo

En una plantilla de trabajo, añada un `variables` mapa a una `stepEnvironments` entrada `jobEnvironments` o. Cada entrada es un par clave-valor en el que la clave es el nombre de la variable y el valor es el valor de la variable.

La siguiente plantilla de trabajo establece la variable de `QT_QPA_PLATFORM` entorno en`offscreen`, lo que permite que las aplicaciones que utilizan Qt Framework se ejecuten en hosts de trabajo sin una pantalla interactiva.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

Puede establecer múltiples variables en una sola definición de entorno.

```
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
```

Puede hacer referencia a los parámetros del trabajo en valores variables mediante la `{{Param.ParameterName}}` sintaxis.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Para anular una variable de nivel de trabajo para un paso específico, defina una `stepEnvironments` entrada con el mismo nombre de variable. En el siguiente ejemplo, se define `JOB_PROJECT_ID` en el nivel de trabajo con el valor y`project-12`, a continuación, se reemplaza el valor en el nivel de paso con. `step-project-12` Las tareas del paso utilizan el valor del nivel de paso.

```
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
```

Pruébelo: ejecute el ejemplo de la variable de entorno

El repositorio de muestras de Deadline Cloud incluye un paquete de trabajos que muestra la configuración y visualización de las variables de entorno. La plantilla de trabajo de muestra define las variables tanto a nivel de trabajo como de paso y, a continuación, ejecuta una tarea que imprime el resultado combinado. Utilice el siguiente procedimiento para ejecutar la muestra e inspeccionar los resultados.

Requisitos previos

1. Si no tiene una granja de Deadline Cloud con una cola y una flota Linux asociada, siga la experiencia de incorporación guiada en la consola de Deadline Cloud para crear una con la configuración predeterminada.
2. Si no tiene la CLI de Deadline Cloud ni el monitor de AWS Deadline Cloud en su estación de trabajo, siga los pasos de Configurar los remitentes de Deadline Cloud.
3. Úselo `git` para clonar el repositorio de muestras GitHub de Deadline Cloud.

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

Ejecución del ejemplo

1. Utilice la CLI de Deadline Cloud para enviar la `job_env_vars` muestra.

```
deadline bundle submit job_env_vars
```

2. En el monitor de Deadline Cloud, selecciona el nuevo trabajo para supervisar su progreso. Cuando la Linux flota asociada a la cola tenga un trabajador disponible, el trabajo se completará en unos segundos. Seleccione la tarea y, a continuación, elija Ver registros en el menú superior derecho del panel de tareas.

Comparar las acciones de la sesión con sus definiciones

La vista de registro muestra tres acciones de la sesión. Abra el archivo job_env_vars/template.yaml en un editor de texto para comparar cada acción con su definición en la plantilla de trabajo.

1. JobEnvSeleccione la acción Iniciar sesión. El resultado del registro muestra las variables de entorno a nivel de trabajo que se están configurando.

```
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
```

   Las siguientes líneas de la plantilla de trabajo definen este entorno.

```
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. Seleccione la acción Iniciar StepEnv sesión. El resultado del registro muestra las variables a nivel de paso, incluidas las anuladasJOB_PROJECT_ID.

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   Las siguientes líneas de la plantilla de trabajo definen este entorno.

```
stepEnvironments:
- name: StepEnv
```

```
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Seleccione la acción Tarea ejecutar sesión. El resultado del registro muestra las variables de entorno combinadas disponibles para la tarea. Observe que `JOB_PROJECT_ID` utiliza el valor `step-project-12` de nivel escalonado.

```
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
```

Establezca la ruta en un entorno de cola

Utilice los entornos OpenJD para proporcionar nuevos comandos en un entorno. Primero, cree un directorio que contenga los archivos de script y, a continuación, añada ese directorio a las variables de `PATH` entorno para que los ejecutables del script puedan ejecutarlos sin necesidad de especificar la ruta del directorio cada vez. La lista de variables de una definición de entorno no proporciona una forma de modificar la variable, por lo que, en su lugar, se ejecuta un script. Una vez que el script configura las cosas y las modificaPATH, exporta la variable al motor de ejecución de OpenJD con el comando. `echo "openjd_env: PATH=$PATH"`

Requisitos previos

Realice los siguientes pasos para ejecutar el paquete de trabajos de muestra con variables de entorno del repositorio de muestras de Deadline Cloud en GitHub.

1. Si no tienes una granja de Deadline Cloud con una cola y una flota Linux asociada, sigue la experiencia de incorporación guiada en la consola de Deadline Cloud para crear una con la configuración predeterminada.
2. Si no tiene la CLI de Deadline Cloud ni el monitor de Deadline Cloud en su estación de trabajo, siga los pasos de Configurar los remitentes de Deadline Cloud de la guía del usuario.
3. Úselo `git` para clonar el repositorio de muestras GitHub de Deadline Cloud.

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

```
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Ejecute el ejemplo de ruta

1. Utilice la CLI de Deadline Cloud para enviar la `job_env_with_new_command` muestra.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. En el monitor de Deadline Cloud, verá el nuevo trabajo y podrá supervisar su progreso. Una vez que la Linux flota asociada a la cola tenga un trabajador disponible para ejecutar la tarea, ésta se completará en unos segundos. Seleccione la tarea y, a continuación, elija la opción Ver registros en el menú superior derecho del panel de tareas.

   A la derecha hay dos acciones de sesión: Iniciar RandomSleepCommand y ejecutar una tarea. El visor de registros en el centro de la ventana corresponde a la acción de sesión seleccionada a la derecha.

Compare las acciones de la sesión con sus definiciones

En esta sección, utiliza el monitor de Deadline Cloud para comparar las acciones de la sesión con el lugar en el que están definidas en la plantilla de trabajo. Es la continuación de la sección anterior.

Abre el archivo job_env_with_new_command/template.yaml en un editor de texto. Compare las acciones de la sesión con el lugar donde están definidas en la plantilla de trabajo.

1. Seleccione la acción Iniciar RandomSleepCommand sesión en el monitor de Deadline Cloud. Verá el resultado del registro de la siguiente manera.

```
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.
```

Las siguientes líneas de la plantilla de trabajo especifican esta acción.

```
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. Seleccione la acción Iniciar StepEnv sesión en el monitor de Deadline Cloud. El resultado del registro se muestra de la siguiente manera.

```
 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. Las siguientes líneas de la plantilla de trabajo especificaban esta acción.

```
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
```

Ejecute un proceso daemon en segundo plano desde el entorno de colas

En muchos casos de uso del renderizado, cargar los datos de la aplicación y de la escena puede llevar un tiempo considerable. Si un trabajo los vuelve a cargar para cada fotograma, pasará la mayor parte del tiempo sobrecargado. A menudo es posible cargar la aplicación una vez como un proceso daemon en segundo plano, hacer que cargue los datos de la escena y, a continuación, enviarle comandos mediante la comunicación entre procesos (IPC) para realizar los renderizados.

Muchas de las integraciones de código abierto de Deadline Cloud utilizan este patrón. El proyecto Open Job Description proporciona una biblioteca de tiempo de ejecución de adaptadores con patrones de IPC sólidos en todos los sistemas operativos compatibles.

Para demostrar este patrón, hay un paquete de trabajos de muestra autónomo que utiliza Python y código bash para implementar un daemon en segundo plano y el IPC para que las tareas se comuniquen con él. El daemon está implementado en Python y escucha una SIGUSR1 señal POSIX para saber cuándo procesar una tarea. Los detalles de la tarea se pasan al daemon en un archivo JSON específico y los resultados de la ejecución de la tarea se devuelven como otro archivo JSON.

Requisitos previos

Realice los siguientes pasos para ejecutar el paquete de trabajos de muestra con un proceso daemon del repositorio de muestras de Deadline Cloud en GitHub.

1. Si no tienes una granja de Deadline Cloud con una cola y una flota Linux asociada, sigue la experiencia de incorporación guiada en la consola de Deadline Cloud para crear una con la configuración predeterminada.
2. Si no tiene la CLI de Deadline Cloud ni el monitor de Deadline Cloud en su estación de trabajo, siga los pasos de Configurar los remitentes de Deadline Cloud de la guía del usuario.
3. Úselo `git` para clonar el repositorio de muestras GitHub de Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Ejecute el ejemplo del daemon

1. Utilice la CLI de Deadline Cloud para enviar la `job_env_daemon_process` muestra.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. En la aplicación de monitoreo de Deadline Cloud, verá el nuevo trabajo y podrá monitorear su progreso. Una vez que la Linux flota asociada a la cola tenga un trabajador disponible para ejecutar la tarea, la tarea se completará en aproximadamente un minuto. Con una de las tareas seleccionada, selecciona la opción Ver registros en el menú superior derecho del panel de tareas.

   A la derecha hay dos acciones de sesión: Iniciar DaemonProcess y Ejecutar tareas. El visor de registros en el centro de la ventana corresponde a la acción de sesión seleccionada a la derecha.

Seleccione la opción Ver los registros de todas las tareas. La cronología muestra el resto de las tareas que se ejecutaron como parte de la sesión y la `Shut down DaemonProcess` acción que salió del entorno.

Vea los registros del daemon

1. En esta sección, utiliza el monitor de Deadline Cloud para comparar las acciones de la sesión con el lugar en el que están definidas en la plantilla de trabajo. Es la continuación de la sección anterior.

   Abre el archivo job_env_daemon_process/template.yaml en un editor de texto. Compare las acciones de la sesión con el lugar donde están definidas en la plantilla de trabajo.

2. Seleccione la acción de la `Launch DaemonProcess` sesión en el monitor de Deadline Cloud. Verá el resultado del registro de la siguiente manera.

```
 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
```

Las siguientes líneas de la plantilla de trabajo especifican esta acción.

```
  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. Seleccione una de las acciones de sesión Ejecutar: N de la tarea en el monitor de Deadline Cloud. Verá el resultado del registro de la siguiente manera.

```
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 ----------------------------------------------
```

Las siguientes líneas de la plantilla de trabajo son las que especifican esta acción. ```pasos:

```
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
```

Proporcione solicitudes para sus puestos de trabajo

Puede utilizar un entorno de colas para cargar aplicaciones y procesar sus trabajos. Al crear una flota gestionada por un servicio mediante la consola de Deadline Cloud, tiene la opción de crear un entorno de colas que utilice el administrador de paquetes conda para cargar las aplicaciones.

Si desea utilizar un administrador de paquetes diferente, puede crear un entorno de colas para ese administrador. Para ver un ejemplo del uso de Rez, consulteUsa un administrador de paquetes diferente.

Deadline Cloud proporciona un canal conda para cargar una selección de aplicaciones de renderizado en su entorno. Apoyan a los remitentes que Deadline Cloud proporciona para las solicitudes de creación de contenido digital.

También puede cargar software para que conda-forge lo utilice en sus trabajos. Los siguientes ejemplos muestran plantillas de trabajos que utilizan el entorno de colas proporcionado por Deadline Cloud para cargar las aplicaciones antes de ejecutar el trabajo.

Temas

Obtener una solicitud de un canal conda

Puedes crear un entorno de colas personalizado para tus trabajadores de Deadline Cloud e instalar el software que prefieras. Este ejemplo de entorno de colas tiene el mismo comportamiento que el entorno utilizado por la consola para las flotas gestionadas por el servicio. Ejecuta conda directamente para crear el entorno.

El entorno crea un nuevo entorno virtual de conda para cada sesión de Deadline Cloud que se ejecute en un trabajador y, a continuación, elimina el entorno cuando finaliza.

Conda almacena en caché los paquetes descargados para que no sea necesario volver a descargarlos, pero cada sesión debe vincular todos los paquetes al entorno.

El entorno define tres scripts que se ejecutan cuando Deadline Cloud inicia una sesión con un trabajador. El primer script se ejecuta cuando se `onEnter` invoca la acción. Llama a los otros dos para configurar las variables de entorno. Cuando el script termina de ejecutarse, el entorno conda está disponible con todas las variables de entorno especificadas configuradas.

Para ver la última versión del ejemplo, consulta conda_queue_env_console_equivalent.yaml en el repositorio de. deadline-cloud-samples GitHub

Si desea utilizar una aplicación que no está disponible en el canal conda, puede crear un canal conda en Amazon S3 y, a continuación, crear sus propios paquetes para esa aplicación. Consulte Cree un canal conda con S3 para obtener más información.

Obtenga bibliotecas de código abierto de conda-forge

En esta sección se describe cómo utilizar las bibliotecas de código abierto del `conda-forge` canal. El siguiente ejemplo es una plantilla de trabajo que usa el paquete `polars` Python.

El trabajo establece los `CondaChannels` parámetros `CondaPackages` y parámetros definidos en el entorno de colas que indican a Deadline Cloud dónde obtener el paquete.

La sección de la plantilla de trabajo que establece los parámetros es:

```
- 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
```

Para ver la versión más reciente de la plantilla de trabajo de ejemplo completa, consulte stage_1_self_contained_template/template.yaml. Para ver la versión más reciente del entorno de colas que carga los paquetes conda, consulta conda_queue_env_console_equivalent.yaml en el repositorio de. deadline-cloud-samples GitHub

Sal Blender del canal deadline-cloud

El siguiente ejemplo muestra una plantilla de trabajo que Blender proviene del canal `deadline-cloud` conda. Este canal admite los remitentes que Deadline Cloud proporciona para el software de creación de contenido digital, aunque puedes usar el mismo canal para cargar software para tu propio uso.

Para ver una lista del software ofrecido por el `deadline-cloud` canal, consulta el entorno de colas predeterminado en la Guía del usuario de AWS Deadline Cloud.

Este trabajo establece el `CondaPackages` parámetro definido en el entorno de colas para indicar a Deadline Cloud que cargue Blender en el entorno.

La sección de la plantilla de trabajo que establece el parámetro es:

```
- 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.
```

Para ver la versión más reciente de la plantilla de trabajo de ejemplo completa, consulta blender_render/template.yaml. Para ver la versión más reciente del entorno de colas que carga los paquetes conda, consulta conda_queue_env_console_equivalent.yaml en el repositorio de. deadline-cloud-samplesGitHub

Usa un administrador de paquetes diferente

El administrador de paquetes predeterminado para Deadline Cloud es conda. Si necesitas usar un administrador de paquetes diferente, por ejemploRez, puedes crear un entorno de colas personalizado que contenga scripts que usen tu administrador de paquetes en su lugar.

Este ejemplo de entorno de colas proporciona el mismo comportamiento que el entorno utilizado por la consola para las flotas gestionadas por el servicio. Sustituye al administrador de paquetes conda por. Rez

El entorno define tres scripts que se ejecutan cuando Deadline Cloud inicia una sesión con un trabajador. El primer script se ejecuta cuando se `onEnter` invoca la acción. Llama a los otros dos para configurar las variables de entorno. Cuando el script termina de ejecutarse, el Rez entorno está disponible con todas las variables de entorno especificadas configuradas.

En el ejemplo se supone que tiene una flota gestionada por el cliente que utiliza un sistema de archivos compartido para los paquetes Rez.

Para ver la versión más reciente del ejemplo, consulta rez_queue_env.yaml en el repositorio de. deadline-cloud-samplesGitHub

Cree un canal conda con S3

Si sus trabajos necesitan ejecutar aplicaciones que no están disponibles en los `conda-forge`canales `deadline-cloud`o, puede alojar un canal conda personalizado para que sirva sus

propios paquetes. Al crear una cola en la consola de AWS Deadline Cloud (Deadline Cloud), la consola añade un entorno de colas condas de forma predeterminada. Para que sus paquetes estén disponibles para los trabajos, añada el canal personalizado al entorno de colas.

Un canal conda es contenido alojado estático que se puede alojar de diversas formas, por ejemplo, en un sistema de archivos o en un bucket de Amazon Simple Storage Service (Amazon S3). Si tu granja de Deadline Cloud usa un sistema de archivos compartido para los activos, puedes usar cualquier ruta del mismo como nombre de canal. Puede alojar el canal en un bucket de Amazon S3 para un acceso más amplio mediante permisos AWS Identity and Access Management (IAM).

Puede crear y probar paquetes de forma local y, a continuación, publicarlos en un canal. Crear paquetes de forma local es una forma sencilla de empezar a iterar recetas de creación de paquetes sin necesidad de configurar la infraestructura. También puedes usar una cola de creación de paquetes de Deadline Cloud para crear paquetes y publicarlos en un canal. Una cola de creación de paquetes simplifica el mantenimiento de los paquetes para varios sistemas operativos y configuraciones de aceleradores. Puede actualizar las versiones y enviar conjuntos completos de compilaciones de paquetes desde cualquier lugar.

Puedes configurar los canales para tu estudio y tu granja de Deadline Cloud de varias maneras. Puede tener un canal Amazon S3 y configurar todas las estaciones de trabajo y los hosts de la granja para que lo utilicen. También puede tener más de un canal y configurar la duplicación con AWS DataSync (). DataSync Por ejemplo, la cola de creación de paquetes de Deadline Cloud puede publicarse en un canal de Amazon S3 que se refleje localmente para estaciones de trabajo y hosts de granjas locales.

Temas

- Cree y pruebe paquetes localmente
- Publica paquetes en un canal conda de Amazon S3
- Configure los permisos de la cola de producción para paquetes conda personalizados
- Añada un canal conda a un entorno de colas
- Cree un paquete conda para una aplicación o un complemento
- Cree una receta de construcción de condas para Blender
- Cree una receta de construcción de conda para Autodesk Maya
- Cree una receta de construcción de conda para el adaptador Maya
- Crea una receta de compilación de conda para el plugin Autodesk Maya to Arnold (MtoA)
- Automatice la creación de paquetes con Deadline Cloud

Cree y pruebe paquetes localmente

Antes de publicar paquetes en Amazon S3 o configurar la CI/CD automatización en su granja de Deadline Cloud, puede crear y probar paquetes conda en su estación de trabajo mediante un canal de sistema de archivos local. Este enfoque le permite iterar rápidamente las recetas a nivel local y verificar los paquetes.

El `rattler-build publish` comando crea una receta, copia el paquete resultante en un canal e indexa el canal en un solo paso. Cuando se dirige a un directorio del sistema de archivos local, `rattler-build` crea e inicializa el canal automáticamente si el directorio no existe.

Las siguientes instrucciones utilizan la receta de ejemplo Blender 4.5 del repositorio de muestras de Deadline Cloud en adelante. GitHub Puedes sustituirla por una receta diferente del repositorio de muestras o usar la tuya propia.

Requisitos previos

Antes de empezar, instale las siguientes herramientas en su estación de trabajo:

- pixi: un administrador de paquetes que se utiliza para instalar `rattler-build` y probar paquetes. Instala pixi desde pixi.sh.
- rattler-build: la herramienta de creación de paquetes utilizada por las recetas conda de Deadline Cloud. Después de instalar pixi, ejecuta el siguiente comando para instalarlo. `rattler-build`

  ```
  pixi global install rattler-build
  ```
- git: necesario para clonar el repositorio de muestras. SíWindows, git for Windows también proporciona un `bash` intérprete de comandos, que requieren algunas de las recetas de Windows muestra.

Crear y publicar un paquete en un canal local

En este procedimiento, se clona el repositorio de muestras de Deadline Cloud y se utiliza `rattler-build publish` para crear y publicar el paquete en un canal del sistema de archivos local.

> **Note**
>
> Las aplicaciones de gran tamaño pueden requerir decenas de GB de espacio libre en disco para el archivo de origen, los archivos extraídos y la generación de resultados. Asegúrese

de utilizar un disco con suficiente espacio disponible para el resultado de la compilación del paquete.

Para crear y publicar un paquete en un canal local

1. Clona el repositorio de muestras de Deadline Cloud.

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

2. Cambie al directorio de `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Ejecute el siguiente comando para crear la receta Blender 4.5 y publicar el paquete en un directorio de canales local.

   LinuxActiva macOS y ejecuta el siguiente comando.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

   En Windows (cmd), ejecute el siguiente comando.

```
rattler-build publish blender-4.5/recipe/recipe.yaml ^
    --to file://%USERPROFILE%/my-conda-channel ^
    --build-number=+1
```

   El `rattler-build publish` comando realiza las siguientes acciones:

   - Construye el paquete a partir de la receta.
   - Crea el directorio de canales si el directorio no existe.
   - Copia el archivo del paquete en el canal.
   - Indexa el canal para que los administradores de paquetes puedan encontrarlo.

   Si la receta de su paquete depende de los paquetes de un canal en particular, como conda-forge, añada `-c conda-forge` al comando.

> **Acerca de los números de compilación**
>
> La `--build-number=+1` opción selecciona automáticamente el siguiente número de compilación en función de lo que ya existe en el canal de destino. La mejor práctica es no sobrescribir nunca un paquete en un canal. Cree siempre con un número de compilación nuevo si, de lo contrario, el paquete tendría el mismo nombre de archivo. El uso lo `--build-number=+1` consigue cuando se crea un canal de producción o un canal provisional que refleja la producción.
>
> Si desea controlar el número de compilación directamente, puede configurarlo con un valor específico, como`--build-number=7`. Si omite la opción, `rattler-build` utiliza el número de compilación definido en el `recipe.yaml` archivo.

Para obtener más información al respecto`rattler-build publish`, consulte la documentación de publicación de rattler-build.

Depuración de compilaciones

Si una compilación falla, `rattler-build` conserva el directorio de compilación para que puedas investigarlo. Ejecuta el siguiente comando para abrir un shell interactivo en el entorno de compilación con todas las variables de entorno configuradas tal y como estaban durante la compilación.

```
rattler-build debug shell
```

Desde el shell de depuración, puedes modificar archivos, ejecutar comandos de compilación individuales y añadir dependencias para aislar el problema. Para obtener más información, consulta la sección Depuración de compilaciones en la documentación de rattler-build.

Probando el paquete

Después de crear y publicar el paquete, crea un proyecto pixi temporal. Utilice el proyecto para instalar el paquete desde el canal local y comprobar que funciona correctamente.

Para probar el paquete

1. Cree un directorio de prueba temporal e inicialice un proyecto pixi con el canal local.

   Active Linux y ejecute macOS los siguientes comandos.

   ```
   mkdir package-test-env
   ```

```
cd package-test-env
pixi init --channel file://$HOME/my-conda-channel
```

En Windows (cmd), ejecute los siguientes comandos.

```
mkdir package-test-env
cd package-test-env
pixi init --channel file://%USERPROFILE%/my-conda-channel
```

2. Añada el paquete al proyecto.

```
pixi add blender=4.5
```

3. Compruebe que el paquete funciona correctamente.

```
pixi run blender --version
```

El `pixi run`comando activa el entorno conda para el directorio del proyecto y ejecuta el comando especificado en él. El entorno permanece en el directorio del proyecto, por lo que puede utilizar el mismo `pixi run` comando desde otros terminales.

Cuando esté satisfecho con el paquete, puede publicarlo en un canal conda de Amazon S3 para que los trabajadores de Deadline Cloud puedan instalarlo. Consulte Publicar paquetes en un canal conda de S3.

Eliminar paquetes del canal

Evite eliminar paquetes de los canales que utiliza para la producción, ya que los archivos de bloqueo hacen referencia a paquetes específicos mediante un hash. Al eliminar un paquete, se impide volver a crear entornos a partir de esos archivos de bloqueo. En el caso de los canales de desarrollo y prueba, puede eliminar un paquete específico eliminando el `.conda` archivo del directorio del canal y, a continuación, volviendo a indexar el canal. Primero, instálalo. `rattler-index`

```
pixi global install rattler-index
```

A continuación, elimine el archivo del paquete y vuelva a indexar el canal.

Active Linux y macOS ejecute los siguientes comandos.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

En Windows (cmd), ejecute los siguientes comandos.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

Package files se almacenan en subdirectorios específicos de la plataforma, como`linux-64`, o. `win-64` `osx-arm64` Enumere el contenido de estos subdirectorios para encontrar el nombre exacto del paquete que desea eliminar.

Limpieza

Tras la prueba, puede eliminar el proyecto de prueba y el canal local.

Para limpiar los recursos de prueba

1. Elimine el directorio del proyecto de prueba.

   Active Linux y macOS ejecute el siguiente comando.

   ```
   rm -rf package-test-env
   ```

   En Windows (cmd), ejecute el siguiente comando.

   ```
   rmdir /s /q package-test-env
   ```

2. Elimine el directorio de canales conda local.

   Active Linux y macOS ejecute el siguiente comando.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   En Windows (cmd), ejecute el siguiente comando.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Opcional) Elimine el directorio `rattler-build` de salida que contiene el archivo del paquete creado.

Active Linux y macOS ejecute el siguiente comando.

```
rm -rf deadline-cloud-samples/conda_recipes/output
```

En Windows (cmd), ejecute el siguiente comando.

```
rmdir /s /q deadline-cloud-samples\conda_recipes\output
```

Publica paquetes en un canal conda de Amazon S3

Puede publicar paquetes conda en un depósito de Amazon Simple Storage Service (Amazon S3) para AWS que los trabajadores de Deadline Cloud (Deadline Cloud) puedan instalarlos para ejecutar tareas. El `rattler-build publish` comando funciona con Amazon S3 de la misma manera que con un canal de sistema de archivos local. El comando puede crear una receta y publicar el resultado, o publicar un archivo de paquete que ya haya creado. En ambos casos, el comando carga el paquete en el depósito e indexa el canal en un solo paso.

El `rattler-build publish` comando se autentica AWS mediante la cadena de credenciales estándar, por lo que utiliza la AWS configuración como cualquier otra herramienta. AWS Para obtener más información sobre la configuración de las credenciales, consulte los ajustes de configuración y del archivo de credenciales en la Guía del usuario de AWS Command Line Interface (AWS CLI).

Requisitos previos

Antes de publicar paquetes en Amazon S3, complete los siguientes requisitos previos:

- pixi y rattler-build: instale pixi desde pixi.sh y, a continuación, instálelo. `rattler-build`

  ```
  pixi global install rattler-build
  ```

- git: necesario para clonar el repositorio de muestras. SíWindows, git for Windows también proporciona un `bash` intérprete de comandos, que requieren algunas de las recetas de Windows muestra.
- Depósito de Amazon S3: un depósito de Amazon S3 para usar como canal conda. Puedes usar el depósito de adjuntos de tareas de tu granja de Deadline Cloud o crear un depósito independiente.

- AWS credenciales: configure las credenciales en su estación de trabajo mediante el `aws configure` comando o el `aws login` comando. Para obtener más información, consulte Configuración de AWS CLI en la Guía del usuario de AWS Command Line Interface .
- Permisos de IAM: (opcional) Para reducir el alcance de los permisos que tienen sus credenciales, puede utilizar una política AWS Identity and Access Management (IAM) que solo conceda los siguientes permisos en el bucket de Amazon S3 y el prefijo de canal que utilice (por ejemplo,): `/Conda/*`
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:DeleteObject`
  - `s3:ListBucket`
  - `s3:GetBucketLocation`

Publicar un paquete en un canal Amazon S3

Úselo `rattler-build publish` con un `s3://` destino para publicar un paquete en su canal conda de Amazon S3. Si el canal no existe en el bucket, lo `rattler-build` inicializa automáticamente. Antes de empezar, asegúrese de haber completado los requisitos previos.

El siguiente ejemplo publica la receta de muestras Blender 4.5 del repositorio de muestras de Deadline Cloud enGitHub. Puedes sustituirla por una receta diferente del repositorio de muestras o usar la tuya propia.

> **Note**
>
> Las aplicaciones de gran tamaño pueden requerir decenas de GB de espacio libre en disco para el archivo fuente, los archivos extraídos y la generación de resultados. Asegúrese de utilizar un disco con suficiente espacio disponible para el resultado de la compilación del paquete.

Para publicar un paquete en un canal Amazon S3

1. Clona el repositorio de muestras de Deadline Cloud.

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

2. Cambie al directorio de `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Ejecute el comando siguiente. Reemplace *amzn-s3-demo-bucket* con el nombre del bucket.

```
rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
Conda/Default --build-number=+1
```

El `/Conda/Default` prefijo organiza el canal dentro del depósito. Puedes usar un prefijo diferente, pero el prefijo debe ser coherente en todos los comandos y configuraciones de cola que hacen referencia al canal.

> **Acerca de los números de compilación**
>
> La `--build-number=+1` opción selecciona automáticamente el siguiente número de compilación en función de lo que ya existe en el canal de destino. La mejor práctica es no sobrescribir nunca un paquete en un canal. Cree siempre con un número de compilación nuevo si, de lo contrario, el paquete tendría el mismo nombre de archivo. Esto se `--build-number=+1` consigue al crear un canal de producción o un canal provisional que refleje la producción.
>
> Si quieres controlar el número de compilación directamente, puedes configurarlo con un valor específico, como`--build-number=7`. Si omite la opción, `rattler-build` utiliza el número de compilación definido en el `recipe.yaml` archivo.

Si la receta de su paquete depende de paquetes de un canal en particular, como conda-forge, añada `-c conda-forge` al comando.

También puede publicar un archivo de paquete que ya haya creado, por ejemplo, un `.conda` archivo de una compilación local. Reemplace *amzn-s3-demo-bucket* con el nombre del bucket.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Inicializar o reindexar un canal

Cuando se publica un paquete, el comando inicializa el canal automáticamente si el canal aún no existe. `rattler-build publish` En la mayoría de los casos, no es necesario inicializar o reindexar el canal manualmente.

Es posible que tengas que inicializar o reindexar un canal manualmente en las siguientes situaciones:

- Quieres crear un canal vacío antes de publicar cualquier paquete, por ejemplo, para comprobar que tu entorno de colas de Deadline Cloud puede conectarse al canal.
- Ha cargado o eliminado `.conda` archivos directamente con las herramientas de Amazon S3 en lugar de `rattler-build publish` utilizarlos y el índice de canales está desactualizado.

Inicializar un canal vacío

Para inicializar un canal vacío, cree un `repodata.json` archivo y cárguelo en el `noarch` subdirectorio del prefijo del canal. Reemplace *amzn-s3-demo-bucket* con el nombre del 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
```

El `/Conda/Default` prefijo debe coincidir con el prefijo del canal que utiliza el entorno de colas. Después de inicializar el canal, puede publicar paquetes en el canal utilizando. `rattler-build publish`

Reindexar un canal

Si el índice del canal está desactualizado, utilícelo `rattler-index` para reconstruir el índice a partir de los archivos del paquete del canal. Primero, instale`rattler-index`.

```
pixi global install rattler-index
```

A continuación, vuelva a indexar el canal. Reemplace *amzn-s3-demo-bucket* con el nombre del bucket.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

Probando el paquete

Tras publicar el paquete, cree un proyecto pixi temporal para comprobar que el paquete funciona correctamente. El proyecto instala el paquete desde el canal Amazon S3.

Para probar el paquete

1. Cree un directorio de prueba temporal e inicialice un proyecto pixi con el canal Amazon S3. Reemplace *amzn-s3-demo-bucket* con el nombre del bucket.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. Añada el paquete al proyecto.

   ```
   pixi add blender=4.5
   ```

3. Compruebe que el paquete funciona correctamente.

   ```
   pixi run blender --version
   ```

   El `pixi run`comando activa el entorno conda para el directorio del proyecto y ejecuta el comando especificado en él. El entorno permanece en el directorio del proyecto, por lo que puede utilizar el mismo `pixi run` comando desde otros terminales.

Eliminar paquetes del canal

Evite eliminar paquetes de los canales que utiliza para la producción, ya que los archivos de bloqueo hacen referencia a paquetes específicos mediante un hash. Al eliminar un paquete, se impide volver a crear entornos a partir de esos archivos de bloqueo. En el caso de los canales de desarrollo y prueba, puedes eliminar un paquete específico si eliminas el `.conda` archivo del depósito y, a continuación, vuelves a indexar el canal.

Elimine el archivo del paquete y, a continuación, vuelva a indexar el canal. Reemplace *amzn-s3-demo-bucket* con el nombre del bucket.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

Tras eliminar el archivo, vuelva a indexar el canal para actualizar los metadatos del canal. Para obtener instrucciones, consulte Reindexación de un canal.

Package files se almacenan en subdirectorios específicos de la plataforma, como `linux-64`, o. `win-64` `osx-arm64` Para enumerar los paquetes de un subdirectorio, ejecute el siguiente comando.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

Limpieza

Tras la prueba, elimine el directorio del proyecto de prueba.

Para limpiar los recursos de prueba

- Elimine el directorio del proyecto de prueba.

  Active Linux y macOS ejecute el siguiente comando.

  ```
  rm -rf package-test-env
  ```

  En Windows (cmd), ejecute el siguiente comando.

  ```
  rmdir /s /q package-test-env
  ```

Depuración de compilaciones

Si una compilación falla, `rattler-build` conserva el directorio de compilación para que puedas investigarlo. Ejecuta el siguiente comando para abrir un shell interactivo en el entorno de compilación con todas las variables de entorno configuradas tal y como estaban durante la compilación.

```
rattler-build debug shell
```

Desde el shell de depuración, puedes modificar archivos, ejecutar comandos de compilación individuales y añadir dependencias para aislar el problema. Para obtener más información, consulta la sección Depuración de compilaciones en la documentación de rattler-build.

Creación de paquetes para otras plataformas

El `rattler-build publish` comando crea paquetes para el sistema operativo de la estación de trabajo en la que se ejecuta el comando. Si su flota de Deadline Cloud usa un sistema operativo diferente al de su estación de trabajo, o si su paquete tiene otros requisitos de host, tiene las siguientes opciones:

- Se ejecuta `rattler-build publish` en un host que coincida con el sistema operativo de destino. Por ejemplo, utilice una instancia de Amazon Elastic Compute Cloud (Amazon EC2) que se Linux ejecute para crear paquetes para una flota. Linux
- Usa una cola de creación de paquetes de Deadline Cloud para automatizar las compilaciones en la plataforma de destino. Consulte Crear una cola de creación de paquetes.
- (Avanzado) Utilice la compilación cruzada para crear paquetes para una plataforma diferente a la de su estación de trabajo. Para obtener más información, consulte Compilación cruzada en la documentación de rattler-build.

Siguientes pasos

Después de publicar los paquetes en su canal conda de Amazon S3, configure sus colas de Deadline Cloud para usar el canal:

- Configure los permisos de las colas de producción para paquetes conda personalizados: conceda a sus colas de producción acceso de solo lectura al canal conda de Amazon S3.
- Añadir un canal conda a un entorno de cola: configure el entorno de cola para instalar paquetes desde el canal conda de Amazon S3.

Configure los permisos de la cola de producción para paquetes conda personalizados

Su cola de producción necesita permisos de solo lectura para el `/Conda` prefijo del bucket S3 de la cola. Abre la página AWS Identity and Access Management (IAM) del rol asociado a la cola de producción y modifica la política de la siguiente manera:

1. Abra la consola de Deadline Cloud y vaya a la página de detalles de la cola de creación del paquete.
2. Elige la función de servicio de colas y, a continuación, selecciona Editar cola.

3. Ve a la sección Función de servicio de colas y, a continuación, selecciona Ver esta función en la consola de IAM.
4. En la lista de políticas de permisos, elija la que desee AmazonDeadlineCloudQueuePolicypara su cola.
5. En la pestaña Permisos, selecciona Editar.
6. Añada una nueva sección a la función de servicio de colas, como se muestra a continuación. Sustituya *amzn-s3-demo-bucket* y *111122223333* por su propio depósito y cuenta.

```
{
   "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"
    }
   }
  },
```

Añada un canal conda a un entorno de colas

Para usar el canal conda S3, debes añadir la ubicación del `s3://amzn-s3-demo-bucket/Conda/Default` canal al `CondaChannels` parámetro de los trabajos que envíes a Deadline Cloud. Los remitentes proporcionados por Deadline Cloud proporcionan campos para especificar paquetes y canales conda personalizados.

Puede evitar modificar todos los trabajos editando el entorno de colas conda para su cola de producción. Use el procedimiento siguiente:

1. Abre la consola de Deadline Cloud y navega hasta la página de detalles de la cola de producción.
2. Selecciona la pestaña de entornos.

3. Seleccione el entorno de colas de Conda y, a continuación, elija Editar.
4. Elija el editor JSON y, a continuación, busque en el script la definición del parámetro. `CondaChannels`
5. Edite la línea `default: "deadline-cloud"` para que comience con el canal conda S3 recién creado:

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Las flotas gestionadas por el servicio permiten una prioridad de canal flexible para conda de forma predeterminada. En el `blender=4.5` caso de una solicitud de trabajo en la que la versión Blender 4.5 esté tanto en el canal nuevo como en el `deadline-cloud` canal, el paquete se extraerá del canal que aparezca primero en la lista de canales. Si no se encuentra una versión de paquete específica en el primer canal, se comprobarán los canales subsiguientes para ver la versión del paquete.

Para las flotas gestionadas por el cliente, puede habilitar el uso de paquetes conda utilizando uno de los ejemplos del entorno de colas de conda del repositorio de muestras de Deadline Cloud. GitHub

Cree un paquete conda para una aplicación o un complemento

Un paquete conda es un archivo comprimido de software escrito en cualquier idioma. Conda admite una variedad de combinaciones de sistemas operativos y arquitecturas, por lo que puede empaquetar aplicaciones completas como BlenderMaya, y Nuke junto con bibliotecas para Python y otros lenguajes. Para obtener más información sobre los paquetes de conda, consulte Paquetes en la documentación de conda.

Para usar un paquete conda, debe instalarlo en un entorno virtual. Un entorno virtual conda tiene un directorio de prefijos donde se instalan los paquetes. La instalación de un paquete implica vincular o volver a vincular los archivos de forma rígida cuando se admite, por lo que la creación de varios entornos con los mismos paquetes no requiere una cantidad significativa de espacio adicional en disco. Para usar un entorno virtual, debe activarlo para establecer variables de entorno. La activación ejecuta los scripts que proporcionan los paquetes, lo que da a cada paquete la oportunidad de modificar PATH u otras variables de entorno. Los paquetes de Conda suelen contener aplicaciones o bibliotecas, pero la activación flexible significa que también pueden apuntar a aplicaciones instaladas en un sistema de archivos compartido.

La creación de un paquete personalizado consta de tres etapas: una receta contiene las instrucciones de construcción, un paquete es el artefacto (`.condao .tar.bz2` archivo) creado y un canal aloja los paquetes para la instalación. El `rattler-build publish` comando gestiona los tres pasos: puede crear una receta en un paquete y publicarla en un canal, o puede utilizar directamente un artefacto del paquete para publicarla.

La comunidad de conda-forge mantiene recetas de paquetes para un amplio conjunto de software de código abierto y aloja los artefactos de paquetes en el canal. `conda-forge` Puede configurar su cola para incluirla `conda-forge` como fuente de paquetes y, a continuación, crear paquetes personalizados que dependan de los paquetes de conda-forge para ejecutarse. PuesLinux, conda-forge aloja una cadena completa de herramientas de compilación que incluye soporte para CUDA, con opciones consistentes de compilación y enlace seleccionadas. Puedes usar los paquetes de conda-forge como dependencias en tus propias recetas o instalarlos junto con tus paquetes personalizados en el mismo entorno.

Puede combinar una aplicación completa, incluidas las dependencias, en un paquete conda. Los paquetes que Deadline Cloud proporciona en el canal Deadline-Cloud para flotas gestionadas por servicios utilizan este enfoque de reempaquetado binario. Esto organiza los mismos archivos que una instalación para adaptarlos al entorno virtual de conda.

> **Note**
>
> Las aplicaciones de gran tamaño pueden requerir decenas de GB de espacio libre en disco para el archivo fuente, los archivos extraídos y la generación de resultados. Asegúrese de utilizar un disco con suficiente espacio disponible para el resultado de la compilación del paquete.

Package una aplicación

Al volver a empaquetar una solicitud para conda, hay dos objetivos:

- La mayoría de los archivos de la aplicación deben estar separados de la estructura principal del entorno virtual de conda. Luego, los entornos pueden mezclar la aplicación con paquetes de otras fuentes, como conda-forge.
- Cuando se activa un entorno virtual conda, la aplicación debería estar disponible en la variable de entorno PATH.

Para volver a empaquetar una aplicación para conda

1. Escriba recetas de compilación de conda que instalen la aplicación en un subdirectorio como. `$CONDA_PREFIX/opt/<application-name>` Esto lo separa de los directorios de prefijos estándar como y. `bin` `lib`
2. Agregue enlaces simbólicos o ejecute scripts `$CONDA_PREFIX/bin` para ejecutar los binarios de la aplicación.

   Como alternativa, cree scripts activados.d que ejecutará el `conda activate` comando para añadir los directorios binarios de la aplicación a la PATH. Si no se admiten enlaces simbólicos en Windows todos los entornos en los que se puedan crear entornos, utilice en su lugar scripts de inicio de aplicaciones o activate.d.
3. Algunas aplicaciones dependen de bibliotecas que no están instaladas de forma predeterminada en las flotas gestionadas por el servicio de Deadline Cloud. Por ejemplo, el sistema de ventanas X11 no suele ser necesario para trabajos no interactivos, pero algunas aplicaciones aún requieren que se ejecute sin una interfaz gráfica. Debe proporcionar esas dependencias en el paquete que cree.
4. Si la aplicación admite complementos, proporcione una convención clara que los paquetes de complementos deben seguir para integrarse con la aplicación en un entorno virtual. Por ejemplo, la receta de ejemplo de Maya 2026 documenta esta convención para Maya los complementos.
5. Asegúrese de cumplir con los acuerdos de derechos de autor y licencia de las aplicaciones que empaquete. Le recomendamos que utilice un bucket privado de Amazon S3 para su canal conda a fin de controlar la distribución y limitar el acceso de los paquetes a su granja.

Los ejemplos de recetas para los paquetes del `deadline-cloud` canal están disponibles en el repositorio de muestras de Deadline Cloud enGitHub.

Package un plugin

Los complementos de la aplicación se pueden empaquetar como sus propios paquetes conda. Al crear un paquete de complementos, siga estas pautas:

- Incluye el paquete de la aplicación host como dependencia de compilación y ejecución en la receta de compilación`recipe.yaml`. Usa una restricción de versión para que la receta de compilación solo se instale con paquetes compatibles.
- Siga las convenciones del paquete de la aplicación anfitriona para registrar el complemento.

Paquetes de adaptadores

Algunas integraciones de aplicaciones de Deadline Cloud utilizan un adaptador que amplía la interfaz de la aplicación para simplificar la redacción de plantillas de trabajo. Un adaptador es una interfaz de línea de comandos que permite ejecutar un daemon en segundo plano, informar del estado y aplicar un mapeo de rutas. Para obtener más información, consulte Open Job Description Adaptor Runtime enGitHub. Por ejemplo, deadline-cloud-for-mayaon GitHub incluye una interfaz gráfica de usuario integrada para el envío de trabajos y un Maya adaptador que está disponible como `maya-openjd` paquete en las flotas gestionadas por servicios.

Los envíos de trabajos del remitente de Deadline Cloud GUIs incluyen un valor de `CondaPackages` parámetro que especifica los paquetes conda que se van a incluir en un entorno virtual para ejecutar el trabajo. El valor del `CondaPackages` parámetro Maya suele tener el mismo aspecto `maya=2026.* maya-openjd=0.15.* maya-mtoa` y puede contener entradas alternativas para los paquetes de complementos. Cuando el entorno de colas configura un entorno virtual conda para ejecutar el trabajo, resuelve estos nombres de paquetes y restricciones de versión para que sean compatibles y agrega todos los paquetes de dependencia que necesitan para ejecutarse. Cada paquete de adaptadores y complementos especifica con qué es compatible, incluidas las versionesMaya, las versiones de Python y otras dependencias.

Para crear sus propios paquetes de adaptadores utilizando nuestros ejemplos, como la receta maya-openjdGitHub, puede crear sobre los paquetes para Python y otras dependencias que proporciona conda-forge. Es posible que primero tengas que establecer la fecha límite y las recetas para satisfacer las dependencias. openjd-adaptor-runtime

Cree una receta de construcción de condas para Blender

Blenderes de uso gratuito y fácil de empaquetar con conda, lo que lo convierte en un buen punto de partida para aprender a crear paquetes de conda para Deadline Cloud (AWS Deadline Cloud). La Blender Fundación proporciona archivos de aplicaciones para varios sistemas operativos. La receta de ejemplo Blender 4.5 del repositorio de muestras de Deadline Cloud GitHub empaqueta estos archivos en un paquete conda.

¿Entendiendo la receta

El archivo recipe.yaml define los metadatos, el origen URLs y las opciones de compilación del paquete en la sintaxis de la plantilla rattler-build. La receta especifica el número de versión una vez y proporciona una fuente diferente según el sistema operativo. URLs

La `build` sección `recipe.yaml` desactiva las comprobaciones de reubicación binaria y vinculación de objetos compartidos dinámicos (DSO). Estas opciones controlan el funcionamiento del paquete cuando se instala en un entorno virtual conda con cualquier prefijo de directorio. Los valores predeterminados utilizados en la `build` sección están diseñados para empaquetar cada biblioteca de dependencias por separado, pero al volver a empaquetar una aplicación de forma binaria, es necesario cambiarlos. Blenderno requiere ningún ajuste de RPATH porque los archivos de la aplicación se crean teniendo en cuenta la reubicabilidad. Consulte Crear una receta de conda para Maya para ver un ejemplo de cómo añadir la reubicabilidad.

Durante la creación del paquete, se ejecuta el script build.sh o build_win.sh para instalar los archivos en el entorno. Estos scripts copian los archivos de instalación`$PREFIX/opt/blender`, crean enlaces simbólicos a partir de `$PREFIX/bin` (onLinux) y configuran scripts de activación que configuran variables de entorno como`BLENDER_LOCATION`. SíWindows, el script de activación añade el Blender directorio a la RUTA en lugar de crear enlaces simbólicos.

El script de Windows compilación utiliza un archivo.bat `bash` en lugar de un `cmd.exe` archivo.bat para mantener la coherencia entre las plataformas. Puedes instalar git for `bash` para Windows permitir la creación de paquetes.

La receta también incluye un `deadline-cloud.yaml` archivo que especifica las plataformas y los metadatos necesarios para enviar los trabajos automatizados de creación de paquetes a Deadline Cloud. Para obtener más información, consulte Enviar un trabajo de creación de paquetes.

Construyendo el Blender paquete

`rattler-build publish`Úselo para crear la receta Blender 4.5 y publicar el paquete en un canal. Puede publicar en un canal del sistema de archivos local para realizar pruebas o directamente en un canal de Amazon S3 para su uso en producción. Si completó la configuración en Compila y prueba paquetes de forma local, ejecute el siguiente comando desde el `conda_recipes` directorio.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Para otras opciones de publicación:

- Para publicar en un canal Amazon S3, consulte Publicar paquetes en un canal conda de S3.
- Para automatizar las compilaciones mediante una cola de creación de paquetes de Deadline Cloud, consulte Automatizar las compilaciones de paquetes con Deadline Cloud.

Pruebe su paquete con un trabajo de Blender renderizado

Después de crear el paquete Blender 4.5, puede probarlo con un trabajo de renderizado. Si no tiene una Blender escena, descargue la escena Blender 3.5: Cozy Kitchen de la página de archivos de Blender demostración. El repositorio de muestras de Deadline Cloud contiene un paquete de `blender_render` tareas y un entorno de colas fijas que puede utilizar para realizar pruebas tanto locales como en la nube.

Pruebas a nivel local

Puede ejecutar la plantilla de trabajo en su estación de trabajo mediante la CLI de Open Job Description. Instale la CLI con`pip`.

```
pip install openjd-cli
```

Desde el `job_bundles` directorio del repositorio de muestras, ejecute el siguiente comando. */path/to/scene.blend*Sustitúyalo por la ruta al archivo de Blender escena.

```
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
```

La `--environment` opción aplica el entorno de cola conda, que crea un entorno virtual conda con los paquetes especificados en. `CondaPackages` El `CondaChannels` parámetro indica al entorno de cola dónde encontrar los paquetes. Si ha publicado en un canal de Amazon S3 en lugar de en un canal local, sustituya la `file://` ruta por la URL de su `s3://` canal.

Pruebas en Deadline Cloud

Tras configurar la cola de producción para utilizar el canal conda de Amazon S3, puede enviar el trabajo de renderizado a Deadline Cloud. Desde el `job_bundles` directorio del repositorio de muestras, ejecute el siguiente comando.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Utilice el monitor de Deadline Cloud para realizar un seguimiento del progreso del trabajo. En el monitor, selecciona la tarea para el trabajo y selecciona Ver registros. Seleccione la acción Iniciar sesión de Conda para comprobar que el paquete se ha encontrado en el canal Amazon S3.

Cree una receta de construcción de conda para Autodesk Maya

Las aplicaciones comerciales Autodesk Maya introducen requisitos de empaquetado adicionales en comparación con las aplicaciones de código abierto, por ejemplo. Blender La Blenderreceta empaqueta un archivo simple y reubicable bajo una licencia de código abierto. Las aplicaciones comerciales suelen distribuirse a través de instaladores y requieren una configuración de administración de licencias.

Consideraciones para las aplicaciones comerciales

Las siguientes consideraciones se aplican al empaquetar aplicaciones comerciales. Los detalles ilustran cómo se aplica cada una de ellas aMaya.

- Licencias: comprenda los derechos de licencia y las restricciones de la aplicación. Es posible que necesite configurar un sistema de administración de licencias. Lea las preguntas frecuentes sobre las ventajas de la Autodesk suscripción sobre los derechos en la nube para entender cuáles son los derechos en la nubeMaya. Autodesklos productos se basan en un `ProductInformation.pit` archivo que normalmente requiere el acceso de un administrador para configurarlos. Las características del producto para clientes ligeros ofrecen una alternativa reubicable. Consulte Thin Client Licensing for Maya y MotionBuilder para obtener más información.
- Dependencias de las bibliotecas del sistema: algunas aplicaciones dependen de bibliotecas que no están instaladas en los hosts de Fleet Worker gestionados por el servicio. Mayadepende de las bibliotecas, incluidas freetype y fontconfig. Cuando estas bibliotecas estén disponibles en el administrador de paquetes del sistema, por ejemplo AL2023, `dnf` for, puede usar el administrador de paquetes como fuente. Como los paquetes RPM no están diseñados para ser reubicables, es necesario utilizar herramientas como las que permiten `patchelf` resolver las dependencias dentro del Maya prefijo de instalación.
- Acceso de administrador para la instalación: algunos instaladores requieren acceso de administrador. Las flotas gestionadas por el servicio no proporcionan acceso de administrador, por lo que es necesario instalar la aplicación en un sistema independiente y crear un archivo con los archivos para la compilación del paquete. El Windows instalador Maya requiere este enfoque. El README.md de la receta documenta un procedimiento repetible mediante una instancia recién lanzada de Amazon Elastic Compute Cloud (Amazon EC2).

- Integración de complementos: el Maya paquete de muestra define cómo aislar la aplicación de la configuración `MAYA_NO_HOME=1` a nivel de usuario y añade rutas de búsqueda de módulos para `MAYA_MODULE_PATH` que los paquetes de complementos puedan colocar `.mod` los archivos en el entorno virtual. Consulte la receta de ejemplo de Maya 2026 para conocer la convención completa de integración de complementos.

¿Entendiendo la receta

El archivo recipe.yaml define los metadatos del paquete en la sintaxis de la plantilla rattler-build.
Revisa las siguientes secciones del archivo:

- source: hace referencia a los archivos del instalador, incluido el hash sha256. SíLinux, la fuente es el archivo del Autodesk instalador. ActivadoWindows, la fuente incluye tanto el archivo del instalador como un `cleanMayaForCloud.py` script Autodesk que prepara Maya el despliegue en la nube. Actualice los hashes al cambiar los archivos de origen, por ejemplo, al empaquetar una nueva versión.
- build: desactiva la reubicación binaria predeterminada y las comprobaciones de vinculación de DSO porque los mecanismos automáticos no funcionan correctamente en la biblioteca y los directorios binarios que se utilizan. Maya ActivadaLinux, la receta incluye una dependencia `patchelf` de compilación para establecer manualmente los valores relativos. RPATHs
- about: metadatos sobre la aplicación para navegar o procesar el contenido de un canal conda.

Los scripts de compilación (build.sh paraLinux, build_win.sh paraWindows) incluyen comentarios que explican cada paso. Los scripts realizan las siguientes tareas clave:

- Extraer el instalador: extrae los archivos Maya de instalación en el prefijo conda. Los Windows scripts Linux y gestionan esto de forma diferente debido a los formatos del instalador. Consulte los scripts de compilación para obtener más información.
- Instalar las dependencias de las bibliotecas del sistema: si está activadoLinux, el script descarga y extrae las bibliotecas del sistema que se Maya necesitan, pero que no están presentes en los hosts de flotas gestionados por el servicio. El script copia estas bibliotecas en el Maya `lib` directorio para que estén disponibles en el entorno conda.
- Establecer como relativo RPATHs con patchelf: activadoLinux, el script utiliza `patchelf --add-rpath` para añadir rutas `$ORIGIN` relativas a las bibliotecas compartidas. Este enfoque sigue la recomendación de conda de no usarlo `LD_LIBRARY_PATH` nunca en entornos de conda. El script parchea las bibliotecas en varios niveles de directorio (`lib`/`lib/python*/`

`site-packages,,lib/python*/lib-dynload`) para que cada biblioteca pueda encontrar sus dependencias en relación con su propia ubicación. La receta sigue la práctica recomendada de configurar `DT_RUNPATH` en lugar de`DT_RPATH`, lo que `LD_LIBRARY_PATH` permite anular la ruta de búsqueda cuando sea necesario para la depuración.

- Configurar las licencias para clientes ligeros: el script configura las licencias para clientes ligeros, tal como lo ha documentado, de Autodesk modo que el `ProductInformation.pit` archivo se pueda ubicar en el entorno de conda en lugar de requerir el acceso de un administrador a nivel del sistema.
- Configurar scripts de activación: los scripts crean scripts de activación y desactivación que establecen variables de entorno`MAYA_LOCATION`, como, `MAYA_VERSION` y. `MAYA_NO_HOME` `MAYA_MODULE_PATH` SíWindows, los scripts producen ambos `.sh` archivos de `.bat` activación, ya que los entornos de colas de muestras de Deadline Cloud utilizan `bash` para activar los entornos. Windows

Construyendo el paquete Maya

Antes de crear el Maya paquete, descargue el Maya instalador de su Autodesk cuenta. Para Linux ello, coloque el archivo directamente en el `conda_recipes/archive_files` directorio. Para elloWindows, siga el procedimiento del archivo README.md para crear el archivo.

Se utiliza `rattler-build publish` para crear y publicar el paquete. La Maya receta requiere `patchelf` una dependencia de compilaciónLinux, que está disponible en conda-forge. `-c conda-forge`Añádala para que la dependencia esté disponible durante la compilación. Desde el `conda_recipes` directorio, ejecuta el siguiente comando.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Para otras opciones de publicación:

- Para publicar en un canal Amazon S3, consulte Publicar paquetes en un canal conda de S3.
- Para automatizar las compilaciones mediante una cola de creación de paquetes de Deadline Cloud, consulte Automatizar las compilaciones de paquetes con Deadline Cloud. Para compilar ambos Linux Windows paquetes, usa la `--all-platforms` opción junto con el `submit-package-job` script.

Para renderizar la muestra del tocadiscos con Maya yArnold, crea los paquetes de MtoAcomplementos y Mayaadaptadores. Después de publicar los tres paquetes, puedes enviar un trabajo de renderizado de prueba utilizando el paquete de tocadiscos conMaya/Arnoldjob del repositorio de muestras de Deadline Cloud. Consulte Pruebe sus paquetes con un trabajo de renderizado de Maya.

Cree una receta de construcción de conda para el adaptador Maya

El `maya-openjd` paquete incluye el adaptador que se integra Maya con las solicitudes de trabajo de AWS Deadline Cloud (Deadline Cloud). Cuando envías un trabajo de Maya renderizado mediante una GUI de presentación de Deadline Cloud, el `CondaPackages` parámetro se incluye `maya-openjd` junto con el `maya` paquete. El adaptador gestiona el inicioMaya, la comunicación de los parámetros de renderizado y la gestión del ciclo de vida de la aplicación durante una sesión de trabajo. Para obtener más información sobre los adaptadores, consulte Paquetes de adaptadores.

Entendiendo la receta

La receta de ejemplo maya-openjd crea el adaptador a partir del paquete deadline-cloud-for-mayafuente publicado en PyPI. El archivo recipe.yaml instala el paquete utilizando el entorno conda. `pip`

La receta depende de Python y de otros dos paquetes del repositorio de muestras de Deadline Cloud que debes compilar primero:

- deadline: la biblioteca de clientes de Deadline Cloud.
- openjd-adaptor-runtime— El tiempo de ejecución del adaptador Open Job Description.

Python y otras dependencias están disponibles en conda-forge, así que agréguelo `-c conda-forge` al `rattler-build publish` comando cuando cree el paquete del adaptador.

Construir el paquete de adaptadores

El `maya-openjd` paquete depende de otros dos paquetes del repositorio de muestras de Deadline Cloud. Cree los tres paquetes en orden desde el `conda_recipes` directorio. La `-c conda-forge` opción de cada comando es satisfacer las dependencias de recetas para Python y las bibliotecas de Python.

Compila el `deadline` paquete.

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Construye el `openjd-adaptor-runtime` paquete.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Construye el `maya-openjd` paquete.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Para otras opciones de publicación:

- Para publicar en un canal Amazon S3, consulte Publicar paquetes en un canal conda de S3.
- Para automatizar las compilaciones mediante una cola de creación de paquetes de Deadline Cloud, consulte Automatizar las compilaciones de paquetes con Deadline Cloud.

Crea una receta de compilación de conda para el plugin Autodesk Maya to Arnold (MtoA)

El Maya to Arnold (MtoA) complemento añade el Arnold renderizador como una opción internaMaya. La receta de ejemplo de mToA muestra cómo empaquetar un complemento como un paquete conda independiente que se integra con el paquete de la aplicación host.

¿Entendiendo la receta

El archivo recipe.yaml especifica una dependencia del `maya` paquete para los requisitos de compilación y ejecución. Esta dependencia usa una restricción de versión para que el complemento solo se instale con una versión compatible. Maya

La receta utiliza los mismos archivos fuente que la Maya receta. El script de compilación instala MtoA y crea un `mtoa.mod` archivo en el `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` directorio en el que se configura el Maya paquete. `MAYA_MODULE_PATH` Arnoldy Maya utilizan la misma tecnología de licencias, por lo que el Maya paquete ya incluye la información de licencia necesaria. Arnold

Construyendo el MtoA paquete

Compila el Maya paquete antes de crearloMtoA, ya que MtoA depende del momento de Maya la compilación. Se utiliza `rattler-build publish` para crear y publicar el paquete. Desde el `conda_recipes` directorio, ejecute el siguiente comando.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

El `rattler-build publish` comando utiliza el canal de destino como el canal de mayor prioridad al resolver las dependencias, por lo que el `maya` paquete que ha publicado anteriormente está disponible automáticamente.

Para otras opciones de publicación:

- Para publicar en un canal Amazon S3, consulte Publicar paquetes en un canal conda de S3.
- Para automatizar las compilaciones mediante una cola de creación de paquetes de Deadline Cloud, consulte Automatizar las compilaciones de paquetes con Deadline Cloud.

Pruebe sus paquetes con un trabajo de Maya renderizado

Después de crear los `maya-openjd` paquetes y MayaMtoA, puedes probarlos con un trabajo de renderizado. El repositorio de muestras de Deadline Cloud contiene un tocadiscos con el paquete de Arnold tareasMaya/que renderiza una animación mediante Maya y. Arnold El paquete de tareas también se utiliza FFmpeg para codificar un vídeo, que está disponible en el canal. `conda-forge`

Pruebas a nivel local

Puede ejecutar la plantilla de trabajo en su estación de trabajo mediante la CLI de Open Job Description. Instale la CLI con`pip`.

```
pip install openjd-cli
```

Desde el `job_bundles` directorio del repositorio de muestras, ejecute el siguiente comando. El `ErrorOnArnoldLicenseFail=false` parámetro indica Arnold que hay que renderizar con marcas de agua en lugar de fallar cuando no hay ninguna licencia disponible.

```
openjd run turntable_with_maya_arnold/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages="maya maya-mtoa maya-openjd ffmpeg" \
    -p CondaChannels="file://$HOME/my-conda-channel conda-forge" \
    -p ErrorOnArnoldLicenseFail=false \
    -p FrameRange=1-5
```

La `--environment` opción aplica el entorno de cola conda, que crea un entorno virtual conda con los paquetes especificados en. `CondaPackages` El `CondaChannels` parámetro incluye tanto el canal local para sus paquetes personalizados como para. `conda-forge` `ffmpeg` Si ha publicado en un canal de Amazon S3 en lugar de en un canal local, sustituya la `file://` ruta por la URL de su `s3://` canal.

Cuando se complete el trabajo, el resultado renderizado estará en el `turntable_with_maya_arnold/output/` directorio.

Probando en Deadline Cloud

Tras configurar la cola de producción para utilizar el canal conda de Amazon S3, envíe el trabajo de renderizado a Deadline Cloud. Agregue el `conda-forge` canal al `CondaChannels` parámetro en su entorno de cola conda para proporcionar una fuente `ffmpeg` y las dependencias de Python que requiere el adaptador. Desde el `job_bundles` directorio del repositorio de muestras, ejecuta el siguiente comando.

```
deadline bundle submit turntable_with_maya_arnold
```

Utilice el monitor de Deadline Cloud para realizar un seguimiento del progreso del trabajo. En el monitor, selecciona la tarea para el trabajo y selecciona Ver registros. Seleccione la acción Iniciar sesión de Conda para comprobar que los `maya-openjd` paquetes `mayamaya-mtoa`, y se han encontrado en el canal Amazon S3.

Automatice la creación de paquetes con Deadline Cloud

Para los CI/CD flujos de trabajo o cuando necesite crear paquetes para varios sistemas operativos, puede crear una cola de creación de paquetes de Deadline Cloud. Las listas de espera crean trabajos en su flota, que crean los paquetes y los publican en su canal conda de Amazon Simple

Storage Service (Amazon S3). Esto simplifica el mantenimiento continuo de la creación de paquetes para las versiones de software en todas las configuraciones requeridas.

Puedes crear una cola de creación de paquetes utilizando una plantilla AWS CloudFormation (CloudFormation) o manualmente desde la consola de Deadline Cloud. La CloudFormation plantilla despliega una granja completa con una cola de producción y una cola de creación de paquetes ya configuradas. Al crear la cola desde la consola, tendrá más control sobre los ajustes individuales.

Cree una cola de creación de paquetes con CloudFormation

Puedes usar una CloudFormation plantilla para crear una granja de Deadline Cloud que incluya una cola de creación de paquetes. La plantilla configura una cola de producción y una cola de creación de paquetes con un canal conda privado de Amazon S3.

Antes de implementar la plantilla, cree un bucket de Amazon S3 para almacenar los adjuntos de trabajo y su canal conda. Puede crear un bucket desde la consola de Amazon S3. Necesitará el nombre del bucket al implementar la plantilla.

Para implementar la CloudFormation plantilla

1. Descargue la plantilla deadline-cloud-starter-farm-template.yaml del repositorio de muestras de Deadline Cloud en. GitHub
2. Desde la CloudFormation consola, selecciona Crear pila y, a continuación, Con nuevos recursos (estándar).
3. Selecciona la opción de cargar un archivo de plantilla y, a continuación, sube el `deadline-cloud-starter-farm-template.yaml` archivo.
4. Introduzca un nombre para la pila, por ejemplo**`StarterFarm`**, y proporcione el nombre de un bucket de Amazon S3 para adjuntar trabajos y el canal conda.
5. Siga los pasos de la CloudFormation consola para completar la creación de la pila.

Para obtener más información sobre los parámetros de la plantilla y las opciones de personalización, consulta el README de la granja de inicio en el repositorio de muestras de Deadline Cloud enGitHub.

Cree una cola de creación de paquetes desde la consola

Sigue las instrucciones de la Guía del usuario de Deadline Cloud sobre cómo crear una cola. Realice los siguientes cambios:

- En el paso 5, elija un bucket de Amazon S3 existente. Especifica un nombre para la carpeta raíz, de **`DeadlineCloudPackageBuild`** forma que los artefactos de construcción permanezcan separados de los archivos adjuntos normales de Deadline Cloud.
- En el paso 6, puede asociar la cola de creación de paquetes a una flota existente, o puede crear una flota completamente nueva si su flota actual no es adecuada.
- En el paso 9, cree un nuevo rol de servicio para su cola de creación de paquetes. Modificará los permisos para conceder a la cola los permisos necesarios para cargar paquetes y volver a indexar un canal conda.

Configure los permisos de creación de colas de paquetes

Para permitir que la cola de creación de paquetes acceda al `/Conda` prefijo del bucket de Amazon S3 de la cola, debe modificar la función de la cola para darle acceso. read/write El rol necesita los siguientes permisos para que los trabajos de creación de paquetes puedan cargar nuevos paquetes y volver a indexar el canal.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Abre la consola de Deadline Cloud y navega hasta la página de detalles de la cola de creación de paquetes.
2. Elige la función de servicio de colas y, a continuación, selecciona Editar cola.
3. Ve a la sección Función de servicio de colas y, a continuación, selecciona Ver esta función en la consola de IAM.
4. En la lista de políticas de permisos, elija la que desee AmazonDeadlineCloudQueuePolicypara su cola.
5. En la pestaña Permisos, selecciona Editar.
6. Añada una nueva sección a la función de servicio de colas, como se muestra a continuación. Sustituya *`amzn-s3-demo-bucket`* y *`111122223333`* por su propio depósito y cuenta.

```
{
   "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"
     }
    }
   },
```

Envíe un trabajo de construcción de paquetes

Después de crear una cola de creación de paquetes y configurar los permisos de la cola, puede enviar trabajos para crear paquetes conda. El `submit-package-job` script del repositorio de muestras de Deadline Cloud GitHub envía un trabajo de compilación para una receta de conda.

Necesitará lo siguiente:

- La CLI de Deadline Cloud instalada en su estación de trabajo.
- Una sesión de inicio de sesión activa AWS en el monitor de Deadline Cloud (Deadline Cloud monitor).
- Un clon del repositorio de muestras de Deadline Cloud.

Para enviar un trabajo de creación de paquetes

1. Abre la GUI de configuración de Deadline Cloud y establece la granja y la cola predeterminadas en la cola de creación de paquetes.

   ```
   deadline config gui
   ```

2. Cambie al `conda_recipes` directorio del repositorio de muestras.

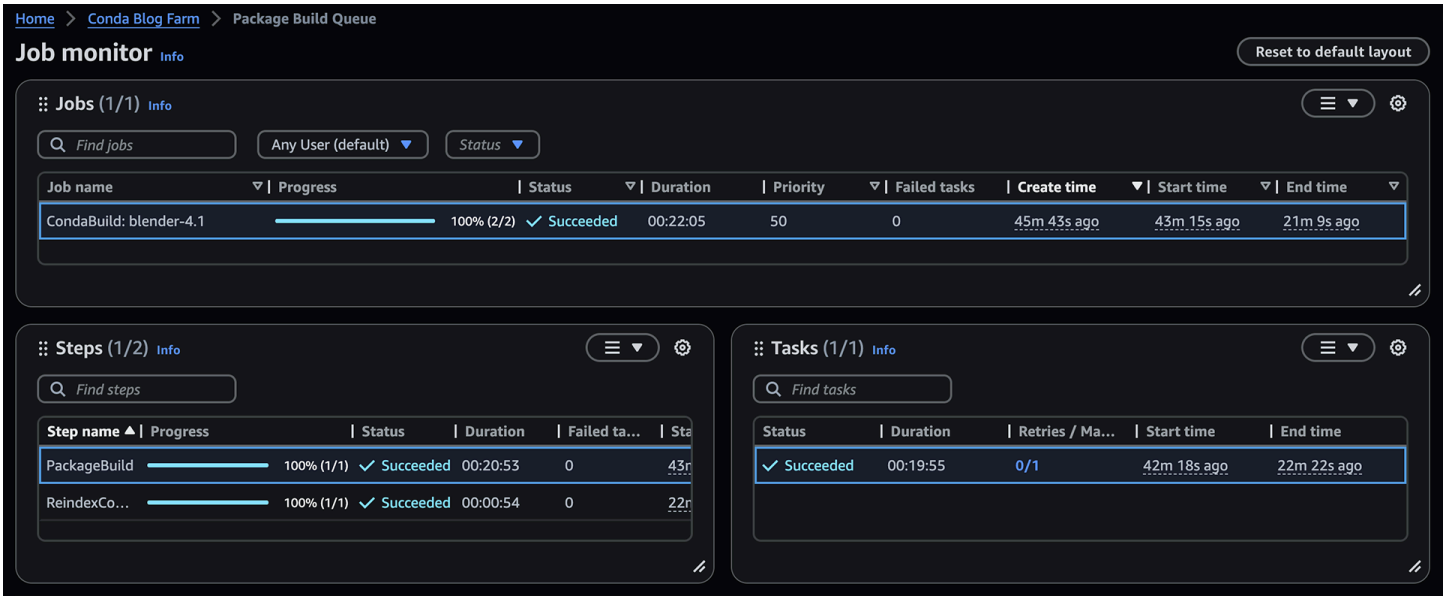
```
cd deadline-cloud-samples/conda_recipes
```

3. Ejecute el `submit-package-job` script con el directorio de recetas. El siguiente ejemplo crea la receta Blender 4.5.

```
./submit-package-job blender-4.5/
```

   Si la receta requiere un archivo fuente que aún no haya descargado, el script proporciona las instrucciones de descarga. Descargue el archivo y vuelva a ejecutar el script.

Después de enviar el trabajo, utilice el monitor de Deadline Cloud para ver el progreso y el estado del trabajo.



El monitor muestra los dos pasos del trabajo: crear el paquete y, a continuación, volver a indexar el canal conda. Al hacer clic con el botón derecho en la tarea correspondiente al paso de creación del paquete y seleccionar Ver registros, el monitor muestra las acciones de la sesión:

- Sincronizar los archivos adjuntos: copia los archivos adjuntos del trabajo de entrada o monta un sistema de archivos virtual.
- Inicie Conda, la acción del entorno de colas. El trabajo de compilación no especifica los paquetes de conda, por lo que esta acción finaliza rápidamente.
- Launch CondaBuild Env: crea un entorno virtual conda con el software necesario para crear un paquete conda y volver a indexar un canal.

- Ejecución de tareas: crea el paquete y carga los resultados en Amazon S3.

A medida que se ejecutan las acciones, envían registros a Amazon CloudWatch (CloudWatch). Cuando se complete un trabajo, seleccione Ver registros para todas las tareas para ver registros adicionales sobre la configuración y el desmontaje del entorno.

Ejecute scripts de configuración del host con privilegios de administrador

Los scripts de configuración del host le permiten realizar tareas administrativas, como la instalación de software, en los trabajadores de su flota gestionada por el servicio. Estos scripts se ejecutan con privilegios elevados (`sudo`Linuxactivados y de administrador activadosWindows), lo que le brinda la flexibilidad de configurar a sus trabajadores para su sistema.

Deadline Cloud ejecuta el script después de que el trabajador entre en el `STARTING` estado y antes de que ejecute cualquier tarea.

> **Important**
>
> El script se ejecuta con permisos elevados. Es su responsabilidad asegurarse de que el script no presente ningún problema de seguridad.
> Cuando utiliza un script de configuración de host, es responsable de supervisar el estado de su flota.

Los usos más comunes de los scripts de configuración del host incluyen:

- Instalar software que requiera acceso de administrador
- Instalación de Docker contenedores
- Instalación de soluciones de almacenamiento en la nube de terceros, comoLucidLink. Para ver un tutorial, consulte Configurar LucidLink con scripts de flota gestionados por servicios para Deadline Cloud en el AWS blog de M&E.

Puede crear y actualizar un script de configuración de host mediante la consola o mediante el. AWS CLI

Console

1. En la página de detalles de la flota, seleccione la pestaña Configuraciones.
2. En el campo Secuencia de comandos, introduzca la secuencia de comandos que desee ejecutar con permisos elevados. Puede elegir Importar para cargar un script desde su estación de trabajo.
3. Establezca un período de espera en segundos para ejecutar el script. El valor predeterminado es 300 segundos (5 minutos).
4. Seleccione Guardar cambios para guardar el script.

Create with CLI

Utilice el siguiente AWS CLI comando para crear una flota con un script de configuración del host. Sustituya el *placeholder* texto por su información.

```
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 el siguiente AWS CLI comando para actualizar el script de configuración del anfitrión de una flota. Sustituya el *placeholder* texto por su información.

```
aws deadline update-fleet \
```

```
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Los siguientes scripts muestran lo siguiente:

- Las variables de entorno disponibles para el script
- Esas AWS credenciales funcionan en el shell
- Que el script se ejecuta en un shell elevado

Linux

Utilice el siguiente script para mostrar que un script se está ejecutando con `root` privilegios:

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Utilice el siguiente PowerShell script para mostrar que un script se está ejecutando con privilegios de administrador:

```
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
```

Solucione problemas con los scripts de configuración del host

Al ejecutar el script de configuración del host:

- En caso de éxito: el trabajador ejecuta el trabajo
- En caso de fallo (código de salida distinto de cero o bloqueo):
  - El trabajador cierra

  La flota lanza automáticamente a un nuevo trabajador mediante el último script de configuración del host

Para supervisar el script:

1. Abre la página de la flota en la consola de Deadline Cloud.
2. Selecciona Ver trabajadores para abrir el monitor de Deadline Cloud.
3. Vea el estado del trabajador en la página del monitor.

> **Tip**
>
> Al probar los scripts de configuración del host, establezca el número máximo de trabajadores de la flota en 1 para evitar iniciar varios trabajadores mientras se sigue iterando en el script.

Notas importantes:

- Los trabajadores que dejan de trabajar debido a un error no están disponibles en la lista de trabajadores del monitor. Utilice CloudWatch los registros para ver los registros de los trabajadores del siguiente grupo de registros:

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

Dentro de ese grupo de registros, busca una transmisión con el nombre`worker-ZZZZZ`.

- CloudWatch Los registros conservan los registros de los trabajadores según el período de retención configurado.

Supervise la ejecución del script de configuración del host

Con los scripts de configuración del host, puedes tomar el control total de un trabajador de Deadline Cloud. Puede instalar cualquier paquete de software, reconfigurar los parámetros del sistema operativo o montar sistemas de archivos compartidos. Con esta función avanzada y la capacidad de Deadline Cloud de ampliarse a miles de trabajadores, puede supervisar si los scripts de configuración se ejecutan correctamente o si fallan.

Recomendamos las siguientes soluciones para supervisar la ejecución del script de configuración del host.

CloudWatch Supervisión de registros

Todos los registros de configuración del host de la flota se transmiten al grupo de CloudWatch registros de la flota y, específicamente, al flujo de CloudWatch registros del trabajador. Por ejemplo, `/aws/deadline/farm-123456789012/fleet-777788889999` es el grupo de registros de la granja `123456789012` o la flota`777788889999`.

Por ejemplo, cada trabajador aprovisiona un flujo de registro dedicado`worker-123456789012`. Los registros de configuración del host incluyen rótulos de registro como Running Host Configuration Script y Finished running Host Configuration Script, código de salida: 0. El código de salida del script se incluye en el encabezado final y se puede consultar mediante CloudWatch herramientas.

CloudWatch Registra e información

CloudWatch Logs Insights ofrece funciones avanzadas para analizar la información de los registros. Por ejemplo, la siguiente consulta de Log Insights analiza el código de salida de la configuración del host, ordenado por hora:

```
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
```

Para obtener más información sobre CloudWatch Logs Insights, consulte Análisis de datos de registro con CloudWatch Logs Insights en la Guía del usuario de Amazon CloudWatch Logs.

Registro estructurado por agentes de trabajo

El agente de trabajo de Deadline Cloud publica registros JSON estructurados en CloudWatch. El agente obrero ofrece una amplia gama de registros estructurados para analizar la salud de los trabajadores. Para obtener más información, consulte Inicio de sesión de un agente de trabajo en Deadline Cloud GitHub.

Los atributos de los registros estructurados se agrupan en campos en Log Insights. Puede utilizar esta CloudWatch capacidad para contar y analizar los errores de inicio de la configuración del host. Por ejemplo, se puede utilizar una consulta de recuento y contenedor para determinar la frecuencia con la que se producen los errores:

```
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 filtros métricos para métricas y alarmas

Puede configurar filtros de CloudWatch métricas para generar CloudWatch métricas a partir de los registros. Los filtros métricos le permiten crear alarmas y paneles para supervisar la ejecución del script de configuración del host.

Para crear un filtro de métricas

1. Abra la CloudWatch consola.
2. En el panel de navegación, seleccione Registros y, a continuación, Grupos de registros.
3. Seleccione el grupo de registros de su flota.
4. Elija Create metric filter (Crear filtro de métricas).
5. Defina su patrón de filtro mediante una de las siguientes opciones:

   - Para obtener métricas de éxito:

     ```
     {$.message = "*Worker Agent host configuration succeeded.*"}
     ```

   - Para las métricas de errores:

```
{$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
 exit code*"}
```

6. Elija Siguiente para crear una métrica con los siguientes valores:

   - Espacio de nombres de métricas: su espacio de nombres de métricas (por ejemplo,) **`MyDeadlineFarm`**
   - Nombre de la métrica: el nombre de la métrica solicitada (por ejemplo,) **`host_config_failure`**
   - Valor métrico: **`1`** (cada instancia es un recuento de 1)
   - Valor predeterminado: dejar en blanco
   - Unidad: **`Count`**

Tras crear los filtros de métricas, puede configurar CloudWatch las alarmas estándar para que actúen en caso de tasas elevadas de errores en la configuración del host, o bien añadir las métricas a un CloudWatch panel de control para day-to-day las operaciones y la supervisión.

Para obtener más información, consulte Sintaxis de filtros y patrones en la Guía del usuario de Amazon CloudWatch Logs.

Uso de licencias de software con Deadline Cloud

Deadline Cloud ofrece dos métodos para proporcionar licencias de software para sus trabajos:

- Licencias basadas en el uso (UBL): rastrean y facturan en función del número de horas que su flota dedica a procesar un trabajo. No hay un número fijo de licencias, por lo que su flota puede ampliarse según sea necesario. El UBL es el estándar para las flotas gestionadas por el servicio. Para las flotas gestionadas por el cliente, puede conectar un punto final de licencia de Deadline Cloud para UBL. UBL proporciona licencias para que las rendericen tus trabajadores de Deadline Cloud, pero no proporciona licencias para tus aplicaciones de DCC.
- Traiga su propia licencia (BYOL): le permite utilizar las licencias de software existentes con sus flotas gestionadas por el servicio o por el cliente. Puede usar BYOL para conectarse a servidores de licencias de software que no sean compatibles con las licencias basadas en el uso de Deadline Cloud. Puedes usar BYOL con flotas gestionadas por servicios conectándote a un servidor de licencias personalizado.

Temas

Combinación de BYOL y UBL

Puedes combinar BYOL y UBL para que tus trabajadores usen primero tus licencias existentes y recurran automáticamente a las licencias basadas en el uso de Deadline Cloud cuando se agoten las licencias BYOL. Este enfoque resulta útil cuando tienes un número limitado de licencias existentes, pero necesitas escalar más allá de esa capacidad durante los picos de carga de trabajo.

Cómo funcionan las licencias combinadas

Al configurar las licencias combinadas, el entorno de colas configura las variables de entorno de licencias para que el servidor de licencias BYOL aparezca antes del punto final de licencias UBL. La mayoría de las aplicaciones de terceros comprueban los servidores de licencias en el orden en que aparecen en la variable de entorno. Cuando un trabajador solicita una licencia, la aplicación contacta

primero con el servidor de licencias BYOL. Si no hay ninguna licencia BYOL disponible, la aplicación recurre al punto final de la licencia UBL.

La plantilla de entorno de colas BYOL que se proporciona en Connect las flotas gestionadas por el servicio a un servidor de licencias personalizado configura este comportamiento alternativo automáticamente. El script de Python del entorno de colas antepone la dirección del servidor de licencias BYOL a las variables de entorno de licencias UBL existentes. Para usar licencias combinadas, mantenga las secciones UBL en el script del entorno de colas para los productos en los que desee un comportamiento alternativo.

Para utilizar únicamente BYOL sin UBL como alternativa para un producto específico, elimine la sección UBL correspondiente a ese producto del script y añada la variable de entorno de licencias directamente a la sección del entorno de colas. `variables` Por ejemplo, para usar solo BYOL para Cinema 4D, elimina la sección Cinema 4D del script y agrégala a la sección. `g_licenseServerRLM: 127.0.0.1:7057` `variables`

Ejemplo: usar licencias BYOL de Cinema 4D con el respaldo UBL

Pensemos en un estudio que tiene licencias de Cinema 4D existentes en un servidor de licencias de su red local. El estudio quiere usar esas licencias para renderizar en Deadline Cloud, pero también quiere ir más allá de su número de licencias y recurrir a la UBL cuando todas las licencias BYOL estén en uso.

Para configurar esta configuración, sigue los pasos que se indican Connect las flotas gestionadas por el servicio a un servidor de licencias personalizado y realiza los siguientes cambios en la plantilla del entorno de colas:

Para configurar las licencias BYOL Cinema 4D con UBL de respaldo

1. Defina el `LicenseInstanceId` parámetro en el ID de instancia de Amazon Elastic Compute Cloud (Amazon EC2) del servidor de licencias o proxy que tiene acceso al servidor de licencias de Cinema 4D.
2. Configura el `LicensePorts` parámetro para incluir el puerto `7057` (el puerto de licencia RLM de Cinema 4D).
3. En el script de Python, conserva la sección Cinema 4D que antepone el servidor BYOL a la configuración 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']}")
```

Esta configuración se establece en. `g_licenseServerRLM` `127.0.0.1:7057;`*`UBL_endpoint`*`:7057` Cinema 4D comprueba `127.0.0.1:7057` primero el servidor BYOL. Si no hay ninguna licencia disponible, Cinema 4D recurre al punto final UBL.

4. Elimina las secciones de los productos que no utilices (por ejemplo, Arnold, Nuke o SideFX) para mantener limpia la configuración.

Si también tiene otros productos que utilizan únicamente BYOL sin el respaldo de UBL, añada esas variables de entorno de licencia directamente a la `variables` sección del entorno de colas y elimine las secciones correspondientes del script de Python.

Consideraciones para la concesión de licencias combinadas

Tenga en cuenta las siguientes consideraciones cuando utilice licencias combinadas:

- Algunas aplicaciones no admiten varios servidores de licencias en una sola variable de entorno. Por ejemplo, V-Ray utiliza en su lugar un archivo de configuración XML. La plantilla de entorno de colas gestiona la configuración de V-Ray por separado. Para obtener más información, consulte la sección V-Ray de la plantilla de entorno de colas en. Connect las flotas gestionadas por el servicio a un servidor de licencias personalizado
- El orden de los servidores de licencias en la variable de entorno determina la prioridad. Incluya primero el servidor BYOL para que las licencias existentes se consuman antes que las licencias UBL.
- En el Windows caso de los trabajadores, separe las entradas del servidor de licencias en las variables de entorno con un punto y coma (`;`) en lugar de dos puntos (). `:` Para obtener más información sobre la configuración de las variables de entorno de licencias, consulteConnect las flotas gestionadas por el cliente a un punto final de licencia.
- Para utilizar la opción alternativa de UBL con flotas gestionadas por el cliente, configure un punto final de licencia. Para obtener más información, consulte Connect las flotas gestionadas por el cliente a un punto final de licencia.

Connect las flotas gestionadas por el servicio a un servidor de licencias personalizado

Puedes usar tu propio servidor de licencias con una flota gestionada por el servicio de Deadline Cloud. Para traer su propia licencia, puede configurar un servidor de licencias mediante un entorno de colas en su granja. Para configurar el servidor de licencias, ya debe tener una granja y una cola configuradas.

La forma de conectarse a un servidor de licencias de software depende de la configuración de su flota y de los requisitos del proveedor del software. Por lo general, se accede al servidor de dos maneras:

- Directamente al servidor de licencias. Sus trabajadores obtienen una licencia del servidor de licencias del proveedor de software a través de Internet. Todos sus trabajadores deben poder conectarse al servidor.
- A través de un proxy de licencia. Sus trabajadores se conectan a un servidor proxy de su red local. Solo el servidor proxy puede conectarse al servidor de licencias del proveedor a través de Internet.

Siguiendo las instrucciones que aparecen a continuación, utilizará Amazon EC2 Systems Manager (SSM) para reenviar los puertos de una instancia de trabajo a su servidor de licencias o instancia proxy. En el siguiente ejemplo, si su servidor de licencias no puede proporcionar una licencia, se utilizarán las licencias basadas en el uso de Deadline Cloud. Elimine las secciones que no se apliquen a su cartera o a los productos para los que no desee utilizar licencias basadas en el uso una vez agotadas las licencias.

Temas

Paso 1: Configurar el entorno de colas

Puede configurar un entorno de colas en su cola para acceder a su servidor de licencias. En primer lugar, asegúrese de tener una AWS instancia configurada con acceso al servidor de licencias mediante uno de los siguientes métodos:

- Servidor de licencias: la instancia aloja los servidores de licencias directamente.
- Proxy de licencias: la instancia tiene acceso de red al servidor de licencias y reenvía los puertos del servidor de licencias al servidor de licencias. Para obtener más información sobre cómo configurar una instancia de proxy de licencias, consultePaso 2: Configuración (opcional) de la instancia de proxy de licencia.

Para obtener información sobre la configuración de las variables de entorno de licencia, consultePaso 3: Conectar una aplicación de renderizado a un punto final. Para una configuración de servidor de licencias personalizada, la dirección del servidor de licencias sigue siendo localhost en lugar del punto final de Amazon VPC.

Para añadir los permisos necesarios a la función de cola

1. En la consola de Deadline Cloud, selecciona Ir al panel de control.
2. En el panel de control, selecciona la granja y, a continuación, la cola que deseas configurar.
3. En los detalles de la cola > función de servicio, seleccione la función.
4. Selecciona Añadir permiso y, a continuación, selecciona Crear política integrada.
5. Selecciona el editor de políticas JSON y, a continuación, copia y pega el siguiente texto en el 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. Antes de guardar la nueva política, sustituya los siguientes valores en el texto de la política:
   - `region`Sustitúyalos por la AWS región en la que se encuentra su granja
   - `instance_id`Sustitúyalo por el ID de instancia del servidor de licencias o la instancia proxy que estés utilizando
   - `account_id`Sustitúyalo por el número de AWS cuenta que contiene tu granja
7. Elija Siguiente.
8. Para el nombre de la póliza, introduzca**`LicenseForwarding`**.
9. Seleccione Crear política para guardar los cambios y crear la política con los permisos necesarios.

Para añadir un nuevo entorno de colas a la cola

1. En la consola de Deadline Cloud, selecciona Ir al panel de control si aún no lo has hecho.
2. En el panel de control, selecciona la granja y, a continuación, la cola que deseas configurar.
3. Selecciona Entornos de cola **>** Acciones **>** Crear nuevos con YAML.
4. Copia y pega el siguiente texto en el editor de scripts de 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. Antes de guardar el entorno de colas, realice los siguientes cambios en el texto del entorno según sea necesario:

   - Actualice los valores predeterminados de los siguientes parámetros para que reflejen su entorno:
     - LicenseInstanceID: el ID de instancia de Amazon EC2 de su servidor de licencias o instancia proxy
     - LicenseInstanceRegion— La AWS región en la que se encuentra su granja
     - LicensePorts— Una lista de puertos separados por comas que se reenviará al servidor de licencias o a la instancia proxy (por ejemplo, 2700,2701)
   - Si desea utilizar las licencias basadas en el uso (UBL) una vez agotada la licencia Bring your own (BYOL), asegúrese de que el puerto es el correcto para su servidor de licencias. Si no desea utilizar la UBL después de quedarse sin BYOL, añada las variables de entorno de licencias necesarias a la sección de variables.

     Estas variables deberían dirigirla DCCs a localhost en el puerto del servidor de licencias. Por ejemplo, si su servidor de licencias de Foundry escucha en el puerto 6101, debe añadir la variable como. **`foundry_LICENSE: 6101@localhost`**
6. (Opcional) Puede dejar la prioridad establecida en 0 o puede cambiarla para ordenar la prioridad de forma diferente entre varios entornos de colas.
7. Seleccione Crear entorno de cola para guardar el nuevo entorno.

   Con el entorno de colas configurado, los trabajos enviados a esta cola recuperarán las licencias del servidor de licencias configurado.

Paso 2: Configuración (opcional) de la instancia de proxy de licencia

Como alternativa al uso de un servidor de licencias, puede utilizar un proxy de licencias. Para crear un proxy de licencias, cree una nueva instancia de Amazon Linux 2023 que tenga acceso de red al servidor de licencias. Si es necesario, puede configurar este acceso mediante una conexión VPN. Para obtener más información, consulte Conexiones VPN en la Guía del usuario de Amazon VPC.

Para configurar una instancia de proxy con licencia para Deadline Cloud, siga los pasos de este procedimiento. Realice los siguientes pasos de configuración en esta nueva instancia para permitir el reenvío del tráfico de licencias a su servidor de licencias

1. Para instalar el HAProxy paquete, introduzca

```
sudo yum install haproxy
```

2. Actualice la sección listen license-server del archivo de configuración/etc/haproxy/haproxy.cfg con lo siguiente:

   a. Sustituya LicensePort1 y LicensePort2 por los números de puerto que se van a reenviar al servidor de licencias. Añada o elimine valores separados por comas para acomodar la cantidad de puertos requerida.

   b. LicenseServerHostSustitúyalo por el nombre de host o la dirección IP del servidor de licencias.

```
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. Para habilitar e iniciar el HAProxy servicio, ejecute los siguientes comandos:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Tras completar los pasos, las solicitudes de licencia enviadas a localhost desde el entorno de cola de reenvío deben reenviarse al servidor de licencias especificado.

Paso 3: configuración de la plantilla CloudFormation

Puede usar una CloudFormation plantilla para configurar una granja completa para que utilice sus propias licencias.

1. Modifique la plantilla proporcionada en el siguiente paso para añadir las variables de entorno de licencias necesarias a la sección de variables de BYOLQueueEntorno.
2. Utilice la siguiente CloudFormation plantilla.

```
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. Al implementar la CloudFormation plantilla, proporcione los siguientes parámetros:

   - Actualice el LicenseInstanceID con el ID de instancia de Amazon EC2 de su servidor de licencias o instancia proxy

- LicensePortsActualícelo con una lista de puertos separados por comas para reenviarlos al servidor de licencias o a la instancia proxy (por ejemplo, 2700,2701)
- Agregue las variables de entorno de la licencia sustituyéndolas en la plantilla **`example_LICENSE: 2700@localhost`**

4. Implemente la plantilla para configurar su granja con la capacidad de traer su propia licencia.

Connect las flotas gestionadas por el cliente a un punto final de licencia

El servidor de licencias basado en el uso de AWS Deadline Cloud ofrece licencias bajo demanda para determinados productos de terceros. Con las licencias basadas en el uso, puede pagar por uso. Solo se le cobrará por el tiempo que utilice. Las licencias basadas en el uso proporcionan licencias para que las rendericen tus trabajadores de Deadline Cloud, pero no proporcionan licencias para tus aplicaciones de DCC.

El servidor de licencias basado en el uso de Deadline Cloud se puede usar con cualquier tipo de flota siempre que los trabajadores de Deadline Cloud puedan comunicarse con el servidor de licencias. El servidor de licencias se configura automáticamente en las flotas gestionadas por el servicio. La siguiente configuración solo es necesaria para las flotas gestionadas por el cliente.

Para crear el servidor de licencias, necesita un grupo de seguridad para la VPC de su granja que permita el tráfico de licencias de terceros.

Temas

Paso 1: Cree un grupo de seguridad

Utilice la consola Amazon VPC para crear un grupo de seguridad para la VPC de su granja. Configure el grupo de seguridad para permitir las siguientes reglas de entrada:

- Autodesk Maya y Arnold: 2701 - 2702, TCP,, IPv4 IPv6

- Cinema 4D — 7057, TCP, IPv4 IPv6
- Foundry Nuke — 6101, TCP, IPv4 IPv6
- Red Giant — 7055, TCP, IPV4
- Redshift: 7054, TCP, IPv4 IPv6
- SideFX Houdini, Mantra y Karma — 1715 - 1717, TCP, IPv4 IPv6
- VRay — 30304, TCP, IPV4

La fuente de cada regla de entrada es el grupo de seguridad de los trabajadores de la flota.

Para obtener más información sobre la creación de un grupo de seguridad, consulte Crear un grupo de seguridad en la guía del usuario de Amazon Virtual Private Cloud.

Paso 2: Configure el punto final de la licencia

Un punto final de licencia proporciona acceso a los servidores de licencias para productos de terceros. Las solicitudes de licencia se envían al punto final de la licencia. El punto final las dirige al servidor de licencias correspondiente. El servidor de licencias rastrea los límites de uso y los derechos. Al crear un punto final de licencia en Deadline Cloud, se aprovisiona un punto final de AWS PrivateLink interfaz en su VPC. Estos puntos de conexión se facturan según el precio estándar. AWS PrivateLink Para obtener más información, consulte Precios de AWS PrivateLink.

Con los permisos adecuados, puede crear el punto de conexión de su licencia. Para conocer la política requerida para crear un punto de enlace de licencia, consulte Política para permitir la creación de un punto de enlace de licencia.

Puede crear el punto de conexión de su licencia desde el panel de control de la consola de Deadline Cloud.

1. En el panel de navegación izquierdo, selecciona Puntos de enlace de licencia y, a continuación, selecciona Crear punto de enlace de licencia.
2. En la página Crear punto de enlace de licencia, complete lo siguiente:

    - Seleccione una VPC.
    - Seleccione las subredes que contienen a sus trabajadores de Deadline Cloud. Puede seleccionar hasta 10 subredes.
    - Seleccione el grupo de seguridad que creó en el paso 1. Puede seleccionar hasta 10 grupos de seguridad para situaciones más complicadas.

- (Opcional) Seleccione Añadir nueva etiqueta y añada una o más etiquetas. Puedes añadir hasta 50 etiquetas.

3. Elija Crear punto final de licencia. Cuando se crea el punto de enlace de la licencia, se muestra en la página de puntos de enlace de la licencia.
4. En la sección de productos medidos, elija Añadir productos y, a continuación, seleccione los productos que desee añadir al punto de conexión de su licencia. Elija Añadir.

Para eliminar un producto de un punto final de licencia, en la sección de productos con contador, seleccione el producto y, a continuación, elija Eliminar. En la confirmación, vuelva a seleccionar Eliminar.

Paso 3: Conectar una aplicación de renderizado a un punto final

Una vez configurado el punto final de la licencia, las aplicaciones lo utilizan de la misma manera que lo hacen con un servidor de licencias de terceros. Por lo general, se configura el servidor de licencias para la aplicación estableciendo una variable de entorno u otro ajuste del sistema, como una clave de registro de Microsoft Windows, en un puerto y una dirección del servidor de licencias.

Para obtener el nombre DNS del punto de conexión de la licencia, seleccione el punto final de la licencia en la consola y, a continuación, elija el icono de copia en la sección del nombre de DNS.

Ejemplos de configuraciones

Example— Autodesk Maya y Arnold

> **Note**
>
> Puede utilizar Autodesk Maya y Arnold juntos o por separado. Utilice el puerto 2702 para Autodesk Maya y el puerto 2701 para Arnold.

Para Autodesk Maya, defina la variable de entorno en: `ADSKFLEX_LICENSE_FILE`

```
2702@VPC_Endpoint_DNS_Name
```

Para Arnold, defina la variable `ADSKFLEX_LICENSE_FILE` de entorno en:

```
2701@VPC_Endpoint_DNS_Name
```

Para Autodesk Maya y Arnold, `ADSKFLEX_LICENSE_FILE` defina la variable de entorno en:

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Para Windows los trabajadores, utilice un punto y coma (;) en lugar de dos puntos (:) para separar los puntos finales.

Example— Cinema 4D

Defina la variable de entorno `g_licenseServerRLM` en:

```
VPC_Endpoint_DNS_Name:7057
```

Tras crear la variable de entorno, debería poder renderizar una imagen mediante una línea de comandos similar a esta:

```
"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

Defina la variable `foundry_LICENSE` de entorno en:

```
6101@VPC_Endpoint_DNS_Name
```

Para comprobar que las licencias funcionan correctamente, puedes ejecutar Nuke en una terminal:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Gigante rojo

Defina la variable de entorno `redshift_LICENSE` en:

```
7055@VPC_Endpoint_DNS_Name
```

Tenga en cuenta que Red Giant y Redshift tienen la misma variable de `redshift_LICENSE` entorno. Si desea utilizar ambas aplicaciones, puede configurar la variable de entorno en:

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> En el Windows caso de los trabajadores, utilice un punto y coma (;) en lugar de dos puntos (:) para separar los puntos finales.

Para comprobar que las licencias funcionan correctamente, asegúrese de tener instalados After Effects y Red Giant. A continuación, puede renderizar un proyecto mediante un comando similar a este:

```
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

Defina la variable de entorno `redshift_LICENSE` en:

```
7054@VPC_Endpoint_DNS_Name
```

Tras crear la variable de entorno, debería poder renderizar una imagen mediante una línea de comandos similar a esta:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SideFX, Houdini, Mantra y Karma

Use el siguiente comando:

```
/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;"
```

Para comprobar que las licencias funcionan correctamente, puede renderizar una escena de Houdini mediante este comando:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Defina la variable de entorno `VRAY_AUTH_CLIENT_SETTINGS` en:

```
licset://VPC_Endpoint_DNS_Name:30304
```

Defina la variable de entorno `VRAY_AUTH_CLIENT_FILE_PATH` en:

```
/null
```

Para comprobar que las licencias funcionan correctamente, puede renderizar una imagen VRay mediante un comando similar al siguiente:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Paso 4: Eliminar un punto final de licencia

Al eliminar la flota gestionada por el cliente, no olvide eliminar el punto de conexión de la licencia. Si no elimina el punto de conexión de la licencia, se le seguirán cobrando los costes fijos AWS PrivateLink

Puedes eliminar el punto de conexión de tu licencia desde el panel de control de la consola de Deadline Cloud.

1. En el panel de navegación izquierdo, selecciona License Endpoints.
2. Seleccione el punto final que desee eliminar y elija eliminar y, a continuación, vuelva a seleccionar eliminar para confirmar.

Uso de agentes de IA con Deadline Cloud

Usa agentes de IA para redactar paquetes de trabajos, desarrollar paquetes de conda y solucionar problemas relacionados con los trabajos en Deadline Cloud. En este tema se explica qué son los agentes de IA, los puntos clave para trabajar con ellos de forma eficaz y los recursos para ayudar a los agentes a entender Deadline Cloud.

Un agente de IA es una herramienta de software que utiliza un gran modelo de lenguaje (LLM) para realizar tareas de forma autónoma. Los agentes de IA pueden leer y escribir archivos, ejecutar comandos e iterar soluciones basándose en los comentarios. Algunos ejemplos son las herramientas de línea de comandos, como Kiro, y los asistentes integrados en IDE.

Puntos clave para trabajar con agentes de IA

Los siguientes puntos clave te ayudan a obtener mejores resultados cuando utilizas agentes de IA con Deadline Cloud.

- Proporcione una base sólida: los agentes de IA obtienen mejores resultados cuando tienen acceso a la documentación, las especificaciones y los ejemplos pertinentes. Para proporcionar información básica, puede indicar al agente páginas de documentación específicas, compartir el código de ejemplo existente como referencia, clonar los repositorios de código abierto pertinentes en el espacio de trabajo local y proporcionar documentación para aplicaciones de terceros.
- Especifique los criterios de éxito: defina el resultado esperado y los requisitos técnicos del agente. Por ejemplo, cuando le pida a un agente que desarrolle un paquete de tareas, especifique las entradas, los parámetros y los resultados esperados del trabajo. Si no está seguro de las especificaciones, pídale primero al agente que le proponga opciones y, a continuación, defina los requisitos juntos.
- Habilite un ciclo de retroalimentación: los agentes de IA realizan iteraciones de manera más eficaz cuando pueden probar sus soluciones y recibir comentarios. En lugar de esperar una solución que funcione en el primer intento, deja que el agente ejecute su solución y revise los resultados. Este enfoque funciona bien cuando el agente puede acceder a las actualizaciones de estado, los registros y los errores de validación. Por ejemplo, cuando desarrolle un paquete de trabajos, permita que el agente envíe el trabajo y revise los registros.
- Espere a repetir: incluso con un buen contexto, los agentes pueden desviarse del rumbo o hacer suposiciones que no se ajustan a su entorno. Observe cómo el agente aborda la tarea y guíelo a lo largo del proceso. Añada el contexto que falte si el agente tiene problemas, ayude a encontrar

errores apuntando a archivos de registro específicos, perfeccione los requisitos a medida que los descubra y añada requisitos negativos para indicar de forma explícita lo que el agente debe evitar.

Recursos para el contexto de los agentes

Los siguientes recursos ayudan a los agentes de IA a entender los conceptos de Deadline Cloud y a producir resultados precisos.

- Servidor Deadline-Cloud Model Context Protocol (MCP): para los agentes que admiten el Model Context Protocol, el repositorio Deadline-Cloud contiene el cliente Deadline Cloud, que incluye un servidor MCP para interactuar con los trabajos.
- AWSServidor MCP de documentación: en el caso de los agentes compatibles con el MCP, configure el servidor MCP de AWS documentación para que el agente tenga acceso directo a la AWS documentación, incluidas la Guía del usuario y la Guía del desarrollador de Deadline Cloud.
- Especificación de descripción de trabajo abierta: la especificación de descripción de trabajo abierta GitHub activa define el esquema de las plantillas de trabajo. Consulte este repositorio cuando los agentes necesiten entender la estructura y la sintaxis de las plantillas de trabajo.
- deadline-cloud-samples— El deadline-cloud-samplesrepositorio contiene ejemplos de paquetes de trabajos, recetas y CloudFormation plantillas para aplicaciones y casos de uso comunes.
- Organización aws-deadline: la GitHub organización aws-deadline GitHub contiene complementos de referencia para muchas aplicaciones de terceros que puede utilizar como ejemplos para otras integraciones.

Ejemplo de mensaje: Redactar un paquete de tareas

El siguiente mensaje de ejemplo muestra cómo usar un agente de IA para crear paquetes de trabajos que capaciten a un adaptador LoRa (adaptación de rango bajo) para generar imágenes de IA. El mensaje ilustra los puntos clave discutidos anteriormente: proporciona información básica al señalar los repositorios relevantes, define los criterios de éxito para los resultados de los paquetes de trabajo y describe un circuito de retroalimentación para el desarrollo 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.
```

Supervisión de AWS Deadline Cloud

El monitoreo es una parte importante para mantener la confiabilidad, la disponibilidad y el rendimiento de AWS Deadline Cloud (Deadline Cloud) y sus AWS soluciones. Recopile datos de supervisión de todas las partes de su AWS solución para poder depurar con mayor facilidad una falla multipunto en caso de que se produzca. Antes de comenzar a monitorear Deadline Cloud, debe crear un plan de monitoreo que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la monitorización?
- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?
- ¿Qué herramientas de monitorización va a utilizar?
- ¿Quién se encargará de realizar las tareas de supervisión?
- ¿Quién debería recibir una notificación cuando surjan problemas?

AWSy Deadline Cloud proporcionan herramientas que puede utilizar para supervisar sus recursos y responder a posibles incidentes. Algunas de estas herramientas se encargan de la supervisión por usted, mientras que otras requieren una intervención manual. Debe automatizar las tareas de supervisión en la medida de lo posible.

- Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puedes CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de tus EC2 instancias de Amazon y lanzar automáticamente nuevas instancias cuando sea necesario. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

  Deadline Cloud tiene tres CloudWatch métricas.
- Amazon CloudWatch Logs le permite supervisar, almacenar y acceder a sus archivos de registro desde EC2 instancias de Amazon y otras fuentes. CloudTrail CloudWatch Los registros pueden monitorear la información de los archivos de registro y notificarle cuando se alcanzan ciertos umbrales. También se pueden archivar los datos del registro en un almacenamiento de larga duración. Para obtener más información, consulta la Guía del usuario CloudWatch de Amazon Logs.

- Amazon se EventBridge puede utilizar para automatizar sus AWS servicios y responder automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios de recursos. Los eventos de AWS los servicios se envían casi EventBridge en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Para obtener más información, consulta la Guía EventBridge del usuario de Amazon.
- AWS CloudTrailcaptura las llamadas a la API y los eventos relacionados realizados por su AWS cuenta o en su nombre y entrega los archivos de registro a un bucket de Amazon S3 que especifique. Puede identificar qué usuarios y cuentas llamaronAWS, la dirección IP de origen desde la que se realizaron las llamadas y cuándo se produjeron. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.

Temas

Registro de llamadas a la Deadline Cloud API mediante AWS CloudTrail

Deadline Cloud está integrado con AWS CloudTrailun servicio que proporciona un registro de las acciones realizadas por un usuario, rol o un Servicio de AWS. CloudTrail captura todas las llamadas a la API Deadline Cloud como eventos. Las llamadas capturadas incluyen llamadas desde la Deadline Cloud consola y llamadas en código a las operaciones de la Deadline Cloud API. Con la información recopilada por CloudTrail, puede determinar a qué solicitud se realizó Deadline Cloud, la dirección IP desde la que se realizó la solicitud, cuándo se realizó y detalles adicionales.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del usuario raíz o del usuario.
- Si la solicitud se realizó en nombre de un usuario de IAM Identity Center.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

CloudTrail está activa en tu cuenta Cuenta de AWS al crear la cuenta y automáticamente tienes acceso al historial de CloudTrail eventos. El historial de CloudTrail eventos proporciona un registro visible, consultable, descargable e inmutable de los últimos 90 días de eventos de gestión registrados en un. Región de AWSPara obtener más información, consulte Uso del historial de CloudTrail eventos en la Guía del usuario.AWS CloudTrail La visualización del historial de eventos no conlleva ningún CloudTrail cargo.

Para tener un registro continuo de los eventos de Cuenta de AWS los últimos 90 días, crea un almacén de datos de eventos de senderos o CloudTraillagos.

CloudTrail senderos

Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. Todos los senderos creados con él Consola de administración de AWS son multirregionales. Puede crear un registro de seguimiento de una sola región o multirregionales mediante la AWS CLI. Se recomienda crear un sendero multirregional, ya que puedes capturar toda la actividad de tu Regiones de AWS cuenta. Si crea un registro de seguimiento de una sola región, solo podrá ver los eventos registrados en la Región de AWS del registro de seguimiento. Para obtener más información acerca de los registros de seguimiento, consulte Creación de un registro de seguimiento para su Cuenta de AWS y Creación de un registro de seguimiento para una organización en la Guía del usuario de AWS CloudTrail .

Puede enviar una copia de sus eventos de administración en curso a su bucket de Amazon S3 sin coste alguno CloudTrail mediante la creación de una ruta; sin embargo, hay cargos por almacenamiento en Amazon S3. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios. Para obtener información acerca de los precios de Amazon S3, consulte Precios de Amazon S3.

CloudTrail Almacenes de datos de eventos en Lake

CloudTrail Lake le permite ejecutar consultas basadas en SQL en sus eventos. CloudTrail Lake convierte los eventos existentes en formato JSON basado en filas al formato Apache ORC. ORC es un formato de almacenamiento en columnas optimizado para una recuperación rápida de datos. Los eventos se agregan en almacenes de datos de eventos, que son recopilaciones inmutables de eventos en función de criterios que se seleccionan aplicando selectores de eventos avanzados. Los selectores que se aplican a un almacén de datos de eventos controlan los eventos que perduran y están disponibles para la consulta. Para obtener más información sobre CloudTrail Lake, consulte Cómo trabajar con AWS CloudTrail Lake en la Guía del AWS CloudTrail usuario.

CloudTrail Los almacenes de datos y las consultas sobre eventos de Lake conllevan costes. Cuando crea un almacén de datos de eventos, debe elegir la opción de precios que desee utilizar para él. La opción de precios determina el costo de la incorporación y el almacenamiento de los eventos, así como el período de retención predeterminado y máximo del almacén de datos de eventos. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios.

Deadline Cloud eventos de datos en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos realizadas en o dentro de un recurso (por ejemplo, leer o escribir en un objeto de Amazon S3). Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra los eventos de datos. El historial de CloudTrail eventos no registra los eventos de datos.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre CloudTrail los precios, consulta AWS CloudTrail Precios.

Puede registrar eventos de datos para los tipos de Deadline Cloud recursos mediante la CloudTrail consola o las operaciones de la CloudTrail API. AWS CLI Para obtener más información sobre cómo registrar los eventos de datos, consulte Registro de eventos de datos con la Consola de administración de AWS y Registro de eventos de datos con la AWS Command Line Interface en la Guía del usuario de AWS CloudTrail .

En la siguiente tabla se enumeran los tipos de Deadline Cloud recursos para los que puede registrar eventos de datos. La columna Tipo de evento de datos (consola) muestra el valor que se puede elegir en la lista de tipos de eventos de datos de la CloudTrail consola. La columna de valores resources.type muestra el `resources.type` valor, que se especificaría al configurar los selectores de eventos avanzados mediante o. AWS CLI CloudTrail APIs La CloudTrail columna Datos APIs registrados muestra las llamadas a la API registradas CloudTrail para el tipo de recurso.

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
Deadline Fleet	`AWS::Deadline::Fleet`	• SearchWorkers
Lista de fechas límite	`AWS::Deadline::Fleet`	• SearchJobs

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
Trabajador con fecha límite	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers
Deadline Job	`AWS::Deadline::Job`	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName`, `readOnly` y `resources.ARN` y así registrar solo los eventos que son importantes para usted. Para obtener más información acerca de estos campos, consulte AdvancedFieldSelector en la Referencia de la API de AWS CloudTrail .

Deadline Cloud eventos de gestión en CloudTrail

Los eventos de administración proporcionan información sobre las operaciones de administración que se llevan a cabo en los recursos de su empresa Cuenta de AWS. Se denominan también operaciones del plano de control. De forma predeterminada, CloudTrail registra los eventos de administración.

AWS Deadline Cloud registra las siguientes operaciones del plano de Deadline Cloud control CloudTrail como eventos de administración.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-leer
- assume-fleet-role-for-trabajador
- assume-queue-role-for-leer
- assume-queue-role-for-usuario
- assume-queue-role-for-trabajador
- crear presupuesto
- crear granja
- create-fleet
- create-license-endpoint
- límite de creación
- crear monitor
- crear cola
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association

- create-storage-profile
- create-worker
- eliminar-presupuesto
- delete-farm
- delete-fleet
- delete-license-endpoint
- límite de eliminación
- delete-metered-product
- eliminar-monitor
- eliminar cola
- 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
- obtener presupuesto
- get-farm
- get-feature-map
- get-fleet
- get-license-endpoint
- get-limit
- get-monitor
- get-queue
- get-queue-environment
- get-queue-fleet-association

- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-cola
- list-available-metered-products
- listas de presupuestos
- list-farm-members
- listas de granjas
- list-fleet-members
- listas de flotas
- list-job-members
- list-license-endpoints
- límite de lista
- list-metered-products
- monitores de lista
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- list-queues
- list-storage-profiles
- list-storage-profiles-for-cola
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- actualizar el presupuesto
- update-farm
- actualizar flota

- límite de actualizaciones
- monitor de actualización
- cola de actualizaciones
- update-queue-environment
- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- update-worker

Deadline Cloud ejemplos de eventos

Un evento representa una solicitud única de cualquier fuente e incluye información sobre la operación de API solicitada, la fecha y la hora de la operación, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que los eventos no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra un CloudTrail evento que demuestra la `CreateFarm` operación.

```
{
    "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",
}
```


El ejemplo de JSON muestra la Región de AWS dirección IP y otros «`requestParameters`», como el «`displayName`» y el «`kmsKeyArn`», que pueden ayudarle a identificar el evento.

Para obtener información sobre el contenido de los CloudTrail registros, consulte el contenido de los CloudTrail registros en la Guía del AWS CloudTrail usuario.

Monitorización con CloudWatch

Amazon CloudWatch (CloudWatch) recopila datos sin procesar y los procesa para convertirlos en métricas legibles y casi en tiempo real. Puedes abrir la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/para ver y filtrar las métricas de Deadline Cloud.

Estas estadísticas se guardan durante 15 meses para que pueda acceder a la información histórica y obtener una mejor perspectiva del rendimiento de su aplicación o servicio web. También puede establecer alarmas que vigilen determinados umbrales y enviar notificaciones o realizar acciones cuando se cumplan dichos umbrales. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

Deadline Cloud tiene dos tipos de registros: registros de tareas y registros de trabajadores. Un registro de tareas es cuando se ejecutan registros de ejecución como un script o mientras se ejecuta un DCC. Un registro de tareas puede mostrar eventos como la carga de activos, la representación de teselas o la falta de localización de texturas.

Un registro de trabajo muestra los procesos de los agentes de trabajo. Estos pueden incluir datos como el momento en que el agente de trabajo se pone en marcha, se registra, informa del progreso, carga las configuraciones o completa las tareas.

El espacio de nombres de estos registros es. `/aws/deadline/*`

En el caso de Deadline Cloud, los trabajadores suben estos registros a CloudWatch Logs. De forma predeterminada, los registros nunca caducan. Si un trabajo genera un gran volumen de datos, puede incurrir en costes adicionales. Para obtener más información, consulta los CloudWatch precios de Amazon.

Puede ajustar la política de retención para cada grupo de registros. Una retención más corta elimina los registros antiguos y puede ayudar a reducir los costos de almacenamiento. Para conservar los registros, puede archivarlos en Amazon Simple Storage Service antes de eliminarlos. Para obtener más información, consulte Exportación de datos de registro a Amazon S3 mediante la consola en la guía del CloudWatch usuario de Amazon.

> **Note**
>
> CloudWatch las lecturas de registro están limitadas porAWS. Si tienes pensado incorporar a muchos artistas, te sugerimos que contactes con el servicio de AWS atención al cliente y

solicites un aumento de la `GetLogEvents` cuota CloudWatch. Además, te recomendamos cerrar el portal de registro cuando no estés depurando.

Para obtener más información, consulta las cuotas de CloudWatch registros en la guía del CloudWatch usuario de Amazon.

CloudWatch métricas

Deadline Cloud envía las métricas a Amazon CloudWatch. Puedes usar la Consola de administración de AWSAWS CLI, la o una API para enumerar las métricas a las que envía Deadline Cloud CloudWatch. De forma predeterminada, cada punto de datos cubre el minuto que sigue a la hora de inicio de la actividad. Para obtener información sobre cómo ver las métricas disponibles mediante el Consola de administración de AWS o elAWS CLI, consulta Ver las métricas disponibles en la Guía del CloudWatch usuario de Amazon.

Métricas de flota gestionadas por el cliente

El espacio de `AWS/DeadlineCloud` nombres contiene las siguientes métricas para las flotas gestionadas por los clientes:

Métrica	Description (Descripción)	Unidad
`RecommendedFleetSize`	La cantidad de trabajadores que Deadline Cloud recomienda que utilices para procesar los trabajos. Puedes usar esta métrica para ampliar o reducir el número de trabajadores de tu flota.	Recuento
`UnhealthyWorkerCount`	La cantidad de trabajadores asignados para procesar los trabajos que no están en buen estado.	Recuento

Puede utilizar las siguientes dimensiones para refinar las métricas de flota gestionadas por el cliente:

Dimensión	Description (Descripción)
FarmId	Esta dimensión filtra los datos que solicita a la granja especificada.
FleetId	Esta dimensión filtra los datos que solicita a la flota de trabajadores especificada.

Métricas de licencias

El espacio de `AWS/DeadlineCloud` nombres contiene las siguientes métricas de licencias:

Métrica	Description (Descripción)	Unidad
`LicensesInUse`	El número de sesiones de licencia en uso.	Recuento

Puede usar las siguientes dimensiones para refinar las métricas de licencias:

Dimensión	Description (Descripción)
FleetId	Utilice esta dimensión para filtrar los datos y convertirlos en la flota gestionada por el servicio especificada. En el caso de las flotas gestionadas por los clientes, utilice la dimensión en su lugar. LicenseEndpointId
LicenseEndpointId	Utilice esta dimensión para filtrar los datos hasta el punto final de la licencia especificado.
Producto	Utilice esta dimensión para filtrar los datos hasta el producto medido especificado.

Métricas de límite de recursos

El espacio de `AWS/DeadlineCloud` nombres contiene las siguientes métricas para los límites de recursos:

Métrica	Description (Descripción)	Unidad
`CurrentCount`	La cantidad de recursos modelados según este límite de uso.	Recuento
`MaxCount`	El número máximo de recursos modelados según este límite. Si estableces el `maxCount` valor en -1 mediante la API, Deadline Cloud no emite la `MaxCount` métrica.	Recuento

Puedes usar las siguientes dimensiones para refinar las métricas de límite simultáneas:

Dimensión	Description (Descripción)
FarmId	Esta dimensión filtra los datos que solicita a la granja especificada.
LimitId	Esta dimensión filtra los datos que solicita hasta el límite especificado.

Alarmas recomendadas

Con CloudWatch ellas, puede crear alarmas que vigilen las métricas y le envíen una notificación o realicen otra acción cuando se supere un umbral. Para obtener más información sobre la configuración de CloudWatch alarmas, consulta la Guía del CloudWatch usuario de Amazon.

Te recomendamos configurar alarmas para las siguientes métricas de Deadline Cloud:

LicensesInUse

Dimensiones: FleetId, LicenseEndpointId

Descripción de la alarma: esta alarma detecta cuando las sesiones de licencia activas de una flota gestionada por un servicio o un punto final de licencia se acercan a la cuota de su cuenta. Si esto ocurre, puede aumentar la cuota de la cuenta para las sesiones de licencia. Consulta tus cuotas actuales y solicita aumentos mediante Service Quotas. Para obtener más información, consulte la Guía del usuario de Service Quotas.

Objetivo: Evite los errores en la tramitación de las licencias supervisando el uso antes de que alcance el límite de cuota.

Estadística: Maximum

Umbral recomendado: 90% de la cuota de sesión de licencia

Justificación del umbral: establezca el umbral en un porcentaje de su cuota, de modo que pueda tomar medidas antes de que alcance el límite.

Periodo: 1 minuto

Puntos de datos para la alarma: 1

Períodos de evaluación: 1

Operador de comparación: `GREATER_THAN_THRESHOLD`

Recursos adicionales

- Guía CloudWatch del usuario de Amazon
- Guía del usuario de Service Quotas

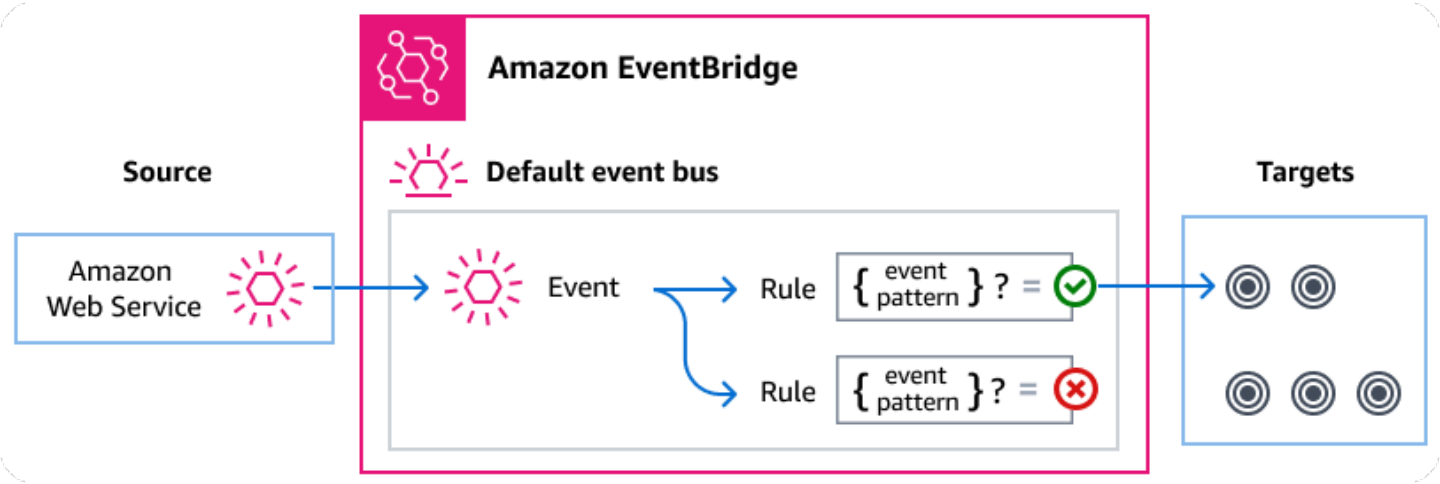
Gestión de eventos de Deadline Cloud mediante Amazon EventBridge

Amazon EventBridge es un servicio sin servidor que utiliza eventos para conectar los componentes de la aplicación, lo que facilita la creación de aplicaciones escalables basadas en eventos. La

arquitectura basada en eventos es un estilo de creación de sistemas de software de acoplamiento flexible que funcionan juntos emitiendo eventos y respondiendo a ellos. Los eventos representan un cambio en un recurso o entorno.

Así es como funciona:

Como ocurre con muchos AWS servicios, Deadline Cloud genera y envía eventos al bus de eventos EventBridge predeterminado. (El bus de eventos predeterminado se aprovisiona automáticamente en todas las AWS cuentas). Un bus de eventos es un enrutador que recibe eventos y los envía a cero o más destinos u objetivos. Las reglas que se especifican al bus de eventos evalúan los eventos a medida que llegan. Cada regla comprueba si un evento coincide con el patrón de evento de la regla. Si el evento coincide, el bus de eventos envía el evento a los destinos especificados.



Temas

- Eventos de Deadline Cloud
- Entregar eventos de Deadline Cloud mediante EventBridge reglas
- Referencia detallada de los eventos de Deadline Cloud

Eventos de Deadline Cloud

Deadline Cloud envía automáticamente los siguientes eventos al bus de EventBridge eventos predeterminado. Los eventos que coinciden con el patrón de eventos de una regla se envían a los destinos especificados de la mejor manera posible. Es posible que los eventos se entreguen fuera de servicio.

Para obtener más información, consulte Eventos de EventBridge en la Guía del usuario de Amazon EventBridge .

Tipo de detalle del evento	Description (Descripción)
Se ha alcanzado el umbral presupuestario	Se envía cuando una cola alcanza un porcentaje del presupuesto asignado.
Cambio de estado del ciclo de vida del trabajo	Se envía cuando se produce un cambio en el estado del ciclo de vida de un trabajo.
Cambio de estado de ejecución de una tarea	Se envía cuando cambia el estado general de las tareas de un trabajo.
Cambio de estado del ciclo de vida escalonado	Se envía cuando se produce un cambio en el estado del ciclo de vida de un paso de una tarea.
Paso: Ejecutar: cambio de estado	Se envía cuando cambia el estado general de las tareas de un paso.
Cambio de estado de ejecución de la tarea	Se envía cuando cambia el estado de una tarea.

Entregar eventos de Deadline Cloud mediante EventBridge reglas

Para que el bus de eventos EventBridge predeterminado envíe los eventos de Deadline Cloud a un destino, debes crear una regla. Cada regla contiene un patrón de eventos que EventBridge coincide con cada evento recibido en el bus de eventos. Si los datos del evento coinciden con el patrón de eventos especificado, EventBridge envía ese evento a los objetivos de la regla.

Para obtener instrucciones detalladas sobre cómo crear reglas de bus de eventos, consulte Creación de reglas que reaccionan a eventos en la Guía del usuario de EventBridge .

Crear patrones de eventos que coincidan con los eventos de Deadline Cloud

Cada patrón de eventos es un objeto JSON que contiene:

- Un atributo `source` que identifica el servicio que envía el evento. En el caso de los eventos de Deadline Cloud, la fuente es `aws.deadline`.
- (Opcional): un atributo `detail-type` que contiene una matriz de los tipos de eventos que deben coincidir.

- (Opcional): un atributo `detail` que contiene cualquier otro dato de evento con el que coincidir.

Por ejemplo, el siguiente patrón de eventos coincide con todos los eventos de cambio de tamaño de flota recomendados `farmId` para Deadline Cloud:

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

Para obtener más información sobre la escritura de los patrones de eventos, consulte Patrones de eventos en la Guía del usuario de EventBridge .

Referencia detallada de los eventos de Deadline Cloud

Todos los eventos de los AWS servicios tienen un conjunto común de campos que contienen metadatos sobre el evento, como el AWS servicio que lo origina, la hora en que se generó el evento, la cuenta y la región en las que tuvo lugar el evento, etc. Para ver las definiciones de estos campos generales, consulte Referencia de estructura de eventos en la Guía del usuario de Amazon EventBridge .

Además, cada evento tiene un campo `detail` que contiene datos específicos de ese evento en particular. La siguiente referencia define los campos de detalle de los distintos eventos de Deadline Cloud.

Al EventBridge utilizarlos para seleccionar y gestionar eventos de Deadline Cloud, es útil tener en cuenta lo siguiente:

- El `source` campo para todos los eventos de Deadline Cloud está configurado en`aws.deadline`.
- El campo `detail-type` especifica el tipo de evento.

  Por ejemplo, `Fleet Size Recommendation Change`.
- El campo `detail` contiene los datos específicos de ese evento en particular.

Para obtener información sobre cómo crear patrones de eventos que permitan que las reglas coincidan con los eventos de Deadline Cloud, consulta los patrones de eventos en la Guía del Amazon EventBridge usuario.

Para obtener más información sobre los eventos y cómo se EventBridge procesan, consulte Amazon EventBridge los eventos en la Guía del Amazon EventBridge usuario.

Temas

Evento alcanzado el umbral presupuestario

Puede utilizar el evento Se ha alcanzado el umbral presupuestario para controlar el porcentaje del presupuesto que se ha utilizado. Deadline Cloud envía los eventos cuando el porcentaje utilizado supera los siguientes umbrales:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

La frecuencia con la que Deadline Cloud envía eventos relacionados con el umbral de presupuesto alcanzado aumenta a medida que el presupuesto se acerca a su límite. Esta frecuencia le permite supervisar de cerca un presupuesto a medida que se acerca a su límite y tomar medidas para evitar gastos excesivos. También puede establecer sus propios umbrales presupuestarios. Deadline Cloud envía un evento cuando el uso supera tus umbrales personalizados.

Si cambias el importe de un presupuesto, la próxima vez que Deadline Cloud envíe un evento sobre el límite presupuestario alcanzado, se basará en el porcentaje actual del presupuesto que se haya utilizado. Por ejemplo, si añades 50$ a un presupuesto de 100$ que ha alcanzado su límite, el siguiente evento denominado «Se ha alcanzado el umbral presupuestario» indicará que el presupuesto es del 75 por ciento.

A continuación, se muestran los campos de detalle del evento `Budget Threshold Reached`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.

```
{
    "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 el tipo de evento.

Para este evento, este valor es `Budget Threshold Reached`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`budgetId`

El identificador del presupuesto que ha alcanzado un umbral.

`thresholdInPercent`

El porcentaje del presupuesto que se ha utilizado.

Evento de cambio de recomendación de tamaño de flota

Cuando configuras tu flota para usar el escalado automático basado en eventos, Deadline Cloud envía eventos que puedes usar para administrar tus flotas. Cada uno de estos eventos contiene información sobre el tamaño actual y el tamaño solicitado de una flota. Para ver un ejemplo del uso de un EventBridge evento y un ejemplo de función Lambda para gestionar el evento, consulte Escalar automáticamente su flota de Amazon EC2 con la función de recomendación de escalado de Deadline Cloud.

El evento de cambio de recomendación de tamaño de la flota se envía cuando ocurre lo siguiente:

- Cuando el tamaño de flota recomendado cambia y oldFleetSize es diferente de newFleetSize.
- Cuando el servicio detecta que el tamaño real de la flota no coincide con el tamaño de flota recomendado. Puede obtener el tamaño real de la flota a partir del WorkerCount en la respuesta a la GetFleet operación. Esto puede ocurrir cuando una instancia activa de Amazon EC2 no se registra como trabajadora de Deadline Cloud.

A continuación, se muestran los campos de detalle del evento `Fleet Size Recommendation Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.

```
{
    "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 el tipo de evento.

Para este evento, este valor es `Fleet Size Recommendation Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`fleetId`

El identificador de la flota que necesita un cambio de tamaño.

`oldFleetSize`

El tamaño actual de la flota.

`newFleetSize`

El nuevo tamaño recomendado para la flota.

Evento de cambio de estado del ciclo de vida laboral

Al crear o actualizar un trabajo, Deadline Cloud establece el estado del ciclo de vida para mostrar el estado de la acción iniciada por el usuario más recientemente.

Se envía un evento de cambio de estado del ciclo de vida del trabajo para cualquier cambio de estado del ciclo de vida, incluso cuando se crea el trabajo.

A continuación, se muestran los campos de detalle del evento `Job Lifecycle Status Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.

```
{
    "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 el tipo de evento.

Para este evento, este valor es `Job Lifecycle Status Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`queueId`

El identificador de la cola que contiene el trabajo.

`jobId`

El identificador del trabajo.

`previousLifecycleStatus`

El estado del ciclo de vida en el que se va a dejar el trabajo. Este campo no se incluye cuando envías un trabajo por primera vez.

`lifecycleStatus`

El estado del ciclo de vida al que está ingresando el trabajo.

Evento Job Run Status Change

Un trabajo se compone de muchas tareas. Cada tarea tiene un estado. El estado de todas las tareas se combina para proporcionar un estado general de un trabajo. Para obtener más información, consulta los estados de los trabajos en Deadline Cloud en la Guía del usuario de AWS Deadline Cloud.

Se envía un evento de cambio de estado de ejecución de un trabajo cuando:

- El `taskRunStatus` campo combinado cambia.
- El trabajo se vuelve a poner en cola, a menos que esté en el estado LISTO.

NO se envía un evento de cambio de estado de ejecución de una tarea cuando:

- El trabajo se crea por primera vez. Para supervisar la creación de puestos de trabajo, supervise los eventos de cambio de estado del ciclo de vida del trabajo para detectar cambios.

- El `taskRunStatusCounts` campo del trabajo cambia, pero el estado de ejecución de la tarea combinada no cambia.

A continuación, se muestran los campos de detalle del evento `Job Run Status Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.

```
{
    "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 el tipo de evento.

Para este evento, este valor es `Job Run Status Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`queueId`

El identificador de la cola que contiene el trabajo.

`jobId`

El identificador del trabajo.

`previousTaskRunStatus`

La tarea ejecutada indica que el trabajo se va a finalizar.

`taskRunStatus`

La ejecución de la tarea indica que el trabajo está ingresando.

`taskRunStatusCounts`

El número de tareas del trabajo en cada estado.

Evento Step Lifecycle Status Change

Al crear o actualizar un evento, Deadline Cloud establece el estado del ciclo de vida del trabajo para describir el estado de la acción iniciada por el usuario más recientemente.

Se envía un evento de cambio de estado del ciclo de vida escalonado cuando:

- Se inicia una actualización escalonada (UPDATE_IN_PROGRESS).
- La actualización de un paso se completó correctamente (UPDATE_SUCEDED).
- Error en la actualización de un paso (UPDATE_FAILED).

No se envía un evento cuando se crea el paso por primera vez. Para supervisar la creación de pasos, supervise los eventos de cambio de estado del ciclo de vida del trabajo para ver si hay cambios.

A continuación, se muestran los campos de detalle del evento `Step Lifecycle Status Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.

```
{
    "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 el tipo de evento.

Para este evento, este valor es `Step Lifecycle Status Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`queueId`

El identificador de la cola que contiene el trabajo.

`jobId`

El identificador del trabajo.

`stepId`

El identificador del paso de trabajo actual.

`previousLifecycleStatus`

El estado del ciclo de vida del que sale el paso.

`lifecycleStatus`

El estado del ciclo de vida al que ingresa el paso.

Evento Step Run Status Change

Cada paso de un trabajo se compone de muchas tareas. Cada tarea tiene un estado. Los estados de las tareas se combinan para proporcionar un estado general de los pasos y los trabajos.

Se envía un evento de cambio de estado de ejecución escalonada cuando:

- La combinación `taskRunStatus` cambia.

- El paso se vuelve a poner en cola, a menos que esté en el estado LISTO.

No se envía un evento cuando:

- El paso se crea primero. Para supervisar la creación de pasos, supervise los eventos de cambio de estado del ciclo de vida del trabajo para ver si hay cambios.
- El paso `taskRunStatusCounts` cambia, pero el estado de ejecución de la tarea de los pasos combinados no cambia.

A continuación, se muestran los campos de detalle del evento `Step Run Status Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.


```
{
    "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 el tipo de evento.

Para este evento, este valor es `Step Run Status Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`queueId`

El identificador de la cola que contiene el trabajo.

`jobId`

El identificador del trabajo.

`stepId`

El identificador del paso de trabajo actual.

`previousTaskRunStatus`

El estado de ejecución por el que se va el paso.

`taskRunStatus`

El estado de ejecución al que está ingresando el paso.

`taskRunStatusCounts`

El número de tareas del paso en cada estado.

Evento de cambio de estado de ejecución de la tarea

El `runStatus` campo se actualiza a medida que se ejecuta la tarea. Se envía un evento cuando:

- El estado de ejecución de la tarea cambia.
- La tarea se vuelve a poner en cola, a menos que esté en el estado LISTA.

No se envía un evento cuando:

- La tarea se crea por primera vez. Para supervisar la creación de tareas, supervise los eventos de cambio de estado del ciclo de vida del trabajo para ver si hay cambios.

A continuación, se muestran los campos de detalle del evento `Task Run Status Change`.

Los `detail-type` campos `source` y se incluyen a continuación porque contienen valores específicos para los eventos de Deadline Cloud. Para ver las definiciones de los demás campos de metadatos que se incluyen en todos los eventos, consulte la referencia a la estructura de eventos en la Guía del Amazon EventBridge usuario.


```
{
    "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 el tipo de evento.

Para este evento, este valor es `Fleet Size Recommendation Change`.

`source`

Identifica el servicio que generó el evento. Para los eventos de Deadline Cloud, este valor es`aws.deadline`.

`detail`

Un objeto JSON que contiene información sobre el evento. El servicio que genera el evento determina el contenido de este campo.

Para este evento, estos datos incluyen lo siguiente:

`farmId`

El identificador de la granja que contiene el trabajo.

`queueId`

El identificador de la cola que contiene el trabajo.

`jobId`

El identificador del trabajo.

stepId

El identificador del paso de trabajo actual.

`taskId`

El identificador de la tarea en ejecución.

`previousRunStatus`

El estado de ejecución por el que sale la tarea.

`runStatus`

El estado de ejecución al que está ingresando la tarea.

Consulta de datos agregados de estadísticas de sesión mediante el AWS CLI

Para realizar un seguimiento de los costes, analizar el uso de los recursos o identificar qué usuarios consumen más recursos, puedes utilizar AWS Command Line Interface (AWS CLI) para consultar las estadísticas de sesión agregadas de tus granjas de AWS Deadline Cloud (Deadline Cloud). La API de estadísticas de sesión proporciona datos sobre los costes, el tiempo de ejecución y el uso que puedes agrupar por diversas dimensiones, como cola, flota, tipo de instancia o usuario.

La consulta de las estadísticas de la sesión es un proceso asíncrono. En primer lugar, se inicia una solicitud de agregación y, a continuación, se recuperan los resultados mediante el ID de agregación.

Iniciar una solicitud de agregación

Para iniciar una solicitud de agregación, ejecute el `start-sessions-statistics-aggregation` comando. En el siguiente ejemplo, se agrupan las estadísticas por ID de usuario para una cola específica. Sustituya el *`placeholder`* texto por su información.

```
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
```

Puede agrupar las estadísticas por otras dimensiones`QUEUE_ID`, como `FLEET_ID``JOB_ID`,`INSTANCE_TYPE`, o`LICENSE_PRODUCT`. Para obtener más información sobre todos los parámetros disponibles, consulte start-sessions-statistics-aggregationla Referencia de AWS CLI comandos.

La respuesta contiene un identificador de agregación:

```
{
```

```
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Recuperación de resultados

Ejecute el `get-sessions-statistics-aggregation` comando con el ID de agregación para recuperar los resultados. Sustituya el *`placeholder`* texto por su información.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

En el siguiente ejemplo, se muestra una respuesta al agrupar las estadísticas por ID de usuario. El `userId` campo contiene un UUID que debe asignar a un nombre de usuario para identificar al usuario:

```
{
    "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"
}
```

Para encontrar el nombre de usuario asociado a `auserId`, consultethe section called “Recuperación de metadatos de usuario mediante UserID”.

Para obtener más información sobre la API, consulta la referencia de la API de Deadline Cloud.

Temas

- the section called “Recuperación de metadatos de usuario mediante UserID”

Recuperación de metadatos y atributos de usuario mediante el ID de usuario de un almacén de identidades

> **Note**
>
> Este procedimiento también se aplica al `createdBy` campo devuelto por la SearchJobsAPI, que utiliza el mismo formato de ID de usuario.

El `userId` campo de las estadísticas de sesión contiene uno de los siguientes valores:

- Un AWS Identity and Access Management ARN de rol (IAM), por ejemplo:. `arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`
- Un ID de usuario (UUID) del IAM Identity Center, por ejemplo:. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Para el rol de IAM ARNs, el nombre de usuario está visible en el propio ARN. Para el usuario del IAM Identity Center IDs, puede buscar el nombre de usuario mediante la API Identity Store de IAM Identity Center.

Para identificar el nombre de usuario asociado a un ID de usuario del IAM Identity Center, utilice el siguiente procedimiento. Antes de empezar, obtenga el ID del almacén de identidades en la configuración del Centro de identidades de IAM. Para obtener más información, consulte the section called “Cómo encontrar tu ID de Identity Store”.

Para mapear un ID de usuario

1. Ejecute el siguiente comando y *`IdentityStoreId`* sustitúyalo por su ID de Identity Store y *`userUUID`* por la respuesta `userId` de las estadísticas de la sesión:

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Revisa la respuesta, que incluye el nombre de usuario:

```
{
    "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"
}
```

Cómo encontrar tu ID de Identity Store

Para asignar un usuario IDs a un nombre de usuario, necesita el ID de Identity Store. Puede encontrar el ID del almacén de identidades mediante la consola del IAM Identity Center o el. AWS CLI

Consola

Para encontrar su ID de almacén de identidades mediante la consola, utilice el siguiente procedimiento.

1. Inicie sesión en la consola AWS de administración y abra la consola de IAM Identity Center.
2. En el panel de navegación, seleccione Configuración.
3. Copie el valor del ID del almacén de identidades de IAM Identity Center. El formato es `d-xxxxxxxxxx`.

AWS CLI

Ejecute el siguiente comando y *`region-name`* sustitúyalo por la región en la que está configurada la instancia del IAM Identity Center:

```
aws sso-admin list-instances --region region-name
```

La respuesta incluye lo siguiente: `IdentityStoreId`

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
    ]
}
```

Verificar el mapeo de usuarios

Tras asignar un ID de usuario a un nombre de usuario, puede comprobar en la consola del IAM Identity Center que el ID de usuario coincide con el usuario esperado. Para verificar la asignación de usuarios, utilice el siguiente procedimiento.

1. Inicie sesión en la consola AWS de administración y abra la consola de IAM Identity Center.
2. En el panel de navegación, seleccione Usuarios.
3. Elija el nombre de usuario de la AWS CLI respuesta.
4. En la sección Información general, compruebe que el seudónimo coincide con el `userId` de las estadísticas de su sesión.

Recursos adicionales

- Guía del usuario de IAM Identity Center
- Referencia de la API Identity Store de IAM Identity Center

Seguridad en Deadline Cloud

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. El se refiere a estos conceptos como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que se ejecuta Servicios de AWS en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Third-party los auditores prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS cumplimiento de los programas de . Para obtener más información sobre los programas de cumplimiento aplicables AWS Deadline Cloud, consulte Servicios de AWS Alcance by Compliance Servicios de AWS .
- Seguridad en la nube: su responsabilidad viene determinada por lo Servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Deadline Cloud. Los siguientes temas muestran cómo configurarlo Deadline Cloud para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros Servicios de AWS que le ayuden a supervisar y proteger sus Deadline Cloud recursos.

Temas

- Mejores prácticas de seguridad para Deadline Cloud

Protección de datos en Deadline Cloud

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en AWS Deadline Cloud. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre privacidad de datos y los . Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Para fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales

como el campo Nombre. Esto incluye cuando trabaja Deadline Cloud o Servicios de AWS utiliza la consola, la API o los SDK. AWS CLI AWS Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Los datos introducidos en los campos de nombres de las plantillas de Deadline Cloud trabajo también pueden incluirse en los registros de facturación o diagnóstico y no deben contener información confidencial o delicada.

Temas

Cifrado en reposo

AWS Deadline Cloud protege los datos confidenciales cifrándolos en reposo mediante claves de cifrado almacenadas en AWS Key Management Service ()AWS KMS. El cifrado en reposo está disponible en todos los Regiones de AWS lugares donde Deadline Cloud esté disponible.

El cifrado de datos significa que un usuario o una aplicación no pueden leer los datos confidenciales guardados en los discos sin una clave válida. Solo una parte con una clave gestionada válida puede descifrar los datos.

Deadline Cloud elimina los volúmenes de Amazon Elastic Block Store cuando finalizan las instancias de trabajadores de flota gestionados por el servicio.

Para obtener información sobre cómo se Deadline Cloud utiliza el cifrado AWS KMS de datos en reposo, consulte. Administración de claves

Cifrado en tránsito

Para los datos en tránsito, AWS Deadline Cloud utiliza Transport Layer Security (TLS) 1.2 o 1.3 para cifrar los datos enviados entre el servicio y los trabajadores. Exigimos TLS 1.2 y recomendamos

TLS 1.3. Además, si utiliza una nube privada virtual (VPC), puede utilizarla AWS PrivateLink para establecer una conexión privada entre su VPC y. Deadline Cloud

Administración de claves

Al crear una granja nueva, puede elegir una de las siguientes claves para cifrar los datos de la granja:

- AWS clave KMS propia: tipo de cifrado predeterminado si no especificas una clave al crear la granja. La clave KMS es propiedad de AWS Deadline Cloud. No puede ver, administrar ni usar las claves AWS propias. Sin embargo, no es necesario que realices ninguna acción para proteger las claves que cifran tus datos. Para obtener más información, consulta las claves AWS propias en la guía para AWS Key Management Service desarrolladores.
- Clave de KMS gestionada por el cliente: al crear una granja, se especifica una clave gestionada por el cliente. Todo el contenido de la granja se cifra con la clave KMS. La clave se almacena en su cuenta y es usted quien la crea, es de su propiedad y la administra, por lo que se aplican AWS KMS cargos. Usted controla plenamente la clave KMS. Puede realizar tareas como las siguientes:
  - Establecer y mantener políticas clave
  - Establecer y mantener concesiones y políticas de IAM
  - Habilitar y deshabilitar políticas de claves
  - Adición de etiquetas de
  - Crear alias de clave

  No se puede rotar manualmente una clave propiedad del cliente utilizada en una Deadline Cloud granja. Se admite la rotación automática de la llave.

  Para obtener más información, consulte las claves propiedad del cliente en la Guía para AWS Key Management Service desarrolladores.

  Para crear una clave gestionada por el cliente, sigue los pasos para crear claves simétricas gestionadas por el cliente que se indican en la Guía para AWS Key Management Service desarrolladores.

Cómo Deadline Cloud uses AWS KMS subsidios

Deadline Cloud requiere una concesión para utilizar la clave gestionada por el cliente. Cuando crea una granja cifrada con una clave gestionada por el cliente, Deadline Cloud crea una concesión en su

nombre enviando una `CreateGrant` solicitud AWS KMS para obtener acceso a la clave KMS que especificó.

Deadline Cloud utiliza varias concesiones. Cada subvención es utilizada por una parte diferente Deadline Cloud que necesita cifrar o descifrar sus datos. Deadline Cloud también utiliza subvenciones para permitir el acceso a otros AWS servicios utilizados para almacenar datos en su nombre, como Amazon Simple Storage Service, Amazon Elastic Block Store o OpenSearch.

Las subvenciones que permiten Deadline Cloud gestionar las máquinas de una flota gestionada por un servicio incluyen un número de Deadline Cloud cuenta y una función en el centro del servicio, en `GranteePrincipal` lugar de un director de servicio. Si bien no es habitual, esto es necesario para cifrar los volúmenes de Amazon EBS para los trabajadores de las flotas gestionadas por el servicio mediante la clave de KMS gestionada por el cliente especificada para la granja.

Política de claves administradas por el cliente

Las políticas de clave controlan el acceso a la clave administrada por el cliente. Cada clave debe tener exactamente una política de claves que contenga instrucciones que determinen quién puede usar la clave y cómo puede usarla. Al crear la clave gestionada por el cliente, puede especificar una política clave. Para obtener más información, consulte Administración del acceso a las claves en la Guía para desarrolladores de AWS Key Management Service .

Política de IAM mínima para CreateFarm

Para usar la clave administrada por el cliente para crear granjas mediante la consola o la operación de `CreateFarm` API, deben estar permitidas las siguientes operaciones de AWS KMS API:

- `kms:CreateGrant`: agrega una concesión a una clave administrada por el cliente. Concede acceso a la consola a una AWS KMS clave específica. Para obtener más información, consulta Cómo usar las subvenciones en la guía para AWS Key Management Service desarrolladores.
- `kms:Decrypt`— Permite Deadline Cloud descifrar los datos de la granja.
- `kms:DescribeKey`— Proporciona los detalles clave gestionados por el cliente Deadline Cloud para permitir su validación.
- `kms:GenerateDataKey`— Permite cifrar Deadline Cloud los datos mediante una clave de datos única.

La siguiente declaración de política otorga los permisos necesarios para la `CreateFarm` operación.

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"
                }
            }
        }
    ]
}
```

Política de IAM mínima para operaciones de solo lectura

Utilizar la clave gestionada por el cliente para Deadline Cloud operaciones de solo lectura, como obtener información sobre granjas, colas y flotas. Se deben permitir las siguientes operaciones AWS KMS de API:

- `kms:Decrypt`— Permite Deadline Cloud descifrar los datos de la granja.
- `kms:DescribeKey`— Proporciona los detalles clave gestionados por el cliente Deadline Cloud para permitir su validación.

La siguiente declaración de política otorga los permisos necesarios para las operaciones de solo lectura.

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"
                }
            }
        }
    ]
}
```

Política de IAM mínima para las operaciones de lectura-escritura

Utilizar la clave gestionada por el cliente para Deadline Cloud operaciones de lectura-escritura, como la creación y actualización de granjas, colas y flotas. Deben permitirse las siguientes operaciones AWS KMS de API:

- `kms:Decrypt`— Permite Deadline Cloud descifrar los datos de la granja.
- `kms:DescribeKey`— Proporciona los detalles clave gestionados por el cliente Deadline Cloud para permitir su validación.
- `kms:GenerateDataKey`— Permite cifrar Deadline Cloud los datos mediante una clave de datos única.

La siguiente declaración de política otorga los permisos necesarios para la `CreateFarm` operación.

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"
                }
            }
        }
    ]
}
```

Supervisión de sus claves de cifrado

Cuando utilizas una clave gestionada por el AWS KMS cliente en tus Deadline Cloud granjas, puedes utilizar AWS CloudTrailAmazon CloudWatch Logs para realizar un seguimiento de las solicitudes que se Deadline Cloud envían a AWS KMS.

CloudTrail evento de concesión de subvenciones

El siguiente CloudTrail evento de ejemplo se produce cuando se crean las concesiones, normalmente cuando se llama a la `CreateFleet` operación `CreateFarmCreateMonitor`, o.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


CloudTrail evento de descifrado

El siguiente CloudTrail evento de ejemplo se produce al descifrar valores mediante la clave KMS administrada por el 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-abcdef1234567890098765432 1fedcba",
            "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 de cifrado

El siguiente CloudTrail evento de ejemplo se produce al cifrar valores mediante la clave KMS administrada por el 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-abcdef1234567890987654321fedcba",
            "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"
}
```

Eliminar una clave KMS administrada por el cliente

Eliminar una clave de KMS gestionada por el cliente en AWS Key Management Service (AWS KMS) es destructivo y potencialmente peligroso. Elimina el material de claves y todos los metadatos asociados con la clave. Esta acción es irreversible. Una vez que se elimina una clave KMS administrada por el cliente, ya no puede descifrar los datos que se habían cifrado con ella. Al eliminar la clave, los datos se vuelven irrecuperables.

Por eso, los clientes AWS KMS tienen un período de espera de hasta 30 días antes de eliminar la clave KMS. El periodo de espera predeterminado es de 30 días.

Acerca del período de espera

Dado que eliminar una clave de KMS gestionada por el cliente es destructivo y potencialmente peligroso, te pedimos que establezcas un período de espera de 7 a 30 días. El periodo de espera predeterminado es de 30 días.

Sin embargo, el período de espera real puede ser hasta 24 horas más largo que el período que programaste. Para obtener la fecha y la hora reales en las que se eliminará la clave, utilice la DescribeKeyoperación. O en la consola AWS KMS, en la página de detalles para la clave, en la sección Configuración general, consulte la eliminación programada. Fíjese en la zona horaria.

Durante el periodo de espera, el estado de la clave administrada por el cliente y el estado de la clave es Eliminación pendiente.

- Una clave KMS administrada por el cliente que está pendiente de eliminación no puede utilizarse en ninguna operación criptográfica.
- AWS KMS no rota las claves de respaldo de las claves de KMS administradas por el cliente que están pendientes de ser eliminadas.

Para obtener más información sobre cómo eliminar una clave KMS administrada por el cliente, consulte Eliminar las claves maestras del cliente en la Guía para AWS Key Management Service desarrolladores.

Inter-network privacidad del tráfico

AWS Deadline Cloud es compatible con Amazon Virtual Private Cloud (Amazon VPC) para proteger las conexiones. Amazon VPC ofrece características que puede utilizar para aumentar y monitorear la seguridad de su nube privada virtual (VPC):

Puede configurar una flota gestionada por el cliente (CMF) con instancias de Amazon Elastic Compute Cloud (Amazon EC2) que se ejecuten dentro de una VPC. Al implementar los puntos de enlace de Amazon VPC para su uso AWS PrivateLink, el tráfico entre los trabajadores de su CMF y el Deadline Cloud punto final permanece dentro de su VPC. Además, puede configurar su VPC para restringir el acceso a Internet a sus instancias.

En las flotas gestionadas por servicios, no se puede acceder a los trabajadores desde Internet, pero sí tienen acceso a Internet y se conectan al servicio a través de Deadline Cloud Internet. Cada flota gestionada por el servicio funciona en su propia red aislada, y las instancias de trabajo permanecen dedicadas a los clientes individuales.

cancelación de la suscripción

AWS Deadline Cloud recopila cierta información operativa para ayudarnos a desarrollarnos y mejorar. Deadline Cloud Los datos recopilados incluyen datos como su ID de AWS cuenta y su ID de usuario, para que podamos identificarlo correctamente si tiene algún problema con ellos Deadline Cloud. También recopilamos información Deadline Cloud específica, como los identificadores de recursos (un FarmID o un QueueID, cuando proceda), el nombre del producto (por ejemplo,, JobAttachments WorkerAgent, etc.) y la versión del producto.

Puede optar por excluirse de esta recopilación de datos mediante la configuración de la aplicación. Cada ordenador con el que interactúe Deadline Cloud, tanto las estaciones de trabajo del cliente como los trabajadores de la flota, debe excluirse por separado.

Deadline Cloud monitor - sobremesa

Deadline Cloud monitor: desktop recopila información operativa, como cuándo se producen bloqueos y cuándo se abre la aplicación, para ayudarnos a saber cuándo tiene problemas con la aplicación. Para excluirse de la recopilación de esta información operativa, vaya a la página de configuración y desactive la opción Activar la recopilación de datos para medir el rendimiento de Deadline Cloud Monitor.

Tras excluirse, el monitor de escritorio ya no envía los datos operativos. Todos los datos recopilados anteriormente se conservan y pueden seguir utilizándose para mejorar el servicio. Para obtener más información, consulte Preguntas frecuentes sobre la privacidad de datos de .

AWS Deadline Cloud CLI y herramientas

La AWS Deadline Cloud CLI, los remitentes y el agente laboral recopilan información operativa, como cuándo se producen accidentes y cuándo se envían los trabajos, para ayudarnos a saber cuándo tiene problemas con estas solicitudes. Para excluirse de la recopilación de esta información operativa, utilice cualquiera de los siguientes métodos:

- En la terminal, ingresa**`deadline config set telemetry.opt_out true`**.

  Esto excluirá la CLI, los remitentes y el agente de trabajo cuando se ejecute como el usuario actual.
- Al instalar el agente de Deadline Cloud trabajo, añada el argumento de la línea de **`--telemetry-opt-out`** comandos. Por ejemplo, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.
- Antes de ejecutar el agente de trabajo, la CLI o el remitente, establezca una variable de entorno: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Tras excluirse, las Deadline Cloud herramientas dejarán de enviar los datos operativos. Todos los datos recopilados anteriormente se conservan y pueden seguir utilizándose para mejorar el servicio. Para obtener más información, consulte Preguntas frecuentes sobre la privacidad de datos de .

Identity and Access Management en Deadline Cloud

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos de Deadline Cloud. La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Resolución de problemas AWS Identidad y acceso a Deadline Cloud).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Deadline Cloud con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Identity-based ejemplos de políticas para Deadline Cloud).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las

credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Al crear una Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz, que tiene acceso completo a todos los Servicios de AWS recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, del proveedor de identidades web o al Directory Service que se accede Servicios de AWS mediante credenciales de una fuente de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, se recomienda AWS IAM Identity Center. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

AWS Para controlar el acceso, puede crear políticas y adjuntarlas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Identity-based políticas

Identity-based las políticas son documentos de política de permisos de JSON que se adjuntan a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Identity-based las políticas pueden ser políticas integradas (integradas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Resource-based políticas

Resource-based las políticas son documentos de políticas de JSON que se adjuntan a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Resource-based las políticas son políticas en línea que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que conceden los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCP): definen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations .
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Deadline Cloud con IAM

Antes de utilizar IAM para gestionar el acceso a Deadline Cloud, infórmese sobre las funciones de IAM disponibles para su uso con Deadline Cloud.

Funciones de IAM que puede utilizar con AWS Deadline Cloud

Característica de IAM	Soporte de Deadline Cloud
Identity-based políticas	Sí
Resource-based políticas	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Sesiones de acceso directo (FAS)	Sí
Roles de servicio	Sí
Service-linked roles	No

Para obtener una visión general de cómo Servicios de AWS funcionan Deadline Cloud y otros con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Identity-based políticas de Deadline Cloud

Compatibilidad con las políticas basadas en identidad: sí

Identity-based las políticas son documentos de política de permisos de JSON que puedes adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de la política de JSON de IAM en la Guía del usuario de IAM.

Identity-based ejemplos de políticas para Deadline Cloud

Para ver ejemplos de políticas basadas en la identidad de Deadline Cloud, consulte. Identity-based ejemplos de políticas para Deadline Cloud

Resource-based políticas dentro de Deadline Cloud

Admite políticas basadas en recursos: no

Resource-based las políticas son documentos de políticas de JSON que se adjuntan a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política basada en recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones políticas para Deadline Cloud

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de las acciones de Deadline Cloud, consulte las acciones definidas por AWS Deadline Cloud en la Referencia de autorización de servicios.

Las acciones políticas en Deadline Cloud usan el siguiente prefijo antes de la acción:

```
deadline
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Para ver ejemplos de políticas basadas en la identidad de Deadline Cloud, consulte. Identity-based ejemplos de políticas para Deadline Cloud

Recursos de políticas para Deadline Cloud

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Deadline Cloud y sus ARN, consulte los recursos definidos por AWS Deadline Cloud en la referencia de autorización de servicios. Para saber con qué acciones puede especificar el ARN de cada recurso, consulte Acciones definidas por AWS Deadline Cloud.

Para ver ejemplos de políticas basadas en la identidad de Deadline Cloud, consulte. Identity-based ejemplos de políticas para Deadline Cloud

Claves de condición de la política de Deadline Cloud

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de las claves de condición de Deadline Cloud, consulte las claves de condición de AWS Deadline Cloud en la Referencia de autorización de servicio. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS Deadline Cloud.

Para ver ejemplos de políticas basadas en la identidad de Deadline Cloud, consulte. Identity-based ejemplos de políticas para Deadline Cloud

ACL en Deadline Cloud

Compatibilidad con ACL: no

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Deadline Cloud

Admite ABAC (etiquetas en las políticas): sí

Attribute-based el control de acceso (ABAC) es una estrategia de autorización que define los permisos en función de unos atributos denominados etiquetas. Puede adjuntar etiquetas a las entidades y AWS los recursos de IAM y, a continuación, diseñar políticas de ABAC para permitir las operaciones cuando la etiqueta del director coincida con la etiqueta del recurso.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Uso de credenciales temporales con Deadline Cloud

Compatibilidad con credenciales temporales: sí

Las credenciales temporales proporcionan acceso a AWS los recursos a corto plazo y se crean automáticamente cuando utilizas una federación o cambias de rol. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Sesiones de acceso directo para Deadline Cloud

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos de la persona principal que realiza la llamada y la que solicita Servicio de AWS para realizar solicitudes a los servicios intermedios. Servicio de AWS Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Sesiones de acceso directo.

Funciones de servicio para Deadline Cloud

Compatible con roles de servicio: sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Deadline Cloud. Edita las funciones de servicio solo cuando Deadline Cloud te dé instrucciones para hacerlo.

Service-linked roles para Deadline Cloud

Compatibilidad con roles vinculados al servicio: no

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir la función de realizar una acción en su nombre. Service-linked las funciones aparecen en su nombre Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya una `Yes` en la columna de Service-linked funciones. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Identity-based ejemplos de políticas para Deadline Cloud

De forma predeterminada, los usuarios y los roles no tienen permiso para crear o modificar los recursos de Deadline Cloud. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Deadline Cloud, incluido el formato de los ARN de cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición de AWS Deadline Cloud en la Referencia de autorización de servicios.

Temas

- Prácticas recomendadas sobre las políticas
- Uso de la consola de Deadline Cloud

- Política de acceso a la consola
- Política para enviar trabajos a una cola
- Política que permite la creación de un punto final de licencia
- Política que permite monitorear una cola de granja específica

Prácticas recomendadas sobre las políticas

Identity-based las políticas determinan si alguien puede crear, acceder o eliminar los recursos de Deadline Cloud de su cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para

más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.

- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Deadline Cloud

Para acceder a la consola de AWS Deadline Cloud, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Deadline Cloud que tiene en su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realicen llamadas a la API AWS CLI o a la AWS API. En su lugar, permita el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando la consola de Deadline Cloud, adjunta también la nube de Deadline *ConsoleAccess* o la política *ReadOnly* AWS gestionada a las entidades. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Política de acceso a la consola

Para conceder acceso a todas las funciones de la consola de Deadline Cloud, adjunta esta política de identidad a un usuario o rol al que quieras tener acceso 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": ["*"]
      }]
}
```

Política para enviar trabajos a una cola

En este ejemplo, se crea una política exhaustiva que concede permiso para enviar trabajos a una cola específica de una granja específica.

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/*"
        }
    ]
}
```

Política que permite la creación de un punto final de licencia

En este ejemplo, se crea una política exhaustiva que concede los permisos necesarios para crear y gestionar los puntos de enlace de licencia. Utilice esta política para crear el punto de enlace de licencia para la VPC asociada a su granja.

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/*"
        ]
    }]
}
```

Política que permite monitorear una cola de granja específica

En este ejemplo, se crea una política exhaustiva que concede permiso para supervisar los trabajos de una cola específica para una granja específica.

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 políticas gestionadas para Deadline Cloud

Una política AWS gestionada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

AWS política gestionada: AWSDeadlineCloud-FleetWorker

Puede adjuntar la `AWSDeadlineCloud-FleetWorker` política a sus identidades AWS Identity and Access Management (de IAM).

Esta política otorga a los trabajadores de esta flota los permisos necesarios para conectarse al servicio y recibir tareas del mismo.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite a los directores gestionar a los trabajadores de una flota.

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-FleetWorkerla guía de referencia de políticas administradas de AWS.

AWS política gestionada: AWSDeadlineCloud-WorkerHost

Puede asociar la política `AWSDeadlineCloud-WorkerHost` a las identidades de IAM.

Esta política concede los permisos necesarios para conectarse inicialmente al servicio. Se puede utilizar como perfil de instancia de Amazon Elastic Compute Cloud (Amazon EC2).

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite al usuario crear trabajadores, asumir el rol de flota para los trabajadores y aplicar etiquetas a los trabajadores

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-WorkerHostla guía de referencia de políticas administradas de AWS.

AWS política gestionada: AWSDeadlineCloud-UserAccessFarms

Puede asociar la política `AWSDeadlineCloud-UserAccessFarms` a las identidades de IAM.

Esta política permite a los usuarios acceder a los datos de las granjas en función de las granjas de las que son miembros y de su nivel de membresía.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite al usuario acceder a los datos de la granja.
- `ec2`— Permite a los usuarios ver detalles sobre los tipos de instancias de Amazon EC2.
- `identitystore`— Permite a los usuarios ver los nombres de usuarios y grupos.
- `kms`— Permite a los usuarios configurar AWS Key Management Service (AWS KMS) claves administradas por el cliente para su instancia AWS IAM Identity Center (IAM Identity Center).

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-UserAccessFarmsla guía de referencia de políticas administradas de AWS.

AWS política gestionada: AWSDeadlineCloud-UserAccessFleets

Puede asociar la política `AWSDeadlineCloud-UserAccessFleets` a las identidades de IAM.

Esta política permite a los usuarios acceder a los datos de la flota en función de las granjas de las que son miembros y de su nivel de membresía.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite al usuario acceder a los datos de la granja.
- `ec2`— Permite a los usuarios ver detalles sobre los tipos de instancias de Amazon EC2.
- `identitystore`— Permite a los usuarios ver los nombres de usuarios y grupos.

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-UserAccessFleetsla guía de referencia de políticas administradas de AWS.

AWS política gestionada: AWSDeadlineCloud-UserAccessJobs

Puede asociar la política `AWSDeadlineCloud-UserAccessJobs` a las identidades de IAM.

Esta política permite a los usuarios acceder a los datos de trabajo en función de las granjas de las que son miembros y de su nivel de membresía.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite al usuario acceder a los datos de la granja.
- `ec2`— Permite a los usuarios ver detalles sobre los tipos de instancias de Amazon EC2.
- `identitystore`— Permite a los usuarios ver los nombres de usuarios y grupos.

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-UserAccessJobsla guía de referencia de políticas administradas de AWS.

AWS política gestionada: AWSDeadlineCloud-UserAccessQueues

Puede asociar la política `AWSDeadlineCloud-UserAccessQueues` a las identidades de IAM.

Esta política permite a los usuarios acceder a los datos de las colas en función de las granjas de las que son miembros y de su nivel de membresía.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `deadline`— Permite al usuario acceder a los datos de la granja.
- `ec2`— Permite a los usuarios ver detalles sobre los tipos de instancias de Amazon EC2.
- `identitystore`— Permite a los usuarios ver los nombres de usuarios y grupos.

Para obtener una lista en JSON de los detalles de la política, consulte AWSDeadlineCloud-UserAccessQueuesla guía de referencia de políticas administradas de AWS.

Deadline Cloud actualiza a AWS políticas administradas

Consulta los detalles sobre las actualizaciones de las políticas AWS gestionadas de Deadline Cloud desde que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS de la página del historial de documentos de Deadline Cloud.

Cambio	Descripción	Fecha
AWSDeadlineCloud-UserAccessFarms: cambio	Deadline Cloud ha añadido una nueva acción `kms:Decrypt` para que puedas usar una clave AWS KMS gestionada por el cliente con tu instancia de IAM Identity Center.	22 de diciembre de 2025
AWSDeadlineCloud-WorkerHost: cambio	Deadline Cloud ha añadido nuevas acciones `deadline:TagResource` y `deadline:ListTagsForResource` te ha permitido añadir y ver las etiquetas asociadas a los trabajadores de tu flota.	30 de mayo de 2025
AWSDeadlineCloud-UserAccessFarms: cambio AWSDeadlineCloud-UserAccessJobs: cambio AWSDeadlineCloud-UserAccessQueues: cambio	Deadline Cloud ha añadido nuevas acciones `deadline:GetJobTemplate` y `deadline:ListJobParameterDefinitions` te ha permitido volver a enviar los trabajos.	7 de octubre de 2024
Deadline Cloud comenzó a rastrear los cambios	Deadline Cloud comenzó a rastrear los cambios en sus políticas AWS gestionadas.	2 de abril de 2024

Roles de servicio

Cómo utiliza Deadline Cloud las funciones de servicio de IAM

Deadline Cloud asume automáticamente las funciones de IAM y proporciona credenciales temporales a los trabajadores, los trabajos y el monitor de Deadline Cloud. Este enfoque elimina la administración manual de credenciales y, al mismo tiempo, mantiene la seguridad mediante un control de acceso basado en roles.

Al crear monitores, flotas y colas, debe especificar las funciones de IAM que Deadline Cloud asume en su nombre. A continuación, los trabajadores y el monitor de Deadline Cloud reciben credenciales temporales de estas funciones para poder acceder a ellas. Servicios de AWS

Función de flota

Configura un rol de flota para dar a los trabajadores de Deadline Cloud los permisos que necesitan para recibir trabajo e informar sobre el progreso de ese trabajo.

Por lo general, no es necesario que configure este rol usted mismo. Este rol se puede crear automáticamente en la consola de Deadline Cloud para incluir los permisos necesarios. Utilice la siguiente guía para comprender las características específicas de esta función a la hora de solucionar problemas.

Al crear o actualizar flotas mediante programación, especifique el ARN del rol de la flota mediante las operaciones o API. `CreateFleet` `UpdateFleet`

¿Qué hace la función de flota

La función de flota proporciona a los trabajadores permisos para:

- Reciba nuevos trabajos e informe sobre el progreso de los trabajos en curso al servicio Deadline Cloud
- Gestione el ciclo de vida y el estado del trabajador
- Registra los eventos de registro en Amazon CloudWatch Logs para los registros de los trabajadores

Configure la política de confianza en los roles de la flota

Tu función en la flota debe confiar en el servicio Deadline Cloud y estar relacionada con tu granja específica.

Como práctica recomendada, la política de confianza debe incluir condiciones de seguridad para la protección de Confused Deputy. Para obtener más información sobre la protección de Confused Deputy, consulte la guía del usuario de Confused Deadline Cloud.

- `aws:SourceAccount`garantiza que solo los recursos de la misma Cuenta de AWS entidad puedan asumir esta función.
- `aws:SourceArn`restringe la asunción de funciones a una granja específica de Deadline Cloud.

```
{
  "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"
        }
      }
    }
  ]
}
```

Adjunta los permisos del rol de Fleet

Adjunta la siguiente política AWS gestionada a tu función de flota:

AWSDeadlineCloud-FleetWorker

Esta política gestionada proporciona permisos para:

- `deadline:AssumeFleetRoleForWorker`- Permite a los trabajadores actualizar sus credenciales.

- `deadline:UpdateWorker`- Permite a los trabajadores actualizar su estado (por ejemplo, a PARADOS al salir).
- `deadline:UpdateWorkerSchedule`- Para obtener trabajo e informar sobre el progreso.
- `deadline:BatchGetJobEntity`- Para buscar información laboral.
- `deadline:AssumeQueueRoleForWorker`- Para acceder a las credenciales de los roles de cola durante la ejecución de un trabajo.

Agregue permisos de KMS para granjas cifradas

Si su granja se creó con una clave KMS, añada estos permisos a su función de flota para garantizar que el trabajador pueda acceder a los datos cifrados de la granja.

Los permisos de KMS solo son necesarios si la granja tiene una clave de KMS asociada. La `kms:ViaService` condición debe usar el formato`deadline.{region}.amazonaws.com`.

Al crear una flota, se crea un grupo de CloudWatch registros para esa flota. El servicio Deadline Cloud utiliza los permisos del trabajador para crear un flujo de registro específico para ese trabajador en particular. Una vez que el trabajador esté configurado y en funcionamiento, utilizará estos permisos para enviar los eventos del registro directamente a CloudWatch Logs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
```

```
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

Modificar el rol de la flota

Los permisos para el rol de flota no se pueden personalizar. Los permisos descritos son siempre obligatorios y añadir permisos adicionales no tiene ningún efecto.

Customer-managed función de anfitrión de flota

Configure un WorkerHost rol si usa flotas administradas por el cliente en instancias de Amazon EC2 o hosts locales.

¿Qué hace el rol WorkerHost

Esta WorkerHost función impulsa a los trabajadores de los anfitriones de flotas gestionados por el cliente. Proporciona los permisos mínimos necesarios para que un anfitrión pueda:

- Crea un trabajador en Deadline Cloud

- Asuma el rol de flota para obtener las credenciales operativas
- Etiquete a los trabajadores con etiquetas de flota (si la propagación de etiquetas está habilitada)

Configure los permisos de los WorkerHost roles

Adjunta la siguiente política AWS gestionada a tu WorkerHost rol:

AWSDeadlineCloud-WorkerHost

Esta política gestionada proporciona permisos para:

- `deadline:CreateWorker`- Permite al anfitrión registrar a un nuevo trabajador.
- `deadline:AssumeFleetRoleForWorker`- Permite al anfitrión asumir el rol de flota.
- `deadline:TagResource`- Permite etiquetar a los trabajadores durante la creación (si está activado).
- `deadline:ListTagsForResource`- Permite leer las etiquetas de la flota para su propagación.

Comprenda el proceso de arranque

El WorkerHost rol solo se usa durante la puesta en marcha inicial del trabajador:

1. El agente de trabajo se inicia en el host con WorkerHost las credenciales.
2. Invoca `deadline:CreateWorker` para registrarse en Deadline Cloud.
3. A continuación, invoca `deadline:AssumeFleetRoleForWorker` para obtener las credenciales del rol de la flota.
4. A partir de este momento, el trabajador solo utilizará las credenciales del rol de flota para todas las operaciones.

El WorkerHost rol no se usa después de que el trabajador comience a correr. Esta política no es obligatoria para Service-managed las flotas. En Service-managed las flotas, el arranque se realiza automáticamente.

Rol de cola

El trabajador asume la función de cola al procesar una tarea. Este rol proporciona los permisos necesarios para completar la tarea.

Al crear o actualizar colas mediante programación, especifique el ARN del rol de cola mediante las operaciones o API. `CreateQueue` `UpdateQueue`

Configure la política de confianza de los roles de cola

Tu rol de cola debe confiar en el servicio Deadline Cloud.

Como práctica recomendada, la política de confianza debe incluir condiciones de seguridad para la protección de Confused Deputy. Para obtener más información sobre la protección de Confused Deputy, consulte la guía del usuario de Confused Deadline Cloud.

- `aws:SourceAccount`garantiza que solo los recursos de la misma Cuenta de AWS entidad puedan asumir esta función.
- `aws:SourceArn`restringe la asunción de funciones a una granja específica de Deadline Cloud.

```
{
  "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}"
        }
      }
    }
  ]
}
```

Comprenda los permisos de los roles de cola

El rol de cola no usa una única política administrada. En su lugar, al configurar la cola en la consola, Deadline Cloud crea una política personalizada para la cola en función de la configuración.

Esta política creada automáticamente proporciona acceso a:

Adjuntos de trabajo

Acceso de lectura y escritura al bucket de Amazon S3 especificado para los archivos de entrada y salida de trabajos:

```
{
  "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"
    }
  }
}
```

Registros de trabajo

Acceso de lectura a CloudWatch los registros de los trabajos de esta cola. Cada cola tiene su propio grupo de registros y cada sesión tiene su propio flujo de registro:

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
```

```
}
```

Third-party software

Acceso para descargar software de terceros compatible con Deadline Cloud (como Maya, Blender y otros):

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Añade permisos para tus trabajos

Añada permisos a su función de cola a los Servicios de AWS que necesiten acceder sus trabajos. Al escribir scripts de OpenJobDescription pasos, el SDK AWS CLI y el SDK utilizarán automáticamente las credenciales de tu rol de cola. Utilícela para acceder a los servicios adicionales necesarios para completar su trabajo.

Entre los casos de uso de ejemplo se incluyen:

- para obtener datos personalizados
- Permisos SSM para acceder a un servidor de licencias personalizado
- CloudWatch para emitir métricas personalizadas
- Permiso de Deadline Cloud para crear nuevos trabajos para flujos de trabajo dinámicos

Cómo se utilizan las credenciales de los roles de cola

Deadline Cloud proporciona credenciales de rol de cola para:

- Trabajadores durante la ejecución del trabajo
- Los usuarios utilizan la CLI de Deadline Cloud y supervisan cuando interactúan con los archivos adjuntos y registros de los trabajos

Deadline Cloud crea grupos de CloudWatch registros independientes para cada cola. Los trabajos utilizan las credenciales de los roles de cola para escribir registros en el grupo de registros de su cola. La CLI y el monitor de Deadline Cloud utilizan la función de cola (mediante`deadline:AssumeQueueRoleForRead`) para leer los registros de trabajos del grupo de registros de la cola. La CLI y el monitor de Deadline Cloud utilizan la función de cola (mediante`deadline:AssumeQueueRoleForUser`) para cargar o descargar datos adjuntos de trabajos.

Función de supervisión

Configura una función de monitor para que las aplicaciones web y de escritorio del monitor de Deadline Cloud accedan a tus recursos de Deadline Cloud.

Al crear o actualizar monitores mediante programación, especifique el ARN del rol de monitor mediante las `CreateMonitor` operaciones o API. `UpdateMonitor`

¿Qué hace la función de monitor

La función de monitor permite a Deadline Cloud Monitor proporcionar a los usuarios finales acceso a:

- Funcionalidad básica requerida para los remitentes integrados, la CLI y el monitor de Deadline Cloud
- Funcionalidad personalizada para usuarios finales

Configure la política de confianza de los roles de monitor

Su función de monitor debe confiar en el servicio Deadline Cloud.

Como práctica recomendada, la política de confianza debe incluir condiciones de seguridad para la protección de Confused Deputy. Para obtener más información sobre la protección de Confused Deputy, consulte la guía del usuario de Confused Deadline Cloud.

`aws:SourceAccount`garantiza que solo los recursos de la misma Cuenta de AWS entidad puedan asumir esta función.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

Adjunte permisos a la función de monitor

Adjunte todas las siguientes políticas AWS administradas a su función de monitor para un funcionamiento básico:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

Cómo funciona la función de monitor

Cuando se utiliza el monitor de Deadline Cloud, un usuario del servicio inicia sesión con AWS IAM Identity Center (IAM Identity Center) y asume la función de monitor. La aplicación del monitor utiliza las credenciales del rol asumido para mostrar la interfaz de usuario del monitor, que incluye la lista de granjas, flotas, colas y otra información.

Cuando se utiliza la aplicación de escritorio de monitoreo Deadline Cloud, estas credenciales también están disponibles en la estación de trabajo mediante un perfil de AWS credenciales con nombre que corresponde al nombre del perfil proporcionado por el usuario final. Obtén más información sobre los perfiles con nombre en la guía de referencia del AWS SDK y las herramientas.

Este perfil con nombre es la forma en que la CLI de Deadline y los remitentes acceden a los recursos de Deadline Cloud.

Personalización de la función de monitor para casos de uso avanzados

Puede personalizar la función de monitor para modificar lo que los usuarios pueden hacer en cada nivel de acceso (espectador, colaborador, administrador, propietario) o para añadir permisos para flujos de trabajo avanzados.

Personalización de los permisos de nivel de acceso

Las cuatro políticas AWS administradas asociadas a la función de monitor controlan lo que puede hacer cada nivel de acceso. Puede añadir políticas personalizadas a la función de monitor para conceder o restringir los permisos para niveles de acceso específicos mediante la clave de `deadline:MembershipLevel` condición.

Por ejemplo, para permitir que los colaboradores actualicen y cancelen trabajos (lo que normalmente está restringido a los gerentes y propietarios), agrega una política como la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Con esta política, los colaboradores pueden actualizar y cancelar trabajos además de enviarlos.

Añadir permisos para flujos de trabajo avanzados

Puede añadir políticas de IAM personalizadas a la función de monitor para conceder permisos adicionales a todos los usuarios del monitor. Esto resulta útil para los flujos de trabajo de creación de

scripts avanzados en los que los usuarios necesitan acceder a una funcionalidad que Servicios de AWS va más allá de la estándar de Deadline Cloud.

Siga estas pautas al modificar su función de monitor:

- No elimines ninguna de las políticas gestionadas. La eliminación de estas políticas interrumpe la funcionalidad del monitor.

Cómo usa Deadline Cloud Monitor las credenciales de los roles de monitor

El monitor de Deadline Cloud obtiene automáticamente las credenciales del rol de monitor cuando te autenticas. Esta capacidad permite que la aplicación de escritorio proporcione capacidades de monitoreo mejoradas más allá de las disponibles en un navegador web estándar.

Cuando inicias sesión con el monitor Deadline Cloud, este crea automáticamente un perfil que puedes usar con esa AWS herramienta AWS CLI o con cualquier otra. Este perfil usa las credenciales del rol de monitor, lo que te da acceso mediante programación en Servicios de AWS función de los permisos de tu rol de monitor.

Los remitentes de Deadline Cloud funcionan de la misma manera: utilizan el perfil creado por el monitor de Deadline Cloud para acceder Servicios de AWS con los permisos de rol adecuados.

Personalización avanzada de los roles de Deadline Cloud

Puedes ampliar las funciones de Deadline Cloud con permisos adicionales para habilitar casos de uso avanzados que vayan más allá de los flujos de trabajo de renderizado básicos. Este enfoque aprovecha el sistema de gestión de acceso de Deadline Cloud para controlar el acceso a otros usuarios en Servicios de AWS función de la cantidad de usuarios que estén en cola.

Colaboración en equipo con AWS CodeCommit

Añade AWS CodeCommit permisos a tu rol de Queue para permitir la colaboración en equipo en los repositorios de proyectos. Este enfoque utiliza el sistema de gestión de acceso de Deadline Cloud para casos de uso adicionales: solo los usuarios con acceso a la cola específica recibirán estos AWS CodeCommit permisos, lo que te permitirá gestionar el acceso al repositorio por proyecto mediante la suscripción a Deadline Cloud.

Esto resulta útil en situaciones en las que los artistas necesitan acceder a activos, scripts o archivos de configuración específicos del proyecto almacenados en AWS CodeCommit repositorios como parte de su flujo de trabajo de renderizado.

Add (Suma) AWS CodeCommit permisos para poner en cola un rol

Añada los siguientes permisos a su rol de cola para habilitar AWS CodeCommit el acceso:

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Configura un proveedor de credenciales en las estaciones de trabajo de los artistas

Configura cada estación de trabajo de artistas para que utilice las credenciales de cola de Deadline Cloud para acceder. AWS CodeCommit Esta configuración se realiza una vez por estación de trabajo.

Para configurar el proveedor de credenciales

1. Agregue un perfil de proveedor de credenciales a su archivo de AWS configuración (`~/.aws/config`):

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Configura Git para usar este perfil en los AWS CodeCommit repositorios:

   ```
   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
   ```

Sustituya *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* y *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* por sus ID actuales de granja y cola. *REGION*Sustitúyalo por el de tu AWS región (por ejemplo,`us-west-2`).

Utilización AWS CodeCommit con credenciales de cola

Una vez configuradas, las operaciones de Git utilizarán automáticamente las credenciales del rol de cola al acceder a los AWS CodeCommit repositorios. El `deadline queue export-credentials` comando devuelve credenciales temporales con el siguiente aspecto:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Estas credenciales se actualizan automáticamente según sea necesario y las operaciones de Git funcionarán sin problemas:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Los artistas ahora pueden acceder a los repositorios de proyectos con sus permisos de cola sin necesidad de credenciales independientes. AWS CodeCommit Solo los usuarios con acceso a la cola específica podrán acceder al repositorio asociado, lo que permitirá un control de acceso detallado a través del sistema de membresía de colas de Deadline Cloud.

Resolución de problemas AWS Identidad y acceso a Deadline Cloud

Usa la siguiente información para ayudarte a diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con Deadline Cloud e IAM.

Temas

- No estoy autorizado a realizar ninguna acción en Deadline Cloud
- No estoy autorizado a realizar tareas como: PassRole
- Quiero permitir que personas ajenas a mi Cuenta de AWS para acceder a mis recursos de Deadline Cloud

No estoy autorizado a realizar ninguna acción en Deadline Cloud

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `deadline:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `deadline:`*`GetWidget`*.

Si necesitas ayuda, ponte en contacto con tu AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibes un mensaje de error que indica que no estás autorizado a realizar la `iam:PassRole` acción, debes actualizar tus políticas para que puedas transferir una función a Deadline Cloud.

Algunas te Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada al servicio. Para ello, debe tener permisos para transferir la función al servicio.

El siguiente ejemplo de error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Deadline Cloud. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir la función al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi Cuenta de AWS para acceder a mis recursos de Deadline Cloud

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Deadline Cloud admite estas funciones, consulteCómo funciona Deadline Cloud con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Validación de conformidad para Deadline Cloud

Para saber si un programa de cumplimiento Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y

reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al utilizarlos Servicios de AWS, consulte AWS la documentación de seguridad.

Resiliencia en Deadline Cloud

La infraestructura AWS global se basa en zonas Regiones de AWS de disponibilidad. Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre las zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

AWS Deadline Cloud no hace copias de seguridad de los datos almacenados en el depósito de S3 de sus adjuntos de trabajo. Puede habilitar las copias de seguridad de los datos adjuntos de su trabajo mediante cualquier mecanismo de copia de seguridad estándar de Amazon S3, como S3 Versioning o AWS Backup.

Seguridad de la infraestructura en Deadline Cloud

Como servicio gestionado, AWS Deadline Cloud está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a Deadline Cloud a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Paquetes de cifrado con perfecto secreto directo (PFS), como el DHE (Ephemeral) o el ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Deadline Cloud no admite el uso de políticas de puntos finales de nube privada AWS PrivateLink virtual (VPC). Utiliza la política AWS PrivateLink predeterminada, que otorga acceso total al punto final. Para obtener más información, consulte la política de puntos finales predeterminada en la guía del AWS PrivateLink usuario.

Análisis de configuración y vulnerabilidad en Deadline Cloud

AWS se encarga de tareas de seguridad básicas, como la aplicación de parches al sistema operativo (SO) huésped y a las bases de datos, la configuración del firewall y la recuperación ante desastres. Estos procedimientos han sido revisados y certificados por los terceros pertinentes. Para obtener más detalles, consulte los siguientes recursos de :

- Modelo de responsabilidad compartida
- Amazon Web Services: información general de procesos de seguridad (documento técnico)

AWS Deadline Cloud gestiona las tareas en flotas gestionadas por el servicio o por el cliente:

- En el caso de las flotas gestionadas por servicios, Deadline Cloud gestiona el sistema operativo huésped.
- En el caso de las flotas gestionadas por el cliente, usted es responsable de gestionar el sistema operativo.

Para obtener información adicional sobre la configuración y el análisis de vulnerabilidades de AWS Deadline Cloud, consulte

- Mejores prácticas de seguridad para Deadline Cloud

Cross-service confusa prevención de diputados

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar el confuso problema de un diputado. Cross-service la suplantación de identidad puede producirse cuando un servicio (el servicio de llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS

proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Se recomienda utilizar las claves de contexto de condición aws:SourceAccountglobal aws:SourceArny las claves de contexto en las políticas de recursos para limitar los permisos que se AWS Deadline Cloud otorgan a otro servicio al recurso. Utiliza `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utiliza `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el nombre de recurso de Amazon (ARN) completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:deadline:*:123456789012:*`.

Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar ambas claves de contexto de condición global para limitar los permisos.

En el siguiente ejemplo, se muestra cómo utilizar las claves de contexto de condición `aws:SourceAccount` global `aws:SourceArn` y las claves contextuales Deadline Cloud para evitar el confuso problema de los diputados.

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"
      }
    }
  }
}
```

Acceso AWS Deadline Cloud utilizando un punto final de interfaz (AWS PrivateLink)

Puede usarlo AWS PrivateLink para crear una conexión privada entre su VPC y. AWS Deadline Cloud Puede acceder Deadline Cloud como si estuviera en su VPC, sin el uso de una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o Direct Connect una conexión. Las instancias de la VPC no necesitan direcciones IP públicas para acceder a Deadline Cloud.

Esta conexión privada se establece mediante la creación de un punto de conexión de interfaz alimentado por AWS PrivateLink. Creamos una interfaz de red de punto de conexión en cada subred habilitada para el punto de conexión de interfaz. Se trata de interfaces de red administradas por el solicitante que sirven como punto de entrada para el tráfico destinado a Deadline Cloud.

Deadline Cloud también dispone de terminales de doble pila. Dual-stack los puntos finales admiten solicitudes a través de IPv6 e IPv4.

Para obtener más información, consulte Acceso a los Servicios de AWS a través de AWS PrivateLink en la Guía de AWS PrivateLink .

Consideraciones para Deadline Cloud

Antes de configurar un punto de enlace de interfaz para Deadline Cloud, consulte Acceder a un servicio de AWS mediante un punto de enlace de VPC de interfaz en la AWS PrivateLink Guía.

Deadline Cloud permite realizar llamadas a todas sus acciones de API a través del punto final de la interfaz.

De forma predeterminada, Deadline Cloud se permite el acceso total a través del punto final de la interfaz. Como alternativa, puede asociar un grupo de seguridad a las interfaces de red del punto final para controlar el tráfico que Deadline Cloud pasa por el punto final de la interfaz.

Deadline Cloud también es compatible con las políticas de puntos finales de VPC. Para obtener más información, consulte Uso de políticas de punto de conexión para controlar el acceso a puntos de conexión de VPC en la Guía de AWS PrivateLink .

Deadline Cloud puntos de conexión

Deadline Cloud utiliza cuatro puntos de conexión para acceder al servicio AWS PrivateLink : dos para IPv4 y dos para IPv6.

Los trabajadores utilizan el `scheduling.deadline.region.amazonaws.com` terminal para obtener las tareas de la cola, informar sobre su progreso y enviar los Deadline Cloud resultados de las tareas. Si utiliza una flota gestionada por el cliente, el punto final de programación es el único punto final que debe crear, a menos que utilice operaciones de gestión. Por ejemplo, si un trabajo crea más puestos de trabajo, debe habilitar el punto final de administración para que llame a la `CreateJob` operación.

El Deadline Cloud monitor lo utiliza `management.deadline.region.amazonaws.com` para administrar los recursos de la granja, por ejemplo, para crear y modificar colas y flotas o para obtener listas de trabajos, pasos y tareas.

Los AWS SDK y la CLI agregan automáticamente los `scheduling` prefijos `management` y al punto final. Si desea deshabilitar este comportamiento, consulte la sección sobre la inyección de prefijos de host en la Guía de referencia de herramientas y AWS SDK.

Deadline Cloud también requiere puntos finales para los siguientes puntos finales de servicio: AWS

- Si configuras tu flota gestionada por el cliente en una subred sin conexión a Internet, debes crear un punto de enlace de VPC para CloudWatch Amazon Logs para que los trabajadores puedan escribir registros. Para obtener más información, consulte Monitorear con. CloudWatch
- Si usa adjuntos de trabajo, debe crear un punto de enlace de VPC para Amazon Simple Storage Service (Amazon S3) para que los trabajadores puedan acceder a los archivos adjuntos. Para obtener más información, consulte Adjuntos de trabajos en Deadline Cloud.

Cree puntos finales para Deadline Cloud

Puede crear puntos de enlace de interfaz para Deadline Cloud utilizar la consola de Amazon VPC o AWS Command Line Interface ().AWS CLI Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía de AWS PrivateLink .

Cree puntos de enlace de administración y programación para Deadline Cloud utilizar los siguientes nombres de servicio. *region*Sustitúyalos por el Región de AWS lugar donde lo implementó Deadline Cloud.

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud admite puntos finales de doble pila.

Si habilita el DNS privado para los puntos finales de la interfaz, puede realizar solicitudes a la API Deadline Cloud utilizando su nombre de DNS regional predeterminado. Por ejemplo, `scheduling.deadline.us-east-1.amazonaws.com` para las operaciones de los trabajadores o `management.deadline.us-east-1.amazonaws.com` para todas las demás operaciones.

Si su flota gestionada por el cliente se encuentra en una subred sin conexión a Internet, debe crear un punto final de CloudWatch Logs con el siguiente nombre de servicio:

```
com.amazonaws.region.logs
```

Si utiliza adjuntos de trabajo para transferir archivos, debe crear un punto de conexión de Amazon S3 con el siguiente nombre de servicio:

```
com.amazonaws.region.s3
```

Entornos de red restringidos

Deadline Cloud proporciona herramientas que utilizan los artistas u otros usuarios en sus estaciones de trabajo locales. Estas herramientas requieren acceso a la AWS API y a los puntos finales web para realizar su función. Si filtra el acceso a AWS dominios o puntos de enlace de URL específicos mediante una solución de filtrado de contenido web, como firewalls de última generación (NGFW) o Secure Web Gateways (SWG), debe añadir los siguientes dominios o puntos de enlace de URL a las listas de permisos de la solución de filtrado de contenido web.

AWS Puntos finales de API a la lista de permitidos

Las herramientas de cliente de Deadline Cloud Consola de administración de AWS, como el monitor, la CLI y los remitentes integrados, requieren acceso a AWS las API además de a Deadline Cloud. Estos puntos de conexión solo son compatibles con 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`

Lista de dominios web que se van a permitir

El monitor Deadline Cloud requiere acceso a los siguientes dominios para funcionar.

Para obtener información adicional sobre los dominios permitidos AWS Sign-In, consulte Dominios para añadir a su lista de dominios permitidos en la Guía del AWS Sign-In usuario.

- `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 puntos finales para permitir la lista

Estos dominios varían según la configuración específica de Deadline Cloud. Si se crean más monitores o colas de Deadline Cloud, será necesario incluir dominios adicionales en la lista de permitidos.

- `[Directory ID or alias].awsapps.com`

  Este dominio está vinculado a la configuración del IAM Identity Center y debe ser el mismo para todas las configuraciones que utilicen la misma instancia del IAM Identity Center. El administrador de la empresa puede encontrar el valor exacto en la consola del IAM Identity Center, en Configuración → URL.Portal de acceso a AWS
- `[Monitor alias].[Region].deadlinecloud.amazonaws.com`

  Este dominio es para la configuración del monitor en Deadline Cloud. Los artistas introducen este enlace en su navegador o en la aplicación de monitoreo de Deadline Cloud. Si Deadline Cloud se configura en cuentas o regiones adicionales en el futuro, este dominio cambiará. Puedes encontrar este valor en la consola de Deadline Cloud, en el panel de control → Descripción general del monitor → Detalles del monitor → URL.
- `[Bucket name].[Region].s3.amazonaws.com`

  Este es el dominio del grupo de adjuntos de trabajos que utilizan las colas de Deadline Cloud. Cada cola puede tener configurado su propio depósito de adjuntos de trabajos. El nombre exacto del bucket se encuentra en la consola de Deadline Cloud, en Colas → Detalles de la cola → Adjuntos de trabajos. Para obtener más información sobre los adjuntos de trabajos, consulta la documentación sobre las colas.

Mejores prácticas de seguridad para Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) ofrece una serie de características de seguridad que debes tener en cuenta a la hora de desarrollar e implementar tus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

> **Note**
>
> Para obtener más información sobre la importancia de muchos temas de seguridad, consulte el Modelo de responsabilidad compartida.

Protección de datos

Para proteger los datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure cuentas individuales con AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon Simple Storage Service (Amazon S3).
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información acerca de los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de formato libre, como el campo Nombre. Esta recomendación se incluye cuando trabajas con AWS Deadline Cloud u otro Servicios de AWS dispositivo que utilice la consola AWS CLI, la API o AWS los SDK. Todos los datos que introduzcas en Deadline Cloud u otros servicios podrían recogerse para incluirlos en los registros de diagnóstico. Cuando le proporcione una URL a un servidor externo, no incluya información sobre las credenciales en la URL para validar la solicitud en ese servidor.

AWS Identity and Access Management permisos

Administre el acceso a los AWS recursos utilizando los usuarios, las funciones AWS Identity and Access Management (IAM) y concediendo el mínimo de privilegios a los usuarios. Establezca políticas y procedimientos de administración de credenciales para crear, distribuir, rotar y revocar AWS las credenciales de acceso. Para obtener más información, consulte Prácticas recomendadas de IAM en la Guía del usuario de IAM.

Ejecute trabajos como usuarios y grupos

Al utilizar la funcionalidad de colas en Deadline Cloud, se recomienda especificar un usuario del sistema operativo (SO) y su grupo principal para que el usuario del sistema operativo tenga los permisos con menos privilegios para los trabajos de la cola.

Si especificas «Ejecutar como usuario» (y grupo), todos los procesos de los trabajos enviados a la cola se ejecutarán con ese usuario del sistema operativo y heredarán los permisos del sistema operativo asociados a ese usuario.

Las configuraciones de flota y cola se combinan para establecer una postura de seguridad. Por el lado de la cola, se pueden especificar el rol «Job run as user» y el rol de IAM para usar el sistema operativo y AWS los permisos para los trabajos de la cola. La flota define la infraestructura (servidores de los trabajadores, redes, almacenamiento compartido montado) que, cuando se asocia a una cola determinada, ejecuta los trabajos dentro de la cola. Los trabajos de una o más colas asociadas deben acceder a los datos disponibles en los hosts de los trabajadores. La especificación de un usuario o un grupo ayuda a proteger los datos de los trabajos frente a otras colas, otro software instalado u otros usuarios con acceso a los hosts de los trabajadores. Cuando una cola no tiene un usuario, se ejecuta como el usuario agente, que puede hacerse pasar por (`sudo`) cualquier usuario de la cola. De esta forma, una cola sin un usuario puede escalar los privilegios a otra cola.

Red

Para evitar que el tráfico sea interceptado o redirigido, es fundamental proteger cómo y hacia dónde se enruta el tráfico de la red.

Le recomendamos que proteja su entorno de red de las siguientes maneras:

- Proteja las tablas de enrutamiento de subred de Amazon Virtual Private Cloud (Amazon VPC) para controlar cómo se enruta el tráfico de la capa IP.

- Si utiliza Amazon Route 53 (Route 53) como proveedor de DNS en la configuración de su granja o estación de trabajo, asegure el acceso a la API de Route 53.
- Si se conecta a Deadline Cloud desde fuera, por AWS ejemplo, mediante estaciones de trabajo locales u otros centros de datos, proteja cualquier infraestructura de red local. Esto incluye los servidores DNS y las tablas de enrutamiento en enrutadores, conmutadores y otros dispositivos de red.

Trabajos y datos de trabajos

Los trabajos de Deadline Cloud se ejecutan dentro de las sesiones en los anfitriones de los trabajadores. Cada sesión ejecuta uno o más procesos en el host del trabajador, que por lo general requieren la introducción de datos para generar resultados.

Para proteger estos datos, puede configurar los usuarios del sistema operativo con colas. El agente de trabajo utiliza el usuario del sistema operativo de colas para ejecutar los subprocesos de la sesión. Estos subprocesos heredan los permisos del usuario del sistema operativo de colas.

Le recomendamos que siga las mejores prácticas para proteger el acceso a los datos a los que acceden estos subprocesos. Para obtener más información, consulte el Modelo de responsabilidad compartida.

Estructura de la granja

Puedes organizar las flotas y colas de Deadline Cloud de muchas maneras. Sin embargo, algunos acuerdos tienen implicaciones de seguridad.

Una granja tiene uno de los límites más seguros porque no puede compartir los recursos de Deadline Cloud con otras granjas, incluidas las flotas, las colas y los perfiles de almacenamiento. Sin embargo, puedes compartir AWS recursos externos dentro de una granja, lo que pone en peligro el límite de seguridad.

También puede establecer límites de seguridad entre las colas de la misma granja mediante la configuración adecuada.

Siga estas prácticas recomendadas para crear colas seguras en la misma granja:

- Asocie una flota únicamente a las colas que se encuentren dentro del mismo límite de seguridad. Tenga en cuenta lo siguiente:

  - Una vez que el trabajo se ejecuta en el host de trabajo, es posible que los datos permanezcan ocultos, por ejemplo, en un directorio temporal o en el directorio principal del usuario de la cola.
  - El mismo usuario del sistema operativo ejecuta todos los trabajos en un host de trabajadores de flota propiedad del servicio, independientemente de la cola a la que envíe el trabajo.
  - Un trabajo puede dejar los procesos ejecutándose en un host de trabajo, lo que permite que los trabajos de otras colas observen otros procesos en ejecución.
- Asegúrese de que solo las colas que se encuentren dentro del mismo límite de seguridad compartan un bucket de Amazon S3 para adjuntar trabajos.
- Asegúrese de que solo las colas que se encuentren dentro del mismo límite de seguridad compartan un usuario del sistema operativo.
- Proteja cualquier otro AWS recurso que esté integrado en la granja hasta el límite.

Colas de adjuntos de trabajos

Los adjuntos de trabajos se asocian a una cola, que utiliza tu bucket de Amazon S3.

- Los adjuntos de trabajo se escriben y se leen desde un prefijo raíz del bucket de Amazon S3. Este prefijo raíz se especifica en la llamada a la `CreateQueue` API.
- El bucket tiene una correspondiente`Queue Role`, que especifica la función que concede a los usuarios de la cola acceso al bucket y al prefijo raíz. Al crear una cola, debe especificar el nombre del recurso de `Queue Role` Amazon (ARN) junto con el depósito de adjuntos de trabajos y el prefijo raíz.
- Las llamadas autorizadas a las operaciones de `AssumeQueueRoleForReadAssumeQueueRoleForUser`, y `AssumeQueueRoleForWorker` API devuelven un conjunto de credenciales de seguridad temporales para. `Queue Role`

Si crea una cola y reutiliza un bucket y un prefijo raíz de Amazon S3, existe el riesgo de que la información se divulgue a terceros no autorizados. Por ejemplo, QueueA y QueueB comparten el mismo bucket y el mismo prefijo raíz. En un flujo de trabajo seguro, Artista tiene acceso a QueueA pero no a QueueB. Sin embargo, cuando varias colas comparten un depósito, Artista puede acceder a los datos de los datos de QueueB porque utiliza el mismo depósito y el mismo prefijo raíz que QueuEa.

La consola configura colas que son seguras de forma predeterminada. Asegúrese de que las colas tengan una combinación distinta de bucket de Amazon S3 y prefijo raíz, a menos que formen parte de un límite de seguridad común.

Para aislar las colas, debe configurarlas de manera que solo se permita el `Queue Role` acceso de las colas al bucket y al prefijo raíz. En el siguiente ejemplo, sustituya cada uno por la información específica del *`placeholder`* recurso.

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/*"
        }
    ]
}
```

También debe establecer una política de confianza para el rol. En el siguiente ejemplo, sustituya el *placeholder* texto por la información específica del recurso.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
```

```
        }
    ]
}
```

Buckets Amazon S3 de software personalizados

Puede añadir la siguiente declaración para acceder `Queue Role` al software personalizado de su bucket de Amazon S3. En el siguiente ejemplo, *`SOFTWARE_BUCKET_NAME`* sustitúyalo por el nombre del bucket de S3 y *`BUCKET_ACCOUNT_OWNER`* por el Cuenta de AWS ID propietario 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"
         }
      }
    }
]
```

Para obtener más información sobre las prácticas recomendadas de seguridad de Amazon S3, consulte las prácticas recomendadas de seguridad para Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

Los trabajadores son anfitriones

Proteja los hosts de los trabajadores para garantizar que cada usuario solo pueda realizar operaciones para el rol que se le ha asignado.

Recomendamos las siguientes prácticas recomendadas para proteger los anfitriones de los trabajadores:

- El uso de un script de configuración de host puede cambiar la seguridad y las operaciones de un trabajador. Una configuración incorrecta puede provocar que el trabajador sea inestable o deje de trabajar. Es su responsabilidad depurar dichos errores.
- No utilice el mismo `jobRunAsUser` valor con varias colas, a menos que los trabajos enviados a esas colas estén dentro del mismo límite de seguridad.
- No `jobRunAsUser` defina la cola con el nombre del usuario del sistema operativo en el que se ejecuta el agente de trabajo.
- Otorgue a los usuarios de la cola los permisos de sistema operativo con menos privilegios necesarios para las cargas de trabajo de cola previstas. Asegúrese de que no tengan permisos de escritura en el sistema de archivos para trabajar, agentes, archivos de programas u otro software compartido.
- Asegúrese de que solo el usuario root Linux y el `Administrator` propietario de la cuenta posean los Windows archivos del programa del agente de trabajo y puedan modificarlos.
- En los Linux hosts de trabajo, considere la posibilidad `umask` de configurar una alternativa `/etc/sudoers` que permita al usuario del agente de trabajo iniciar procesos como usuarios en cola. Esta configuración ayuda a garantizar que otros usuarios no puedan acceder a los archivos escritos en la cola.
- Otorgue a las personas de confianza con menos privilegios el acceso a los anfitriones de los trabajadores.
- Restrinja los permisos a los archivos de configuración de anulación del DNS local (activos y `/etc/hosts` activosWindows) Linux y a las tablas de enrutamiento `C:\Windows\system32\etc\hosts` en las estaciones de trabajo y los sistemas operativos de los hosts de trabajo.
- Restrinja los permisos a la configuración de DNS en las estaciones de trabajo y los sistemas operativos anfitriones de los trabajadores.
- Aplica parches periódicos al sistema operativo y a todo el software instalado. Este enfoque incluye el software que se utiliza específicamente con Deadline Cloud, como los remitentes, los adaptadores, los agentes de trabajo, OpenJD los paquetes y otros.
- Usa contraseñas seguras para la Windows cola. `jobRunAsUser`
- Cambia las contraseñas de la cola `jobRunAsUser` con regularidad.
- Asegúrese de que el acceso a las Windows contraseñas secretas sea lo más mínimo posible y elimine las que no se utilicen.
- No dé `jobRunAsUser` permiso a la cola para que los comandos de programación se ejecuten en el futuro:

- SíLinux, deniega a estas cuentas el acceso a `cron` `yat`.
- ActivadoWindows, deniega el acceso de estas cuentas al programador de Windows tareas.

> **Note**
>
> Para obtener más información sobre la importancia de actualizar periódicamente el sistema operativo y el software instalado, consulte el Modelo de responsabilidad compartida.

Script de configuración del host

- El uso de un script de configuración de host puede cambiar la seguridad y las operaciones de un trabajador. Una configuración incorrecta puede provocar que el trabajador sea inestable o deje de trabajar. Es su responsabilidad depurar dichos errores.

Estaciones de trabajo

Es importante proteger las estaciones de trabajo con acceso a Deadline Cloud. Este enfoque ayuda a garantizar que los trabajos que envíes a Deadline Cloud no puedan ejecutar cargas de trabajo arbitrarias que se te facturen. Cuenta de AWS

Recomendamos las siguientes prácticas recomendadas para proteger las estaciones de trabajo de los artistas. Para obtener más información, consulte Modelo de responsabilidad compartida de .

- Proteja todas las credenciales persistentes a las que pueda acceder AWS, incluida Deadline Cloud. Para obtener más información, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.
- Instale únicamente software seguro y confiable.
- Exija a los usuarios que se federen con un proveedor de identidad para acceder AWS con credenciales temporales.
- Utilice permisos seguros en los archivos del programa de envío de Deadline Cloud para evitar su manipulación.
- Conceda a las personas de confianza con menos privilegios el acceso a las estaciones de trabajo de los artistas.
- Utilice únicamente los remitentes y adaptadores que obtenga a través del Deadline Cloud Monitor.

- Restrinja los permisos a los archivos de configuración de anulación del DNS local (`/etc/hosts`activos Linux y activos macOSWindows) y `C:\Windows\system32\etc\hosts` a las tablas de enrutamiento de las estaciones de trabajo y los sistemas operativos de los anfitriones de los trabajadores.
- Restrinja los permisos a `/etc/resolve.conf` las estaciones de trabajo y a los sistemas operativos anfitriones de los trabajadores.
- Aplica parches periódicos al sistema operativo y a todo el software instalado. Este enfoque incluye el software que se utiliza específicamente con Deadline Cloud, como los remitentes, los adaptadores, los agentes de trabajo, OpenJD los paquetes y otros.

Compruebe la autenticidad del software descargado

Compruebe la autenticidad del software después de descargar el instalador para protegerlo de la manipulación de archivos. Este procedimiento funciona en ambos Windows Linux sistemas.

Windows

Para comprobar la autenticidad de los archivos descargados, complete los siguientes pasos.

1. En el siguiente comando, *`file`* reemplácelo por el archivo que desee comprobar. Por ejemplo, ***`C:\PATH\TO\MY\`*`DeadlineCloudSubmitter-windows-x64-installer.exe`** . Además, *`signtool-sdk-version`* sustitúyalo por la versión del SignTool SDK instalada. Por ejemplo, **`10.0.22000.0`**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Por ejemplo, puede verificar el archivo de instalación del remitente de Deadline Cloud ejecutando el siguiente comando:

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-windows-x64-installer.exe
   ```

Linux

Para comprobar la autenticidad de los archivos descargados, utilice la herramienta de línea de `gpg` comandos.

1. Importe la 0penPGP clave ejecutando el siguiente 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. Determine si se debe confiar en la 0penPGP clave. Algunos factores que se deben tener en cuenta al decidir si se debe confiar en la clave anterior son los siguientes:

   - La conexión a Internet que has utilizado para obtener la clave GPG de este sitio web es segura.
   - El dispositivo desde el que accedes a este sitio web es seguro.
   - AWS ha tomado medidas para garantizar el alojamiento de la clave 0penPGP pública en este sitio web.

3. Si decide confiar en la OpenPGP clave, edítela de forma gpg similar al ejemplo siguiente:

```
$ 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 el instalador del remitente de Deadline Cloud

   Para verificar el instalador del remitente de Deadline Cloud, complete los siguientes pasos:

   a. Descarga el archivo de firma del instalador de Deadline Cloud Submitter.

      Descargue el archivo de firma (.sig)

   b. Compruebe la firma del instalador del remitente de Deadline Cloud ejecutando:

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. Verifica el monitor de Deadline Cloud

   > **Note**
   >
   > Puede verificar la descarga del monitor Deadline Cloud mediante archivos de firmas o métodos específicos de la plataforma. Para conocer los métodos específicos de la plataforma, consulta la Linux (Debian) pestaña, la pestaña Linux (RPM) o la Linux (AppImage) pestaña según el tipo de archivo descargado.

   Para verificar la aplicación de escritorio Deadline Cloud Monitor con los archivos de firmas, complete los siguientes pasos:

   a. Descarga el archivo de firma correspondiente para tu instalador de monitores de Deadline Cloud:

      - Descarga el archivo de firma.deb
      - Descargue el archivo de firma.rpm
      - Descarga. AppImage archivo de firmas

   b. Compruebe la firma:

      Para .deb:

      ```
      gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
      monitor_amd64.deb
      ```

      Para .rpm:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

Para. AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Confirme que el resultado tiene un aspecto similar al siguiente:

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Si el resultado contiene la fraseGood `signature from "AWS Deadline Cloud"`, significa que la firma se ha verificado correctamente y que puede ejecutar el script de instalación del monitor Deadline Cloud.

Claves históricas

```
-----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+VpkpQmLgwtMGpSMLl3KLwnv2k
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)

Para verificar los paquetes que utilizan unLinux. AppImage binario, primero complete los pasos 1 a 3 de la Linux pestaña y, a continuación, complete los pasos siguientes.

1. Desde la AppImageUpdate página de inicio GitHub, descargue el archivo validate-x86_64. AppImagearchivo.
2. Tras descargar el archivo, para añadir permisos de ejecución, ejecute el siguiente comando.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```

3. Para añadir permisos de ejecución, ejecute el siguiente comando.

   ```
   chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

4. Para verificar la firma del monitor de Deadline Cloud, ejecute el siguiente comando.

   ```
   ./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

   Si el resultado contiene la fraseValidation successful, significa que la firma se ha verificado correctamente y que puede ejecutar de forma segura el script de instalación del monitor Deadline Cloud.

Linux (Debian)

Para verificar los paquetes que utilizan un archivo binario Linux .deb, primero complete los pasos 1 a 3 de la Linux pestaña.

dpkg es la herramienta principal de administración de paquetes en la mayoría debian de las distribuciones basadas. Linux Puede verificar el archivo.deb con la herramienta.

1. Descarga el archivo .deb del monitor de Deadline Cloud:

   Descargue el monitor Deadline Cloud (.deb)

2. Compruebe el archivo.deb:

   ```
   dpkg-sig --verify deadline-cloud-monitor_amd64.deb
   ```

3. El resultado será similar al siguiente:

   ```
   Processing deadline-cloud-monitor_amd64.deb...
   GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
   ```

4. Para verificar el archivo.deb, confirme que `GOODSIG` esté presente en la salida.

Linux (RPM)

Para verificar los paquetes que utilizan un archivo binario Linux .rpm, primero complete los pasos 1 a 3 de la pestaña. Linux

1. Descargue el archivo .rpm del monitor de Deadline Cloud:

   Descargue el monitor Deadline Cloud (.rpm)

2. Verifique el archivo .rpm:

   ```
   gpg --export --armor "Deadline Cloud" > key.pub
   sudo rpm --import key.pub
   rpm -K deadline-cloud-monitor.x86_64.rpm
   ```

3. El resultado será similar al siguiente:

   ```
   deadline-cloud-monitor.x86_64.rpm: digests signatures OK
   ```

4. Para verificar el archivo.rpm, confirme que `digests signatures OK` está en la salida.

Historial del documento

Para obtener información sobre las actualizaciones de AWS Deadline Cloud, consulta las notas de lanzamiento de Deadline Cloud.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.