aws

Guia do desenvolvedor

Nuvem de prazos

Nuvem de prazos: Guia do desenvolvedor

Table of Contents

O que é AWS Deadline Cloud?

AWS O Deadline Cloud é um AWS serviço totalmente gerenciado que permite que você tenha uma fazenda de processamento escalável em funcionamento em minutos. Ele fornece um console de administração para gerenciar usuários, fazendas, filas para agendar trabalhos e frotas de trabalhadores que fazem o processamento.

Este guia do desenvolvedor é para desenvolvedores de funis, ferramentas e aplicativos em uma ampla variedade de casos de uso, incluindo os seguintes:

- Desenvolvedores de funis e diretores técnicos podem integrar o Deadline Cloud APIs e os recursos em seus pipelines de produção personalizados.
- Fornecedores independentes de software podem integrar o Deadline Cloud em seus aplicativos, permitindo que artistas e usuários de criação de conteúdo digital enviem trabalhos de renderização do Deadline Cloud sem problemas a partir de suas estações de trabalho.
- Os desenvolvedores de serviços baseados na web e na nuvem podem integrar a renderização do Deadline Cloud em suas plataformas, permitindo que os clientes forneçam ativos para visualizar os produtos virtualmente.

Fornecemos ferramentas que permitem que você trabalhe diretamente com qualquer etapa do seu pipeline:

- Uma interface de linha de comando que você pode usar diretamente ou a partir de scripts.
- O AWS SDK para 11 linguagens de programação populares.
- Uma interface web baseada em REST que você pode chamar a partir de seus aplicativos.

Você também pode usar outros Serviços da AWS em seus aplicativos personalizados. Por exemplo, você pode usar:

- AWS CloudFormationpara automatizar a criação e remoção de fazendas, filas e frotas.
- Amazon CloudWatch reunirá métricas para empregos.
- Amazon Simple Storage Service para armazenar e gerenciar ativos digitais e resultados de trabalhos.
- Centro de Identidade do AWS IAMpara gerenciar usuários e grupos para suas fazendas.

Descrição do Open Job

O Deadline Cloud usa a especificação Open Job Description (OpenJD) para especificar os detalhes de um trabalho. O OpenJD foi desenvolvido para definir trabalhos que são portáteis entre as soluções. Você o usa para definir um trabalho que é um conjunto de comandos executados em hosts de trabalho.

Você pode criar um modelo de trabalho do OpenJD usando um remetente fornecido pelo Deadline Cloud ou pode usar qualquer ferramenta que desejar para criar o modelo. Depois de criar o modelo, você o envia para o Deadline Cloud. Se você usa um remetente, ele se encarrega de enviar o modelo. Se você criou o modelo de outra forma, você chama uma ação de linha de comando do Deadline Cloud ou pode usar uma delas AWS SDKs para enviar o trabalho. De qualquer forma, o Deadline Cloud adiciona o trabalho à fila especificada e agenda o trabalho.

Conceitos e terminologia do Deadline Cloud

Para ajudar você a começar a usar AWS o Deadline Cloud, este tópico explica alguns de seus principais conceitos e terminologia.

Recursos agrícolas

Este diagrama mostra como os recursos agrícolas do Deadline Cloud funcionam juntos.

Farm

Uma fazenda contém todos os outros recursos relacionados ao envio e execução de trabalhos. As fazendas são independentes umas das outras, o que as torna úteis para separar ambientes de produção.

Fila

Uma fila contém trabalhos para agendamento em frotas associadas. Os usuários podem enviar trabalhos para uma fila e gerenciar sua prioridade e status dentro da fila. Uma fila deve estar associada a uma frota com uma associação fila-frota para que seus trabalhos sejam executados, e as filas podem ser associadas a várias frotas.

Frota

Uma frota contém capacidade computacional para executar trabalhos. As frotas podem ser gerenciadas pelo serviço ou pelo cliente. As frotas gerenciadas por serviços funcionam no

Deadline Cloud e incluem funcionalidades integradas, como escalonamento automático, licenciamento e acesso ao software. As frotas gerenciadas pelo cliente são executadas em seus próprios recursos computacionais, como EC2 instâncias da Amazon ou servidores locais.

Orçamento

Um orçamento define limites de gastos para sua atividade de trabalho e permite que você tome medidas quando os limites são atingidos, como interromper o agendamento de trabalhos.

Ambiente de filas

Um ambiente de fila define scripts que são executados em cada trabalhador para configurar ou desmontar o ambiente de carga de trabalho. Eles são úteis para definir variáveis de ambiente, instalar software e configurar o armazenamento de ativos.

Perfil de armazenamento

Um perfil de armazenamento é uma configuração para um grupo de hosts e estações de trabalho, que informa onde os dados estão localizados no sistema de arquivos. O Deadline Cloud usa perfis de armazenamento para mapear caminhos ao executar trabalhos em hosts configurados de forma diferente, como um trabalho enviado Windows e executado emLinux.

Limite

Um limite permite que você acompanhe o uso de recursos compartilhados, como licenças flutuantes, e controle como eles são alocados entre trabalhos. Os limites são associados a filas com associações de limite de fila.

Monitorar

O monitor configura a URL do aplicativo web Deadline Cloud Monitor, permitindo que os usuários finais monitorem e gerenciem trabalhos. Ele pode ser acessado em um navegador ou por meio do aplicativo de desktop Deadline Cloud Monitor.

Recursos de execução de trabalhos

Este diagrama mostra como os recursos de trabalho do Deadline Cloud funcionam juntos.

Trabalho

Um trabalho é um conjunto de trabalhos que um usuário envia ao Deadline Cloud para serem agendados e executados com os trabalhadores disponíveis. Um trabalho pode renderizar uma

cena 3D ou executar uma simulação. Os trabalhos são criados a partir de modelos de trabalho reutilizáveis, que definem o ambiente e os processos de tempo de execução e os parâmetros específicos do trabalho. Os trabalhos contêm etapas e tarefas que definem o trabalho a ser executado e podem ser configurados com prioridades, número máximo de trabalhadores e configurações de repetição.

Prioridade do trabalho

A prioridade do trabalho é a ordem aproximada em que o Deadline Cloud processa um trabalho em uma fila. Você pode definir a prioridade do trabalho entre 1 e 100. Os trabalhos com maior prioridade numérica geralmente são processados primeiro. Os trabalhos com a mesma prioridade são processados na ordem recebida.

Propriedades do trabalho

As propriedades do trabalho são configurações que você define ao enviar um trabalho de renderização. Alguns exemplos incluem faixa de quadros, caminho de saída, anexos de tarefas, câmera renderizável e muito mais. As propriedades variam com base no DCC do qual a renderização é enviada.

Etapa

Uma etapa faz parte de um trabalho que fornece um modelo para executar várias tarefas idênticas, exceto pelos valores dos parâmetros da tarefa. As etapas podem depender de outras etapas, permitindo que você crie fluxos de trabalho complexos com caminhos de execução sequenciais ou paralelos. Em trabalhos de renderização, uma etapa geralmente define o comando para renderizar um quadro e usa o número do quadro como parâmetro da tarefa.

Tarefa

Uma tarefa é a menor unidade de trabalho no Deadline Cloud. As tarefas fazem parte das etapas e são executadas pelos trabalhadores, representando operações individuais que precisam ser executadas como parte de um trabalho. As tarefas podem ser configuradas com parâmetros específicos e são atribuídas aos trabalhadores com base em suas capacidades e disponibilidade. Em trabalhos de renderização, uma tarefa geralmente renderiza um único quadro.

Operador

Os trabalhadores fazem parte de uma frota e executam tarefas a partir dos trabalhos. Os trabalhadores podem ser configurados com recursos específicos, como aceleradores de GPU, arquitetura de CPU e sistema operacional. Em frotas gerenciadas por serviços, os trabalhadores são criados automaticamente à medida que a frota aumenta e aumenta.

Instância

As frotas usam instâncias para recursos de CPU. Uma instância é uma instância de EC2 desempenho da Amazon. O Deadline Cloud usa instâncias sob demanda e spot.

Instância sob demanda

As instâncias sob demanda são precificadas por segundo, não têm compromisso de longo prazo e não serão interrompidas.

Instância spot

As instâncias spot são uma capacidade não reservada que você pode usar com desconto, mas pode ser interrompida por solicitações sob demanda.

Espere e salve

O recurso Esperar e Salvar fornece agendamento de trabalhos atrasado por um custo menor e pode ser interrompido por solicitações sob demanda e pontuais. O Wait and Save só está disponível nas frotas gerenciadas pelo serviço Deadline Cloud.

Wait and Save serve para gerenciar a execução de cargas de trabalho de computação visual no AWS Deadline Cloud. Consulte os termos AWS de serviço para obter detalhes.

Sessão

Uma sessão representa a sequência de trabalho de um trabalhador em um trabalho. Durante uma única sessão, um trabalhador pode receber várias tarefas que ele executa uma após a outra. As sessões geralmente têm ações de configuração que configuram ambientes e carregam ativos antes de executar as ações da tarefa.

Ação da sessão

Uma ação de sessão representa operações específicas realizadas durante uma sessão, como configurar o ambiente, executar uma tarefa e sincronizar ativos.

Outros conceitos e terminologia importantes

Explorador de uso

O explorador de uso é um recurso do monitor Deadline Cloud. Ele fornece uma estimativa aproximada de seus custos e uso.

Gerente de orçamento

O gerente de orçamento faz parte do monitor Deadline Cloud. Use o gerenciador de orçamento para criar e gerenciar orçamentos. Você também pode usá-lo para limitar as atividades para ficar dentro do orçamento.

Biblioteca de cliente do Deadline Cloud

A biblioteca cliente de código aberto inclui uma interface de linha de comando e uma biblioteca para gerenciar o Deadline Cloud. A funcionalidade inclui enviar pacotes de tarefas com base na especificação Open Job Description para o Deadline Cloud, baixar saídas de anexos de tarefas e monitorar sua fazenda usando a interface de linha de comando (CLI).

Aplicativo de criação de conteúdo digital (DCC)

Os aplicativos de criação de conteúdo digital (DCCs) são produtos de terceiros nos quais você cria conteúdo digital. O Deadline Cloud tem integrações integradas com muitos, DCCs como Autodesk Maya, Blender e Maxon Cinema 4D, permitindo que você envie trabalhos de dentro do DCC e renderize em frotas gerenciadas por serviços com software e licenciamento pré-configurados.

Anexos de trabalho

Os anexos de trabalho são um recurso do Deadline Cloud em que você carrega e baixa ativos como parte de um trabalho, como texturas, modelos 3D e equipamentos de iluminação. Os anexos do trabalho são armazenados no Amazon S3 e evitam a necessidade de armazenamento compartilhado em rede.

Modelo de trabalho

Um modelo de trabalho define o ambiente de execução e todos os processos que são executados como parte de um trabalho do Deadline Cloud.

Remetente do Deadline Cloud

Um remetente do Deadline Cloud é um plug-in para um DCC que permite que os usuários enviem trabalhos facilmente de dentro do DCC.

Endpoint de licença

Um endpoint de licença disponibiliza o licenciamento baseado no uso do Deadline Cloud para produtos de terceiros em sua VPC. Esse modelo é pago conforme o uso e você é cobrado pelo número de horas e minutos que usa. Os endpoints de licença não estão conectados às fazendas e podem ser usados de forma independente.

Tags

Uma tag é um rótulo que você pode atribuir a um AWS recurso. Cada tag consiste de uma chave e um valor opcional definido por você. Com as tags, você pode categorizar seus AWS recursos de maneiras diferentes, como por finalidade, proprietário ou ambiente.

Licenciamento baseado no uso (UBL)

O licenciamento baseado no uso (UBL) é um modelo de licenciamento sob demanda que está disponível para produtos selecionados de terceiros. Esse modelo é pago conforme o uso e você é cobrado pelo número de horas e minutos que usa.

Orientação sobre arquitetura de nuvem com prazo

Este tópico fornece orientação e melhores práticas para projetar e criar fazendas de renderização confiáveis, seguras, eficientes e econômicas para suas cargas de trabalho usando o Deadline Cloud. O uso dessa orientação pode ajudar você a criar workloads estáveis e eficientes, permitindo que você se concentre na inovação, reduza custos e melhore a experiência do cliente.

Esse conteúdo é destinado a diretores de tecnologia (CTOs), arquitetos, desenvolvedores e membros da equipe de operações.

Um fluxo de trabalho de end-to-end renderização requer soluções em várias camadas do processo, como geração de trabalhos, acesso a ativos e monitoramento de trabalhos. O Deadline Cloud oferece várias soluções para cada camada do processo de renderização. Ao selecionar as opções do Deadline Cloud em cada camada, você pode criar um fluxo de trabalho que corresponda ao seu caso de uso.

Para cada camada, você precisará decidir qual abordagem é melhor para seu caso de uso. Essas não são definições estritas de cenários e não são a única maneira de usar o Deadline Cloud. Em vez disso, esses são um conjunto de conceitos de alto nível para ajudar você a entender como o Deadline Cloud pode se encaixar em sua empresa ou fluxo de trabalho. Você pode separar as cargas de trabalho do Deadline Cloud nas seguintes camadas: Job Source, Job Submission, Application Management, Application Licensing, Asset Access, Output Management e Worker Infrastructure Management.

Em geral, você pode mix-and-match qualquer cenário em uma camada com qualquer outro cenário em outra camada, exceto combinações específicas que são especificadas abaixo.


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

Fonte de trabalho

A fonte do trabalho é o ponto de acesso em que novos trabalhos entrarão no sistema para serem renderizados pelo Deadline Cloud. Em um alto nível, existem duas fontes principais de empregos: interatividade humana e sistemas computacionais automatizados.

fluxo de trabalho interativo

Nesse cenário, um artista ou outra função criativa é o principal gerador de trabalho a ser processado na fazenda Deadline Cloud. Normalmente, a saída desses trabalhos é o principal artefato para um projeto ou equipe maior. Eles realizam seu trabalho usando software como uma ferramenta de criação de conteúdo digital (DCC) padrão do setor. Eles estão enviando trabalhos manualmente para a fazenda Deadline Cloud e, posteriormente, visualizando os resultados para análise. A estação de trabalho em si não é gerenciada pelo AWS.

Na maioria dos casos, esses artistas usam os remetentes integrados do Deadline Cloud e o monitor do Deadline Cloud nas camadas de aplicativo e monitoramento de carga de trabalho.

Fluxo de trabalho automático

Nesse cenário, um sistema programático de propriedade do cliente é o principal gerador de trabalhos na fazenda Deadline Cloud. Isso pode ser a geração de ativos em um funil de varejo, como um vídeo de mesa giratória gerado a partir de um modelo 3D ou digitalização. Isso pode ser a composição automatizada de gráficos de transmissão e placas de jogadores para esportes. O tema desse cenário é que um indivíduo não está enviando manualmente cada trabalho para o Deadline Cloud, mas, em vez disso, o trabalho é gerado como parte de um sistema maior.

Com trabalhos automatizados, é menos comum que os remetentes integrados do Deadline Cloud e o monitor do Deadline Cloud sejam usados. Freqüentemente, as definições de trabalho serão o desenvolvimento de aplicativos personalizados, escrito por você, e os resultados do trabalho fluirão automaticamente para um sistema de gerenciamento de ativos digitais (DAM) ou sistema de gerenciamento de ativos de mídia (MAM) para aprovação e distribuição.

Envio de emprego

Os trabalhos são enviados para o Deadline Cloud usando OpenJobDescriptionmodelos. OpenJobDescription é uma especificação aberta e flexível para definir trabalhos de processamento

em lote que são portáteis entre diferentes implantações do sistema de agendamento. O arquivo de definição de Job descreve os parâmetros do job, as etapas do job, como uma etapa é parametrizada com base nas entradas do job, bem como o script real que será executado em um Worker para realizar o processamento. A ideia do envio de carga de trabalho é como essas definições de trabalho são criadas, quem as cria e como elas são enviadas.

Remetente integrado com DCC

Um remetente integrado do Deadline Cloud é um software que une o Deadline Cloud a um DCC ou pacote de software padrão do setor. O remetente integrado determina como transformar os dados e a configuração de uma renderização, composição ou outra carga de trabalho em um modelo de trabalho, algo que pode ser entendido pelo Deadline Cloud. Muitos remetentes integrados são criados e mantidos pela equipe do Deadline Cloud ou pelo criador do pacote de software, mas se ainda não existir um para o aplicativo desejado, você pode criar e manter seu próprio remetente. Há um conjunto finito DCCs que é apoiado pela equipe do Deadline Cloud.

Os fluxos de trabalho interativos geralmente envolvem remetentes integrados, mas nem sempre. Para fluxos de trabalho automatizados e modelados, um fluxo de trabalho comum é que um artista configure um trabalho modelo em seu DCC e realize uma exportação única do pacote de trabalhos. Esse pacote de trabalhos define como executar esse tipo específico de trabalho no Deadline Cloud de forma parametrizada. Esse pacote de tarefas pode ser integrado ao cenário de fluxo de trabalho automatizado para fins de automação.

Definição de trabalho personalizada

Para aplicativos e fluxos de trabalho personalizados, é possível controlar totalmente como essas definições de trabalho são criadas e enviadas ao Deadline Cloud. Por exemplo, um site de comércio eletrônico pode pedir aos vendedores que enviem modelos 3D do objeto que estão vendendo. Após esse upload, a plataforma de comércio eletrônico poderia gerar dinamicamente uma definição de trabalho para enviar ao Deadline Cloud para gerar automaticamente uma animação de mesa giratória em um plano de fundo comum usando iluminação comum para combinar com os outros objetos 3D disponíveis no site. Durante o desenvolvimento da plataforma de comércio eletrônico, um desenvolvedor de software criaria uma definição de trabalho, a incorporaria à plataforma de comércio eletrônico com parâmetros eventualmente fornecidos pelos vendedores e codificaria a plataforma para enviar esse trabalho durante o fluxo de trabalho de upload do produto da plataforma.

O Deadline Cloud fornece vários exemplos de definições de tarefas no repositório de amostras no github.

Gerenciamento de aplicações

Depois que um trabalho é enviado ao Deadline Cloud e atribuído a um trabalhador, o script da definição do trabalho é executado no trabalhador. Na maioria dos casos, esse script invocará um aplicativo para realizar o processamento real, como renderizador, composição, codificação, filtragem ou qualquer outra tarefa de computação intensiva. O gerenciamento de aplicativos é o conceito de garantir que a versão necessária do software necessário esteja disponível para os trabalhadores.

Você pode gerenciar aplicativos usando qualquer sistema de gerenciamento de pacotes que desejar, mas o Deadline Cloud fornece várias ferramentas para permitir facilmente o uso de pacotes conda. O Conda é um gerenciador de pacotes e sistema de gerenciamento de ambiente de código aberto, multiplataforma e independente de linguagem.

Deadline Canal conda gerenciado em nuvem para frotas gerenciadas por serviços (SMF)

Ao usar frotas gerenciadas por serviços, um canal conda gerenciado pelo Deadline Cloud é automaticamente configurado e configurado para uso por seus trabalhos. O serviço Deadline Cloud fornece vários aplicativos e renderizações de DCC de parceiros neste canal conda. Para obter mais informações, consulte Criar um ambiente de fila no guia do usuário do Deadline Cloud. Esses pacotes são atualizados automaticamente pelo serviço Deadline Cloud e não exigem manutenção de você. Esse canal conda só está disponível ao usar frotas gerenciadas por serviços e não está disponível ao usar frotas gerenciadas pelo cliente.

Canal conda autogerenciado

Se você não conseguir usar o canal conda gerenciado pelo Deadline Cloud, você deve determinar como instalar, corrigir e gerenciar aplicativos em sua frota do Deadline Cloud. Uma opção é criar um canal conda que você configure e mantenha. Isso interoperará mais estreitamente com o canal conda gerenciado pelo Deadline Cloud. Por exemplo, você pode usar um DCC do canal conda gerenciado pelo Deadline Cloud, mas trazer seu próprio pacote que contém um plug-in DCC específico. Para obter mais informações sobre esse processo, consulte Criar um canal conda usando o S3.

Gerenciamento personalizado de aplicativos

Para o gerenciamento de aplicativos, o requisito do Deadline Cloud é que o aplicativo esteja disponível no PATH quando o script de trabalho for executado no trabalhador.

Se você já cria e mantém pacotes Rez, pode usar um ambiente de fila para instalar os aplicativos dos repositórios Rez. Um exemplo de ambiente de fila pode ser encontrado na GitHub organização AWS Deadline Cloud.

Se você já gerencia aplicativos em frotas gerenciadas pelo cliente com funcionários de longa vida ou em imagens do sistema, nenhum ambiente de fila é necessário para o gerenciamento de aplicativos. Certifique-se de que o aplicativo apareça no caminho do usuário do trabalho e envie o trabalho.

Licenciamento do aplicativo

Muitas cargas de trabalho normalmente executadas no Deadline Cloud exigem licenciamento de software do fornecedor do software. Esses aplicativos geralmente são licenciados por usuário, por CPU ou por host. É sua responsabilidade garantir que o uso de software de terceiros no Deadline Cloud cumpra o contrato de licenciamento de terceiros. Se você estiver usando software de código aberto, software personalizado ou software livre de licença, a configuração dessa camada não será necessária. Lembre-se de que o Deadline Cloud só oferece suporte ao licenciamento de renderização e não oferece suporte ao licenciamento de estações de trabalho.

Frotas gerenciadas por serviços e licenciamento baseado no uso

Ao usar frotas gerenciadas pelo serviço Deadline Cloud, o licenciamento baseado no uso (UBL) é configurado automaticamente para o software compatível. Os trabalhos executados em frotas gerenciadas por serviços têm automaticamente variáveis de ambiente definidas para aplicativos compatíveis para direcioná-los a usar os servidores de licenças do Deadline Cloud. Ao usar o Deadline Cloud UBL, você só é cobrado pelo número de horas de uso do aplicativo licenciado.

Frotas gerenciadas pelo cliente e licenciamento baseado no uso

O licenciamento baseado no uso (UBL) do Deadline Cloud também está disponível quando não se usa frotas gerenciadas por serviços. Nesse cenário, você configurará endpoints de licença do Deadline Cloud que fornecem endereços IP nas sub-redes VPC selecionadas que fornecem acesso aos servidores de licenças do Deadline Cloud. Depois de configurar as variáveis de ambiente específicas de software apropriadas em seus trabalhadores e configurar a conectividade de rede dos trabalhadores com esses endereços IP de endpoint de licença, os trabalhadores poderão fazer check-out e verificar as licenças do software compatível. Você é cobrado por hora pelas licenças da mesma forma que ao usar o UBL em frotas gerenciadas por serviços.

Licenciamento personalizado

Você pode usar um aplicativo que não é compatível com o Deadline Cloud UBL ou pode ter licenças preexistentes que ainda são válidas. Nesse cenário, você é responsável por configurar o caminho de rede dos seus trabalhadores (gerenciados pelo cliente ou pelo serviço) até os servidores de licenças. Para obter mais informações sobre licenciamento personalizado, consulteConecte frotas gerenciadas por serviços a um servidor de licenças personalizado.

Acesso a ativos

Depois que um trabalho é enviado a um trabalhador e o aplicativo é configurado, o trabalhador deve ser configurado para acessar os dados do ativo necessários para o trabalho. Podem ser dados 3D, dados de textura, dados de animação, quadros de vídeo ou qualquer outro tipo de dado usado em seu trabalho.

Comece pensando em onde seus dados estão armazenados atualmente. Isso pode estar no disco rígido da estação de trabalho, em uma ferramenta de colaboração do usuário, no controle de origem, em um sistema de arquivos compartilhado no local ou na nuvem, no Amazon S3 ou em qualquer outro local.

Em seguida, considere o que é necessário para que um trabalhador acesse esses dados. Esses dados são disponibilizados apenas na sua rede corporativa? Quais identidades ou credenciais são necessárias para acessar os dados? A fonte de dados é dimensionada para suportar o trabalho com o número de trabalhadores que você espera que processem o trabalho?

Anexos de trabalho

O mecanismo mais fácil de começar com acesso a ativos são os anexos de tarefas do Deadline Cloud. Quando um trabalho é enviado usando anexos do trabalho, os dados exigidos pelo trabalho são carregados em um bucket do Amazon S3 junto com um arquivo manifesto especificando quais arquivos o trabalho exige. Com os anexos de tarefas, nenhuma configuração complicada de rede ou armazenamento compartilhado é necessária. Os arquivos são enviados apenas uma vez, então os carregamentos subsequentes são concluídos mais rapidamente. Depois que um trabalhador termina de processar um trabalho, os dados de saída são enviados para o Amazon S3 para que possam ser baixados pelo artista ou por outro cliente. Os acessórios Job se adaptam a frotas de qualquer tamanho e são simples e rápidos de integrar e usar.

Job Attachments não é a melhor ferramenta para todas as situações. Se seus dados já estiverem ativos AWS, os anexos do trabalho adicionarão uma cópia adicional dos seus dados, incluindo o

tempo de transferência e os custos de armazenamento associados. Os anexos do trabalho exigem que o trabalho possa especificar totalmente os dados necessários no momento do envio, para que os dados possam ser carregados.

Para usar anexos de trabalho, sua fila do Deadline Cloud deve ter um intervalo de anexos de trabalho associado e a função de fila deve ser usada para fornecer acesso a esse intervalo. Por padrão, o Deadline Cloud integra todos os anexos de trabalho de suporte. Se você não estiver usando um remetente integrado do Deadline Cloud, os anexos de trabalho podem ser usados com seu software personalizado integrando a biblioteca python do Deadline Cloud.

Acesso personalizado ao armazenamento

Se você não usa anexos de trabalho, é responsável por garantir que os trabalhadores tenham acesso aos dados necessários para os trabalhos. O Deadline Cloud fornece várias ferramentas para apoiar isso e manter os trabalhos portáteis. Talvez você queira usar uma solução de armazenamento personalizada quando já tiver armazenamento em rede compartilhado para artistas e trabalhadores, preferir usar um serviço externo LucidLink, por exemplo.

Use perfis de armazenamento para modelar sistemas de arquivos em sua estação de trabalho e hosts de trabalho. Cada perfil de armazenamento descreve o sistema operacional e o layout do sistema de arquivos de uma das configurações do sistema. Usando perfis de armazenamento, quando um artista usando uma estação de trabalho Windows envia um trabalho que é processado por um Linux trabalhador, o Deadline Cloud garante que o mapeamento do caminho ocorra para que o trabalhador possa acessar o armazenamento de dados que você configurou.

Ao usar frotas gerenciadas por serviços do Deadline Cloud, scripts de configuração de host e endpoints de recursos de VPC permitem que os trabalhadores montem e acessem diretamente o armazenamento compartilhado ou outros serviços disponíveis em sua VPC.

Monitoramento de trabalhos e gerenciamento de resultados

Depois que os trabalhos enviados ao Deadline Cloud forem concluídos com sucesso, uma pessoa ou processo baixará a saída do trabalho para usar no fluxo de trabalho da empresa fora do Deadline Cloud. Após a falha no trabalho, os registros do trabalho e as informações de monitoramento ajudam a diagnosticar problemas.

Monitor de nuvem Deadline

O aplicativo de monitoramento Deadline Cloud está disponível na web e para desktop. Essa solução é mais adequada para estúdios que usam fluxos de trabalho interativos para uma ampla variedade de DCCs uso de anexos de trabalho para armazenamento. O monitor só oferece suporte para você ao usar o IAM Identity Center. O IAM Identity Center é um produto de identidade da força de trabalho, não uma solução de identidade do consumidor (B2C), portanto, não é apropriado para muitos cenários B2C.

Aplicativo de monitor personalizado

Se você deseja personalizar a experiência de monitoramento de seus usuários, está criando um produto B2C ou criando um sistema altamente especializado usando o Deadline Cloud, opte por criar um aplicativo de monitoramento personalizado. Você pode usar a API AWS Deadline Cloud para criar esse aplicativo personalizado, combinando o contexto do seu fluxo de trabalho geral com os conceitos do Deadline Cloud. Por exemplo, seu produto B2C pode ter seu próprio conceito de projeto configurado pelos usuários e seu aplicativo pode agrupar trabalhos do Deadline Cloud na mesma interface.

Solução de monitoramento automatizado

Em alguns cenários, nenhum aplicativo de monitoramento dedicado é necessário para o Deadline Cloud. Esse cenário é comum em fluxos de trabalho automatizados em que o Deadline Cloud é usado para renderizar automaticamente ativos em um pipeline, como gráficos de transmissão de esportes ou notícias. Nesse cenário, a API e os EventBridge eventos do Deadline Cloud são usados para se integrar a um sistema externo de gerenciamento de ativos de mídia para aprovações e transferência de dados para a próxima etapa do processo.

Gerenciamento da infraestrutura do trabalhador

As frotas do Deadline Cloud são um agrupamento de servidores (trabalhadores) capazes de processar trabalhos enviados para uma fila do Deadline Cloud e são a infraestrutura principal de qualquer farm do Deadline Cloud.

Frotas gerenciadas por serviços

Em uma frota gerenciada por serviços, o Deadline Cloud assume a responsabilidade pelos hosts, pelo sistema operacional, pela rede, pela aplicação de patches, pelo escalonamento automático e

por outros fatores da execução de uma fazenda de renderização. Você especifica o número mínimo e máximo de trabalhadores que deseja, junto com as especificações do sistema necessárias para seu aplicativo, e o Deadline Cloud faz o resto. As frotas gerenciadas por serviços são a única opção de frota que pode usar os canais conda gerenciados pelo Deadline Cloud para gerenciar facilmente os aplicativos de DCC do setor. Além disso, o Deadline Cloud UBL é configurado automaticamente com frotas gerenciadas por serviços. As frotas Wait and Save para cargas de trabalho mais baratas e tolerantes a atrasos só estão disponíveis usando frotas gerenciadas por serviços.

Frotas gerenciadas pelo cliente

Você usa frotas gerenciadas pelo cliente quando precisa de mais controle sobre os anfitriões dos trabalhadores e seu ambiente. As frotas gerenciadas pelo cliente são mais adequadas ao usar o Deadline Cloud no local. Para saber mais, consulte Crie e use frotas gerenciadas pelo cliente do Deadline Cloud.

Arquiteturas de exemplo

Estúdio de produção tradicional

O estúdio de produção tradicional exige uma infraestrutura significativa de computação, armazenamento e rede que possa abranger vários locais físicos para atender às cargas de trabalho de renderização. Cada pacote de software e fornecedor individual tem requisitos exclusivos de hardware, software, rede e licenciamento que devem ser atendidos ao resolver conflitos de versão, compatibilidade e recursos.

É comum ter requisitos de infraestrutura separados para estações de trabalho de artistas, nós de renderização, armazenamento em rede, servidores de licenças, sistemas de filas de tarefas, ferramentas de monitoramento e gerenciamento de ativos. Normalmente, os estúdios precisam manter várias versões de ferramentas de DCC, renderizadores, plug-ins e ferramentas personalizadas enquanto gerenciam acordos complexos de licenciamento em toda a fazenda de renderização. A infraestrutura do seu estúdio se torna mais complicada quando você considera os ambientes de desenvolvimento, garantia de qualidade e produção.

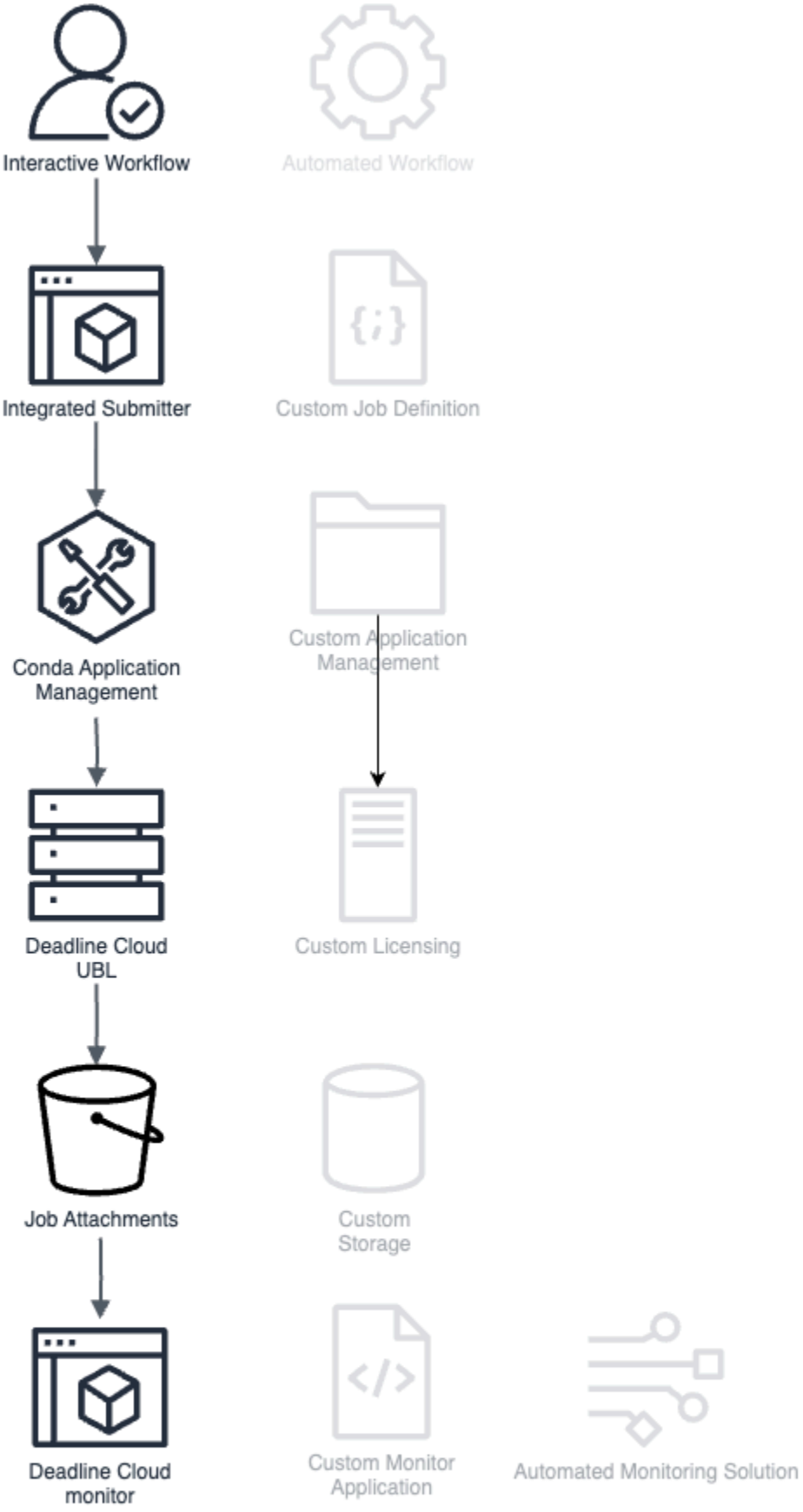
Uma implantação típica do Deadline Cloud usando opções gerenciadas por serviços resolve ou reduz muitos desses desafios por meio de:

- Fluxo de trabalho interativo: envio de trabalhos por meio de remetentes integrados do DCC
- Gerenciamento de aplicativos por meio de canais conda gerenciados pelo Deadline Cloud

- Licenciamento baseado em uso configurado automaticamente para o software compatível
- Gerenciamento de ativos por meio de anexos de tarefas
- Monitoramento por meio do aplicativo de monitoramento Deadline Cloud
- Gerenciamento de infraestrutura por meio de frotas gerenciadas por serviços

Com essa abordagem, os artistas podem enviar trabalhos diretamente de suas ferramentas familiares de DCC para uma fazenda de renderização em nuvem escalável sem gerenciar uma infraestrutura complexa. O serviço gerencia automaticamente a implantação de software, o licenciamento, a transferência de dados e o dimensionamento da infraestrutura. Os artistas podem monitorar seus trabalhos por meio de uma interface web ou aplicativo de desktop, e os resultados são armazenados automaticamente no Amazon S3 para facilitar o acesso.

Com essa configuração, os estúdios podem criar ambientes de desenvolvimento e produção em minutos, pagar apenas pela computação e pelo licenciamento que usam e se concentrar no trabalho criativo em vez do gerenciamento da infraestrutura. A abordagem gerenciada por serviços fornece o caminho mais rápido para adotar a renderização na nuvem e, ao mesmo tempo, manter fluxos de trabalho familiares para 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

Estúdio na nuvem

Os estúdios modernos de efeitos visuais e animação estão cada vez mais migrando todo o seu pipeline para a nuvem, incluindo estações de trabalho de artistas. Essa abordagem elimina a necessidade de infraestrutura local, permite a colaboração global e fornece escalabilidade perfeita para trabalho interativo e renderização. No entanto, ele também apresenta novos desafios no gerenciamento de recursos em nuvem, na garantia de acesso de baixa latência aos dados e na integração de estações de trabalho baseadas em nuvem com fazendas de renderização.

Um estúdio nativo da nuvem típico exige uma abordagem unificada para gerenciar estações de trabalho em nuvem, armazenamento compartilhado, infraestrutura de renderização e implantação de software em todos esses componentes. As abordagens tradicionais geralmente resultavam em sistemas complexos, gerenciados manualmente, que tinham dificuldade em equilibrar desempenho, custo e flexibilidade.

Uma implantação do Deadline Cloud para um estúdio nativo em nuvem pode ser implementada usando:

- Envio de tarefas de fluxo de trabalho interativo por meio de remetentes de DCC integrados em estações de trabalho em nuvem
- Gerenciamento de aplicativos por meio de nós de renderização de canais conda gerenciados pela Deadline Cloud
- Licenciamento baseado em uso configurado automaticamente para o software compatível
- Acesso personalizado ao armazenamento usando o FSx Windows File Server para dados compartilhados do projeto
- Monitoramento por meio do aplicativo de monitoramento Deadline Cloud
- Gerenciamento de infraestrutura usando frotas gerenciadas por serviços

Essa abordagem permite que os artistas trabalhem em estações de trabalho baseadas em nuvem com acesso direto ao armazenamento compartilhado de alto desempenho e enviem trabalhos sem problemas para a fazenda Deadline Cloud. O estúdio pode gerenciar a implantação de software em estações de trabalho e nós de renderização usando os mesmos canais conda, garantindo consistência e reduzindo a sobrecarga de manutenção.

Os principais benefícios dessa configuração incluem:

- Colaboração global com artistas capazes de acessar estações de trabalho de qualquer lugar

- Ambientes de software consistentes em estações de trabalho e nós de renderização
- Armazenamento compartilhado de alto desempenho acessível tanto para estações de trabalho quanto para nós de renderização
- Dimensionamento flexível de recursos de computação interativos e em lote
- Gerenciamento centralizado de toda a infraestrutura do estúdio na nuvem

A configuração de armazenamento nesse cenário geralmente envolve:

- FSx para servidor de Windows arquivos para dados de projetos, acessíveis tanto por estações de trabalho em nuvem quanto por funcionários do Deadline Cloud
- Perfis de armazenamento no Deadline Cloud para gerenciar o mapeamento de caminhos entre estações de trabalho e nós de renderização
- Montagem direta de FSx compartilhamentos em trabalhadores do Deadline Cloud usando endpoints de recursos de VPC e scripts de configuração de host

Essa abordagem nativa da nuvem permite que os estúdios eliminem a infraestrutura local, permitindo um rápido escalonamento para projetos de qualquer tamanho, mantendo fluxos de trabalho familiares para artistas. Ele fornece a flexibilidade de usar uma combinação de recursos gerenciados por serviços e gerenciados pelo cliente, otimizando tanto a facilidade de gerenciamento quanto os requisitos específicos de desempenho.

Ao aproveitar as estações de trabalho em nuvem junto com o Deadline Cloud, os estúdios podem obter um pipeline de produção totalmente integrado e acessível globalmente que se expande perfeitamente de pequenas equipes a grandes produções.

ECommerce Automation

A plataforma moderna de comércio eletrônico exige a geração automatizada de ativos em grande escala para fornecer uma visualização avançada do produto em milhões de itens. As abordagens tradicionais exigiriam um investimento significativo em infraestrutura para processar grandes volumes de modelos 3D em mídia de produto padronizada, geralmente resultando em sistemas subprovisionados que criam atrasos de processamento ou sistemas superprovisionados com capacidade ociosa.

Um fluxo de trabalho de comércio eletrônico automatizado típico precisa lidar com o processamento de upload de produtos, validação de modelos 3D, gerenciamento de fazendas de renderização,

processamento de resultados e integração com sistemas de informações de produtos. O gerenciamento desses fluxos de trabalho tradicionalmente exige a coordenação de vários aplicativos de renderização, recursos computacionais e pipelines de processamento de dados, garantindo qualidade consistente e mantendo a eficiência de custos em grande escala.

Uma implantação do Deadline Cloud para automação de comércio eletrônico pode ser implementada usando:

- Envio automatizado de tarefas de fluxo de trabalho por meio da integração personalizada da API no aplicativo de ingestão de comércio eletrônico existente
- Definições de tarefas personalizadas adaptadas à visualização padronizada do produto
- Gerenciamento de aplicativos por meio de canais conda gerenciados pelo Deadline Cloud
- Licenciamento baseado em uso configurado automaticamente para o software compatível
- Integração direta com o Amazon S3 para gerenciamento de ativos
- Aplicativo de monitoramento personalizado integrado aos sistemas de gerenciamento de produtos existentes
- Frotas gerenciadas por serviços para escalabilidade elástica

Essa abordagem permite o processamento de milhares de produtos por dia, gerando automaticamente visualizações padronizadas de produtos, como animações de toca-discos. A infraestrutura gerenciada por serviços é escalada automaticamente para atender à demanda variável e, ao mesmo tempo, manter a eficiência de custos por meio da reutilização dos funcionários e da implantação otimizada de aplicativos.


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

O software tradicional de criação de conteúdo digital (DCC) normalmente exige que os usuários mantenham sua própria infraestrutura de renderização ou processem renderizações localmente em sua estação de trabalho, resultando em investimentos significativos em hardware ou em longos tempos de espera que interrompem os fluxos de trabalho criativos. Para fornecedores de software, fornecer recursos de renderização em nuvem tradicionalmente exigia a criação e a manutenção de sistemas complexos de infraestrutura e cobrança.

Uma implantação do Deadline Cloud integrada ao software B2C permite uma renderização perfeita na nuvem diretamente na interface familiar do usuário. Essa integração combina:

- Envio de tarefas de fluxo de trabalho interativo incorporado ao aplicativo DCC
- Prazo: canais conda gerenciados em nuvem para implantação de aplicativos de renderização
- Licenciamento baseado em uso configurado automaticamente
- Gerenciamento de ativos por meio de anexos de tarefas com armazenamento gerenciado pelo fornecedor
- Monitoramento personalizado integrado diretamente na interface do DCC
- Frotas gerenciadas por serviços compartilhadas entre usuários

Essa abordagem permite que os usuários finais enviem renderizações para a nuvem com um único clique de dentro do software, sem gerenciar contas, infraestrutura ou configurações complexas. O fornecedor do software mantém um ambiente multilocatário em que:

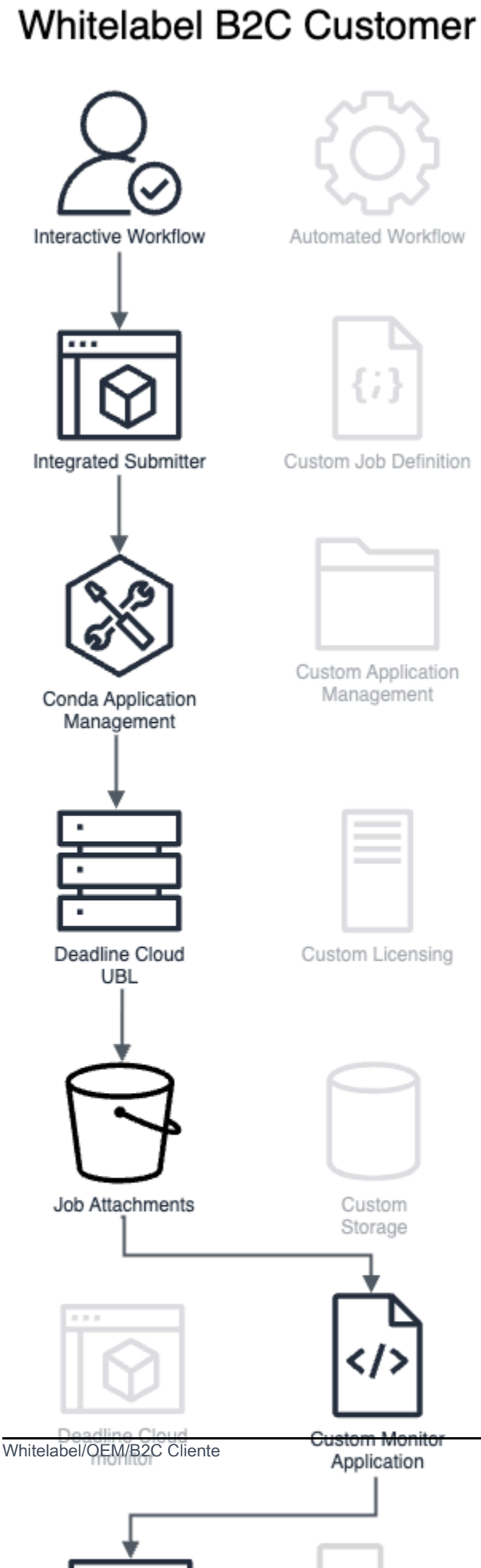
- Os usuários se autenticam por meio de suas credenciais de software existentes
- Os trabalhos são encaminhados automaticamente para filas dedicadas por usuário
- Os ativos são isolados com segurança usando prefixos de armazenamento controlados pelo IAM
- O faturamento é feito por meio dos sistemas existentes do fornecedor
- O status e as saídas do trabalho são transmitidos diretamente para o aplicativo do usuário

A abordagem de frota compartilhada garante um desempenho ideal ao manter um grupo aquecido de trabalhadores, minimizando os tempos de inicialização e maximizando a utilização de recursos em toda a base de usuários. Essa configuração permite que os fornecedores de software ofereçam renderização em nuvem como um recurso de produto perfeito, em vez de um serviço separado que exige configurações ou contas adicionais.

Os usuários finais se beneficiam de:

- Envio com um clique a partir de sua interface familiar
- Pay-as-you-go preços sem gerenciamento de infraestrutura
- Tempos rápidos de inicialização de trabalhos por meio de infraestrutura compartilhada
- Download automático e organização de renderizações concluídas
- Experiência consistente em todas as plataformas

Esse padrão de integração permite que os fornecedores de software forneçam recursos de renderização de nível corporativo para toda a sua base de usuários, mantendo uma experiência simples e amigável ao consumidor que parece nativa de seu aplicativo.


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

O que é uma carga de trabalho do Deadline Cloud

Com AWS o Deadline Cloud, você pode enviar trabalhos para executar seus aplicativos na nuvem e processar dados para a produção de conteúdo ou insights cruciais para seus negócios. O Deadline Cloud usa o Open Job Description (OpenJD) como sintaxe para modelos de trabalho, uma especificação projetada para as necessidades dos pipelines de computação visual, mas aplicável a muitos outros casos de uso. Alguns exemplos de cargas de trabalho incluem renderização de computação gráfica, simulação física e fotogrametria.

As cargas de trabalho variam de pacotes de tarefas simples que os usuários enviam para uma fila com a CLI ou uma GUI gerada automaticamente até plug-ins de envio integrados que geram dinamicamente um pacote de tarefas para uma carga de trabalho definida pelo aplicativo.

Como as cargas de trabalho surgem da produção

Para entender as cargas de trabalho em contextos de produção e como apoiá-las com o Deadline Cloud, considere como elas surgiram. A produção pode envolver a criação de efeitos visuais, animações, jogos, imagens de catálogos de produtos, reconstruções 3D para modelagem de informações de construção (BIM) e muito mais. Esse conteúdo geralmente é criado por uma equipe de especialistas artísticos ou técnicos que executam uma variedade de aplicativos de software e scripts personalizados. Os membros da equipe passam dados entre si usando um pipeline de produção. Muitas tarefas executadas pelo pipeline envolvem cálculos intensivos que levariam dias se executados na estação de trabalho de um usuário.

Alguns exemplos de tarefas nesses pipelines de produção incluem:

- Usando um aplicativo de fotogrametria para processar fotografias tiradas de um set de filmagem para reconstruir uma malha digital texturizada.
- Executando uma simulação de partículas em uma cena 3D para adicionar camadas de detalhes a um efeito visual de explosão para um programa de televisão.
- Reunindo dados de um nível de jogo no formato necessário para lançamento externo e aplicando configurações de otimização e compressão.
- Renderização de um conjunto de imagens para um catálogo de produtos, incluindo variações de cor, plano de fundo e iluminação.
- Executar um script desenvolvido sob medida em um modelo 3D para aplicar uma aparência personalizada e aprovada por um diretor de cinema.

Essas tarefas envolvem muitos parâmetros a serem ajustados para obter um resultado artístico ou para ajustar a qualidade da saída. Geralmente, há uma GUI para selecionar esses valores de parâmetros com um botão ou menu para executar o processo localmente dentro do aplicativo. Quando um usuário executa o processo, o aplicativo e possivelmente o próprio computador host não podem ser usados para realizar outras operações porque ele usa o estado do aplicativo na memória e pode consumir todos os recursos de CPU e memória do computador host.

Em muitos casos, o processo é rápido. Durante o curso da produção, a velocidade do processo diminui quando os requisitos de qualidade e complexidade aumentam. Um teste de personagem que levou 30 segundos durante o desenvolvimento pode facilmente se transformar em 3 horas quando aplicado ao personagem final da produção. Por meio dessa progressão, uma carga de trabalho que começou dentro de uma GUI pode crescer demais para caber. Transportá-lo para o Deadline Cloud pode aumentar a produtividade dos usuários que executam esses processos, pois eles recuperam o controle total da estação de trabalho e podem acompanhar mais iterações a partir do monitor do Deadline Cloud.

Há dois níveis de suporte a serem buscados ao desenvolver suporte para uma carga de trabalho no Deadline Cloud:

- Transferir a carga de trabalho da estação de trabalho do usuário para um farm do Deadline Cloud sem paralelismo ou aceleração. Isso pode subutilizar os recursos computacionais disponíveis na fazenda, mas a capacidade de transferir operações longas para um sistema de processamento em lote permite que os usuários façam mais com sua própria estação de trabalho.
- Otimizando o paralelismo da carga de trabalho para que ela utilize a escala horizontal da fazenda Deadline Cloud para concluir rapidamente.

Às vezes, é óbvio como fazer uma carga de trabalho ser executada paralelamente. Por exemplo, cada quadro de uma renderização de computação gráfica pode ser feito de forma independente. No entanto, é importante não ficar preso a esse paralelismo. Em vez disso, entenda que transferir uma carga de trabalho de longa duração para o Deadline Cloud oferece benefícios significativos, mesmo quando não há uma maneira óbvia de dividir a carga de trabalho.

Os ingredientes de uma carga de trabalho

Para especificar uma carga de trabalho do Deadline Cloud, implemente um pacote de tarefas que os usuários enviem para uma fila com a CLI do Deadline Cloud. Grande parte do trabalho na criação de um pacote de tarefas consiste em escrever o modelo de trabalho, mas há mais fatores, como a forma

de fornecer os aplicativos que a carga de trabalho exige. Aqui estão as coisas essenciais a serem consideradas ao definir uma carga de trabalho para o Deadline Cloud:

- O aplicativo a ser executado. O trabalho deve ser capaz de iniciar processos de aplicativos e, portanto, precisa de uma instalação do aplicativo disponível, bem como de qualquer licenciamento usado pelo aplicativo, como acesso a um servidor de licenças flutuante. Normalmente, isso faz parte da configuração do farm e não está incorporado no próprio pacote de tarefas.
  - Configurar trabalhos usando ambientes de fila
  - Conecte frotas gerenciadas pelo cliente a um endpoint de licença
- Definições de parâmetros de Job. A experiência do usuário ao enviar o trabalho é muito afetada pelos parâmetros que ele fornece. Os parâmetros de exemplo incluem arquivos de dados, diretórios e configuração do aplicativo.
  - Elementos de valores de parâmetros para pacotes de tarefas
- Fluxo de dados do arquivo. Quando um trabalho é executado, ele lê a entrada dos arquivos fornecidos pelo usuário e, em seguida, grava a saída como novos arquivos. Para trabalhar com os anexos da tarefa e os recursos de mapeamento de caminhos, a tarefa deve especificar os caminhos dos diretórios ou arquivos específicos para essas entradas e saídas.
  - Usando arquivos em seus trabalhos
- O script da etapa. O script da etapa executa o binário do aplicativo com as opções de linha de comando corretas para aplicar os parâmetros de trabalho fornecidos. Ele também lida com detalhes, como mapeamento de caminhos, se os arquivos de dados da carga de trabalho incluírem referências de caminho absolutas em vez de relativas.
  - Elementos do modelo de trabalho para pacotes de tarefas

Portabilidade da carga de trabalho

Uma carga de trabalho é portátil quando pode ser executada em vários sistemas diferentes sem alterá-la sempre que você envia um trabalho. Por exemplo, ele pode ser executado em diferentes fazendas de renderização que tenham diferentes sistemas de arquivos compartilhados montados ou em sistemas operacionais diferentes, como Linux ouWindows. Quando você implementa um pacote de tarefas portátil, é mais fácil para os usuários executarem o trabalho em sua fazenda específica ou adaptá-lo para outros casos de uso.

Aqui estão algumas maneiras de tornar seu pacote de tarefas portátil.

- Especifique totalmente os arquivos de dados de entrada necessários para uma carga de trabalho, usando parâmetros de PATH trabalho e referências de ativos no pacote de tarefas.

Essa abordagem torna o trabalho portátil para fazendas baseadas em sistemas de arquivos compartilhados e para fazendas que fazem cópias dos dados de entrada, como o recurso de anexos de tarefas do Deadline Cloud.

- Torne as referências de caminho de arquivo para os arquivos de entrada da tarefa realocáveis e utilizáveis em diferentes sistemas operacionais. Por exemplo, quando os usuários enviam trabalhos de Windows estações de trabalho para serem executados em uma Linux frota.
  - Use referências relativas ao caminho do arquivo, portanto, se o diretório que as contém for movido para um local diferente, as referências ainda serão resolvidas. Alguns aplicativos, como o Blender, oferecem suporte à escolha entre caminhos relativos e absolutos.
  - Se você não puder usar caminhos relativos, ofereça suporte aos metadados de mapeamento de caminhos do OpenJD e traduza os caminhos absolutos de acordo com a forma como o Deadline Cloud fornece os arquivos para o trabalho.
- Implemente comandos em um trabalho usando scripts portáteis. Python e bash são dois exemplos de linguagens de script que podem ser usadas dessa forma. Você deve considerar fornecer os dois em todos os trabalhadores anfitriões de suas frotas.
  - Use o binário do interpretador de script, como `python` ou`bash`, com o nome do arquivo de script como argumento. Essa abordagem funciona em todos os sistemas operacionaisWindows, inclusive, em comparação com o uso de um arquivo de script com o bit de execução ativadoLinux.
  - Escreva scripts bash portáteis aplicando estas práticas:
    - Expanda os parâmetros do caminho do modelo entre aspas simples para lidar com caminhos com espaços e separadores de Windows caminho.
    - Ao executarWindows, observe os problemas relacionados à tradução automática de caminhos do MinGW. Por exemplo, ele transforma um AWS CLI comando like `aws logs tail /aws/deadline/...` em um comando semelhante a um log `aws logs tail "C:/Program Files/Git/aws/deadline/..."` e não envia corretamente. Defina `MSYS_NO_PATHCONV=1` a variável para desativar esse comportamento.
    - Na maioria dos casos, o mesmo código funciona em todos os sistemas operacionais. Quando o código precisar ser diferente, use uma `if/else` construção para lidar com os 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
```

  - Você pode escrever scripts Python portáteis usando `pathlib` para lidar com diferenças de caminho do sistema de arquivos e evitar recursos operacionais específicos. A documentação do Python inclui anotações para isso, por exemplo, na documentação da biblioteca de sinais. Linux-o suporte a recursos específicos está marcado como “Disponibilidade: Linux”.
- Use os parâmetros do trabalho para especificar os requisitos do aplicativo. Use convenções consistentes que o administrador da fazenda possa aplicar em ambientes de fila.
  - Por exemplo, você pode usar os `RezPackages` parâmetros `CondaPackages` e/ou em seu trabalho, com um valor de parâmetro padrão que lista os nomes dos pacotes de aplicativos e as versões que o trabalho exige. Em seguida, você pode usar um dos exemplos de ambientes de fila conda ou Rez para fornecer um ambiente virtual para o trabalho.

Introdução aos recursos do Deadline Cloud

Para começar a criar soluções personalizadas para AWS o Deadline Cloud, você deve configurar seus recursos. Isso inclui uma fazenda, pelo menos uma fila para a fazenda e pelo menos uma frota de trabalhadores para atender a fila. Você pode criar seus recursos usando o console do Deadline Cloud, ou você pode usar AWS Command Line Interface o.

Neste tutorial, você usará AWS CloudShell para criar um farm de desenvolvedores simples e executar o agente de trabalho. Em seguida, você pode enviar e executar um trabalho simples com parâmetros e anexos, adicionar uma frota gerenciada por serviços e limpar os recursos da fazenda quando terminar.

As seções a seguir apresentam os diferentes recursos do Deadline Cloud e como eles funcionam e funcionam juntos. Seguir essas etapas é útil para desenvolver e testar novas cargas de trabalho e personalizações.

Para obter instruções sobre como configurar sua fazenda usando o console, consulte Introdução no Guia do usuário do Deadline Cloud.

Tópicos

Crie um Deadline Cloud Farm

Para criar seu farm de desenvolvedores e enfileirar recursos no AWS Deadline Cloud, use o AWS Command Line Interface (AWS CLI), conforme mostrado no procedimento a seguir. Você também criará uma função AWS Identity and Access Management (IAM) e uma frota gerenciada pelo cliente (CMF) e associará a frota à sua fila. Em seguida, você pode configurar AWS CLI e confirmar se sua fazenda está configurada e funcionando conforme especificado.

Você pode usar essa fazenda para explorar os recursos do Deadline Cloud e, em seguida, desenvolver e testar novas cargas de trabalho, personalizações e integrações de pipeline.

Para criar uma fazenda

1. Abra uma AWS CloudShell sessão. Você usará a CloudShell janela para inserir os comandos AWS Command Line Interface (AWS CLI) para executar os exemplos deste tutorial. Mantenha a CloudShell janela aberta à medida que avança.
2. Crie um nome para sua fazenda e adicione esse nome à`~/.bashrc`. Isso o disponibilizará para outras sessões do terminal.

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

3. Crie o recurso da fazenda e adicione seu ID da fazenda `~/.bashrc` a.

```
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. Crie o recurso de fila e adicione seu ID de fila ao `~/.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. Crie uma função do IAM para a frota. Essa função fornece aos anfitriões de trabalhadores em sua frota as credenciais de segurança necessárias para executar trabalhos em sua fila.

```
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. Crie a frota gerenciada pelo cliente (CMF) e adicione sua ID de frota 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. Associe o CMF à sua fila.

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

8. Instale a interface de linha de comando do Deadline Cloud.

```
pip install deadline
```

9. Para definir a fazenda padrão como a ID da fazenda e a fila como a ID da fila que você criou anteriormente, use o comando a seguir.

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

10. (Opcional) Para confirmar se sua fazenda está configurada de acordo com suas especificações, use os seguintes comandos:

    - Listar todas as fazendas — **`deadline farm list`**
    - Listar todas as filas na fazenda padrão — **`deadline queue list`**
    - Listar todas as frotas na fazenda padrão — **`deadline fleet list`**
    - Obtenha a fazenda padrão — **`deadline farm get`**
    - Obtenha a fila padrão — **`deadline queue get`**
    - Obtenha todas as frotas associadas à fila padrão — **`deadline fleet get`**

Próximas etapas

Depois de criar sua fazenda, você pode executar o agente Deadline Cloud Worker nos hosts da sua frota para processar trabalhos. Consulte Execute o agente Deadline Cloud Worker.

Execute o agente Deadline Cloud Worker

Antes de executar os trabalhos enviados para a fila em seu farm de desenvolvedores, você deve executar o agente de trabalho do AWS Deadline Cloud no modo desenvolvedor em um host de trabalho.

Durante o restante deste tutorial, você executará AWS CLI operações em seu farm de desenvolvedores usando duas AWS CloudShell guias. Na primeira guia, você pode enviar trabalhos. Na segunda guia, você pode executar o agente de trabalho.

> **Note**
>
> Se você deixar sua CloudShell sessão ociosa por mais de 20 minutos, o tempo limite será atingido e o agente de trabalho será interrompido. Para reiniciar o agente de trabalho, siga as instruções no procedimento a seguir.

Antes de iniciar um agente de trabalho, você deve configurar uma fazenda, uma fila e uma frota do Deadline Cloud. Consulte Crie um Deadline Cloud Farm.

Para executar o agente de trabalho no modo de desenvolvedor

1. Com sua fazenda ainda aberta na primeira CloudShell guia, abra uma segunda CloudShell guia `demoenv-logs` e crie os `demoenv-persist` diretórios e.

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

2. Baixe e instale os pacotes do agente Deadline Cloud Worker do PyPI:

   > **Note**
   >
   > Ativado Windows, é necessário que os arquivos do agente sejam instalados no diretório global de pacotes de sites do Python. Atualmente, não há suporte para ambientes virtuais Python.

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

3. Para permitir que o agente de trabalho crie os diretórios temporários para execução de trabalhos, crie um diretório:

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

4. Execute o agente Deadline Cloud Worker no modo de desenvolvedor com as variáveis `DEV_FARM_ID` e `DEV_CMF_ID` que você adicionou ao`~/.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
```

Conforme o agente de trabalho inicializa e, em seguida, pesquisa a operação da `UpdateWorkerSchedule` API, a seguinte saída é exibida:

```
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. Selecione sua primeira CloudShell guia e, em seguida, liste os trabalhadores da frota.

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

Uma saída como a seguinte é exibida:

```
Displaying 1 of 1 workers starting at 0

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

Em uma configuração de produção, o agente do Deadline Cloud Worker exige a configuração de vários usuários e diretórios de configuração como usuário administrativo na máquina host. Você pode substituir essas configurações porque está executando trabalhos em sua própria fazenda de desenvolvimento, que somente você pode acessar.

Próximas etapas

Agora que um agente de trabalho está sendo executado em seus hosts de trabalhadores, você pode enviar trabalhos para seus trabalhadores. É possível:

- Envie com o Deadline Cloudusando um pacote de tarefas simples do OpenJD.
- Envie vagas com anexos de vagas no Deadline Cloudque compartilham arquivos entre estações de trabalho usando sistemas operacionais diferentes.

Envie com o Deadline Cloud

Para executar trabalhos do Deadline Cloud em seus hosts de trabalho, você cria e usa um pacote de trabalhos do Open Job Description (OpenJD) para configurar um trabalho. O pacote configura o trabalho, por exemplo, especificando arquivos de entrada para um trabalho e onde gravar a saída do trabalho. Este tópico inclui exemplos de maneiras pelas quais você pode configurar um pacote de tarefas.

Antes de seguir os procedimentos desta seção, você deve concluir o seguinte:

- Crie um Deadline Cloud Farm
- Execute o agente Deadline Cloud Worker

Para usar o AWS Deadline Cloud para executar trabalhos, use os procedimentos a seguir. Use a primeira AWS CloudShell guia para enviar trabalhos para sua fazenda de desenvolvedores. Use a segunda CloudShell guia para visualizar a saída do agente de trabalho.

Tópicos

Envie a simple_job amostra

Depois de criar uma fazenda e executar o agente de trabalho, você pode enviar a simple_job amostra para o Deadline Cloud.

Para enviar a simple_job amostra para o Deadline Cloud

1. Escolha sua primeira CloudShell guia.
2. Faça o download da amostra em GitHub.

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

3. Navegue até o diretório de amostras do pacote de tarefas.

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

4. Envie a simple_job amostra.

```
deadline bundle submit simple_job
```

5. Escolha sua segunda CloudShell guia para ver a saída de registro sobre chamadas`BatchGetJobEntities`, obtenção de uma sessão e execução de uma ação de sessão.

```
...
[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**
>
> Somente a saída de registro do agente de trabalho é mostrada. Há um registro separado para a sessão que executa o trabalho.

6. Escolha sua primeira guia e, em seguida, inspecione os arquivos de log que o agente de trabalho grava.

   a. Navegue até o diretório de registros do agente de trabalho e visualize seu conteúdo.

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

   b. Imprima o primeiro arquivo de log criado pelo agente de trabalho.

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

   Esse arquivo contém a saída do agente de trabalho sobre como ele chamou a API Deadline Cloud para criar um recurso de trabalhador em sua frota e, em seguida, assumiu a função de frota.

   c. Imprima a saída do arquivo de log quando o agente de trabalho se junta à frota.

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

   Esse registro contém saídas sobre todas as ações que o agente de trabalho executa, mas não contém saídas sobre as filas a partir das quais ele executa trabalhos, exceto esses IDs recursos.

   d. Imprima os arquivos de log de cada sessão em um diretório com o mesmo nome do ID do recurso da fila.

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

   Se o trabalho for bem-sucedido, a saída do arquivo de log será semelhante à seguinte:

   ```
   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 informações sobre o trabalho.

   ```
   deadline job get
   ```

   Quando você envia o trabalho, o sistema o salva como padrão para que você não precise inserir o ID do trabalho.

Envie um simple_job com um parâmetro

Você pode enviar trabalhos com parâmetros. No procedimento a seguir, você edita o simple_job modelo para incluir uma mensagem personalizada, envia a simple_job e imprime o arquivo de log da sessão para visualizar a mensagem.

Para enviar a simple_job amostra com um parâmetro

1. Selecione sua primeira CloudShell guia e, em seguida, navegue até o diretório de amostras do pacote de tarefas.

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

2. Imprima o conteúdo do simple_job modelo.

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

A `parameterDefinitions` seção com o `Message` parâmetro deve ter a seguinte aparência:

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

3. Envie a simple_job amostra com um valor de parâmetro e aguarde a conclusão da execução do trabalho.

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

4. Para ver a mensagem personalizada, veja o arquivo de registro da sessão mais recente.

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

Crie um pacote de tarefas simple_file_job com E/S de arquivo

Um trabalho de renderização precisa ler a definição da cena, renderizar uma imagem a partir dela e depois salvar essa imagem em um arquivo de saída. Você pode simular essa ação fazendo com que o trabalho calcule o hash da entrada em vez de renderizar uma imagem.

Para criar um pacote de tarefas simple_file_job com E/S de arquivo

1. Selecione sua primeira CloudShell guia e, em seguida, navegue até o diretório de amostras do pacote de tarefas.

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

2. Faça uma cópia do `simple_job` com o novo nome`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Edite o modelo de trabalho da seguinte forma:

> **Note**
>
> Recomendamos que você use nano essas etapas. Se você preferir usarVim, defina o modo de colagem usando`:set paste`.

a. Abra o modelo em um editor de texto.

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

b. Adicione o seguinte `typeobjectType`, `dataFlow` `parameterDefinitions` e.

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

c. Adicione o seguinte comando de `bash` script ao final do arquivo que lê o arquivo de entrada e grava no arquivo de saída.

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

A atualização `template.yaml` deve corresponder exatamente ao seguinte:

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

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

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

> **Note**
>
> Se você quiser ajustar o espaçamento `notemplate.yaml`, certifique-se de usar espaços em vez de recuos.

d. Salve o arquivo e saia do editor de texto.

4. Forneça valores de parâmetros para os arquivos de entrada e saída para enviar o simple_file_job.

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

5. Imprima informações sobre o trabalho.

```
deadline job get
```

- Você verá uma saída como a seguinte:

```
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
```

- Embora você tenha fornecido somente caminhos relativos, os parâmetros têm o caminho completo definido. O AWS CLI une o diretório de trabalho atual a todos os caminhos fornecidos como parâmetros quando os caminhos têm o tipo`PATH`.
- O agente de trabalho em execução na outra janela do terminal pega e executa o trabalho. Essa ação cria o `hash.txt` arquivo, que você pode visualizar com o comando a seguir.

  ```
  cat hash.txt
  ```

  Esse comando imprimirá uma saída semelhante à seguinte.

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

Próximas etapas

Depois de aprender a enviar trabalhos simples usando a CLI do Deadline Cloud, você pode explorar:

- Envie vagas com anexos de vagas no Deadline Cloudpara aprender a executar trabalhos em hosts que executam sistemas operacionais diferentes.
- Adicione uma frota gerenciada por serviços à sua fazenda de desenvolvedores no Deadline Cloudpara executar seus trabalhos em hosts gerenciados pelo Deadline Cloud.
- Limpe os recursos da sua fazenda no Deadline Cloudpara encerrar os recursos que você usou neste tutorial.

Envie vagas com anexos de vagas no Deadline Cloud

Muitas fazendas usam sistemas de arquivos compartilhados para compartilhar arquivos entre os hosts que enviam trabalhos e aqueles que executam trabalhos. Por exemplo, no `simple_file_job` exemplo anterior, o sistema de arquivos local é compartilhado entre as janelas do AWS CloudShell

terminal, que são executadas na guia um, na qual você envia o trabalho, e na guia dois, na qual você executa o agente de trabalho.

Um sistema de arquivos compartilhado é vantajoso quando a estação de trabalho remetente e os hosts do trabalhador estão na mesma rede local. Se você armazena seus dados no local próximo às estações de trabalho que os acessam, usar um farm baseado em nuvem significa que você precisa compartilhar seus sistemas de arquivos por meio de uma VPN de alta latência ou sincronizar seus sistemas de arquivos na nuvem. Nenhuma dessas opções é fácil de configurar ou operar.

AWS O Deadline Cloud oferece uma solução simples com anexos de trabalho, que são semelhantes aos anexos de e-mail. Com os anexos do trabalho, você anexa dados ao seu trabalho. Em seguida, o Deadline Cloud trata dos detalhes da transferência e armazenamento dos dados do seu trabalho nos buckets do Amazon Simple Storage Service (Amazon S3).

Os fluxos de trabalho de criação de conteúdo geralmente são iterativos, o que significa que um usuário envia trabalhos com um pequeno subconjunto de arquivos modificados. Como os buckets do Amazon S3 armazenam anexos de trabalho em um armazenamento endereçável por conteúdo, o nome de cada objeto é baseado no hash dos dados do objeto e o conteúdo de uma árvore de diretórios é armazenado em um formato de arquivo manifesto anexado a um trabalho.

Antes de seguir os procedimentos desta seção, você deve concluir o seguinte:

- Crie um Deadline Cloud Farm
- Execute o agente Deadline Cloud Worker

Para executar trabalhos com anexos de trabalhos, conclua as etapas a seguir.

Tópicos

Adicione uma configuração de anexos de tarefas à sua fila

Para habilitar anexos de trabalhos em sua fila, adicione uma configuração de anexos de trabalhos ao recurso de fila em sua conta.

Para adicionar uma configuração de anexos de tarefas à sua fila

1. Escolha sua primeira CloudShell guia e, em seguida, insira um dos seguintes comandos para usar um bucket do Amazon S3 para anexos de trabalhos.
   - Se você não tiver um bucket Amazon S3 privado existente, você pode criar e usar um novo bucket S3.

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

   - Se você já tem um bucket privado do Amazon S3, você pode usá-lo *MY_BUCKET_NAME* substituindo-o pelo nome do seu bucket.

     ```
     DEV_FARM_BUCKET=MY_BUCKET_NAME
     ```

2. Depois de criar ou escolher seu bucket do Amazon S3, adicione o nome do bucket `~/.bashrc` para disponibilizá-lo para outras sessões do terminal.

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

3. Crie uma função AWS Identity and Access Management (IAM) para a fila.

   ```
   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. Atualize sua fila para incluir as configurações de anexos do trabalho e a função do 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 se você atualizou sua fila.

```
deadline queue get
```

Uma saída como a seguinte é mostrada:

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

Envie simple_file_job com anexos de emprego

Quando você usa anexos de tarefas, os pacotes de tarefas devem fornecer ao Deadline Cloud informações suficientes para determinar o fluxo de dados da tarefa, como o uso de parâmetros. PATH No caso dosimple_file_job, você editou o `template.yaml` arquivo para informar ao Deadline Cloud que o fluxo de dados está no arquivo de entrada e no arquivo de saída.

Depois de adicionar a configuração dos anexos do trabalho à sua fila, você pode enviar a amostra simple_file_job com os anexos do trabalho. Depois de fazer isso, você pode visualizar o registro e a saída do trabalho para confirmar se o simple_file_job com anexos do trabalho está funcionando.

Para enviar o pacote de tarefas simple_file_job com anexos de tarefas

1. Escolha sua primeira CloudShell guia e abra o `JobBundle-Samples` diretório.
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```

3. Envie simple_file_job para a fila. Quando solicitado a confirmar o upload, insira**y**.

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

4. Para visualizar a saída do log da sessão de transferência de dados dos anexos do trabalho, execute o comando a seguir.

```
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. Liste as ações da sessão que foram executadas na sessão.

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

Uma saída como a seguinte é mostrada:

```
{
    "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"
                }
```

```
            }
        }
    ]
}
```

A primeira ação da sessão baixou os anexos do trabalho de entrada, enquanto a segunda ação executa a tarefa como nas etapas anteriores e, em seguida, carregou os anexos do trabalho de saída.

6. Liste o diretório de saída.

```
ls *.txt
```

Uma saída como a `hash.txt` que existe no diretório, mas `hash-jobattachments.txt` não existe porque o arquivo de saída da tarefa ainda não foi baixado.

7. Baixe a saída do trabalho mais recente.

```
deadline job download-output
```

8. Visualize a saída do arquivo baixado.

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

Uma saída como a seguinte é mostrada:

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

Entendendo como os anexos de trabalho são armazenados no Amazon S3

Você pode usar o AWS Command Line Interface (AWS CLI) para carregar ou baixar dados para anexos de tarefas, que são armazenados em buckets do Amazon S3. Entender como o Deadline Cloud armazena anexos de trabalhos no Amazon S3 ajudará você a desenvolver cargas de trabalho e integrações de pipeline.

Para inspecionar como os anexos de trabalho do Deadline Cloud são armazenados no Amazon S3

1. Escolha sua primeira CloudShell guia e, em seguida, abra o diretório de amostras do pacote de tarefas.

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

2. Inspecione as propriedades do trabalho.

```
deadline job get
```

Uma saída como a seguinte é mostrada:

```
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
```

O campo de anexos contém uma lista de estruturas de manifesto que descrevem os caminhos de dados de entrada e saída que a tarefa usa quando é executada. Veja `rootPath` o caminho do diretório local na máquina que enviou o trabalho. Para visualizar o sufixo do objeto Amazon S3 que contém um arquivo de manifesto, revise o. `inputManifestFile` O arquivo de manifesto contém metadados para um instantâneo da árvore de diretórios dos dados de entrada do trabalho.

3. Imprima bem o objeto de manifesto do Amazon S3 para ver a estrutura do diretório de entrada para o trabalho.

```
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 .
```

Uma saída como a seguinte é mostrada:

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

4. Crie o prefixo do Amazon S3 que contém manifestos para os anexos do trabalho de saída e liste o objeto abaixo dele.

```
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
```

Os anexos do trabalho de saída não são referenciados diretamente do recurso de trabalho, mas são colocados em um bucket do Amazon S3 com base no recurso da fazenda. IDs

5. Obtenha a chave de objeto de manifesto mais recente para o ID de ação de sessão específico e, em seguida, imprima bem os objetos do manifesto.

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

Você verá propriedades do arquivo `hash-jobattachments.txt` na saída, como as seguintes:

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

Seu trabalho terá apenas um único objeto manifesto por tarefa executada, mas, em geral, é possível ter mais objetos por tarefa executada.

6. Visualize a saída de armazenamento do Amazon S3 endereçável ao conteúdo sob o prefixo. `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 -
```

Uma saída como a seguinte é mostrada:

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

Próximas etapas

Depois de aprender a enviar trabalhos com anexos usando a CLI do Deadline Cloud, você pode explorar:

- Envie com o Deadline Cloudpara aprender a executar trabalhos usando um pacote OpenJD em seus hosts de trabalho.
- Adicione uma frota gerenciada por serviços à sua fazenda de desenvolvedores no Deadline Cloudpara executar seus trabalhos em hosts gerenciados pelo Deadline Cloud.
- Limpe os recursos da sua fazenda no Deadline Cloudpara encerrar os recursos que você usou neste tutorial.

Adicione uma frota gerenciada por serviços à sua fazenda de desenvolvedores no Deadline Cloud

AWS CloudShell não fornece capacidade computacional suficiente para testar cargas de trabalho maiores. Também não está configurado para funcionar com trabalhos que distribuem tarefas em vários hosts de trabalho.

Em vez de usar CloudShell, você pode adicionar uma frota gerenciada por serviços (SMF) do Auto Scaling à sua fazenda de desenvolvedores. Um SMF fornece capacidade computacional suficiente para cargas de trabalho maiores e pode lidar com trabalhos que precisam distribuir tarefas de trabalho em vários hosts de trabalho.

Antes de adicionar um SMF, você deve configurar uma fazenda, uma fila e uma frota do Deadline Cloud. Consulte Crie um Deadline Cloud Farm.

Para adicionar uma frota gerenciada por serviços à sua fazenda de desenvolvedores

1. Escolha sua primeira AWS CloudShell guia e, em seguida, crie a frota gerenciada pelo serviço e adicione sua ID de frota a. `.bashrc` Essa ação o torna disponível para outras sessões do 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. Associe o SMF à sua fila.

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

```
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_SMF_ID
```

3. Envie simple_file_job para a fila. Quando solicitado a confirmar o upload, insira**y**.

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

4. Confirme se o SMF está funcionando corretamente.

```
deadline fleet get
```

   - O trabalhador pode levar alguns minutos para começar. Repita o `deadline fleet get` comando até ver que a frota está funcionando.
   - A frota `queueFleetAssociationsStatus` de quatro serviços gerenciados será. `ACTIVE`
   - O SMF `autoScalingStatus` mudará de `GROWING` para`STEADY`.

   Seu status será semelhante ao seguinte:

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

5. Visualize o registro do trabalho que você enviou. Esse log é armazenado em um log no Amazon CloudWatch Logs, não no sistema de CloudShell arquivos.

```
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
```

Próximas etapas

Depois de criar e testar uma frota gerenciada por serviços, você deve remover os recursos que criou para evitar cobranças desnecessárias.

- Limpe os recursos da sua fazenda no Deadline Cloudpara encerrar os recursos que você usou neste tutorial.

Limpe os recursos da sua fazenda no Deadline Cloud

Para desenvolver e testar novas cargas de trabalho e integrações de pipeline, você pode continuar usando o farm de desenvolvedores do Deadline Cloud que você criou para este tutorial. Se você não precisar mais da sua fazenda de desenvolvedores, poderá excluir seus recursos, incluindo fazenda, frota, fila, funções AWS Identity and Access Management (IAM) e registros no Amazon CloudWatch Logs. Depois de excluir esses recursos, você precisará começar o tutorial novamente para usar os recursos. Para obter mais informações, consulte Introdução aos recursos do Deadline Cloud.

Para limpar os recursos da fazenda de desenvolvedores

1. Escolha sua primeira CloudShell guia e, em seguida, interrompa todas as associações de filas e frotas da sua fila.

```
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. Liste as associações de frotas de filas.

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

Talvez seja necessário executar novamente o comando até que a saída seja reportada e, em seguida`"status": "STOPPED"`, você pode prosseguir para a próxima etapa. O processo pode demorar vários minutos para ser concluído.


```
{
    "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. Exclua todas as associações de filas e frotas da sua fila.

```
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. Exclua todas as frotas associadas à sua fila.

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

5. Exclua a fila.

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

6. Exclua a fazenda.

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

7. Exclua outros AWS recursos da sua fazenda.

   a. Exclua a função de frota 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. Exclua a função do IAM da fila.

```
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. Exclua os grupos de CloudWatch log do Amazon Logs. Cada fila e frota tem seu próprio 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"
```

Crie trabalhos para enviar ao Deadline Cloud

Você envia trabalhos para o Deadline Cloud usando pacotes de trabalhos. Um pacote de tarefas é uma coleção de arquivos, incluindo um modelo de tarefa Open Job Description (OpenJD) e todos os arquivos de ativos necessários para renderizar a tarefa.

O modelo de trabalho descreve como os trabalhadores processam e acessam os ativos e fornece o script que o trabalhador executa. Os pacotes de tarefas permitem que artistas, diretores técnicos e desenvolvedores de funis enviem facilmente trabalhos complexos para o Deadline Cloud a partir de suas estações de trabalho locais ou do render farm local. Os pacotes de tarefas são particularmente úteis para equipes que trabalham em efeitos visuais, animações ou outros projetos de renderização de mídia em grande escala que exigem recursos computacionais escaláveis e sob demanda.

Você pode criar o pacote de tarefas usando o sistema de arquivos local para armazenar arquivos e um editor de texto para criar o modelo de tarefa. Depois de criar o pacote, envie o trabalho para o Deadline Cloud usando a CLI do Deadline Cloud ou uma ferramenta como um remetente do Deadline Cloud

Você pode armazenar seus ativos em um sistema de arquivos compartilhado entre seus funcionários ou pode usar os anexos de trabalho do Deadline Cloud para automatizar a transferência de ativos para compartimentos do S3, onde seus funcionários podem acessá-los. Os anexos de trabalho também ajudam a mover a saída de seus trabalhos de volta para suas estações de trabalho.

As seções a seguir fornecem instruções detalhadas sobre como criar e enviar pacotes de tarefas para o Deadline Cloud.

Tópicos

Modelos de Open Job Description (OpenJD) para Deadline Cloud

Um pacote de tarefas é uma das ferramentas que você usa para definir trabalhos para o AWS Deadline Cloud. Eles agrupam um modelo de Open Job Description (OpenJD) com informações adicionais, como arquivos e diretórios que seus trabalhos usam com anexos de trabalhos. Você usa a interface de linha de comando (CLI) do Deadline Cloud para usar um pacote de trabalhos para enviar trabalhos para execução de uma fila.

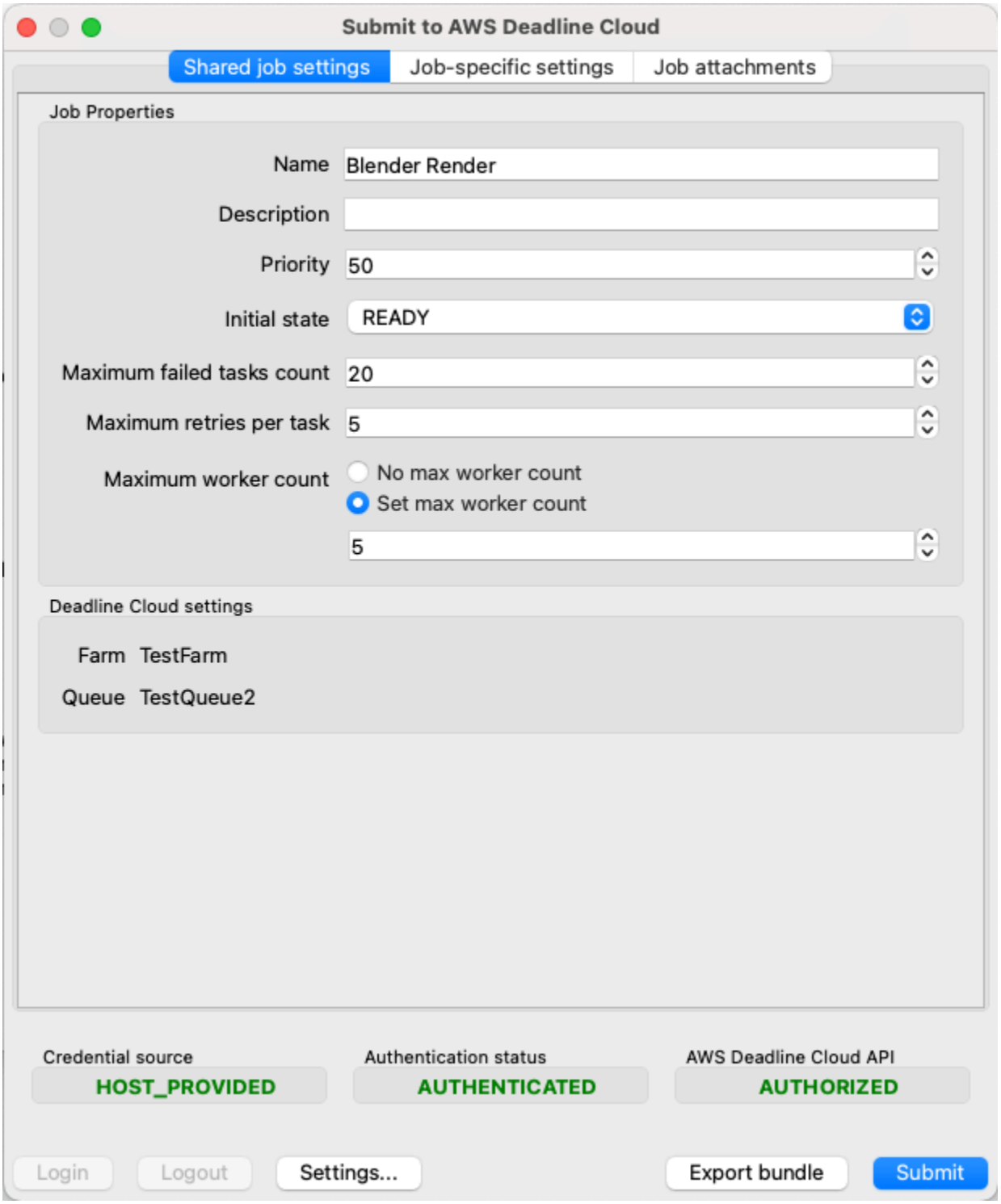
Um pacote de tarefas é uma estrutura de diretórios que contém um modelo de tarefa do OpenJD, outros arquivos que definem a tarefa e arquivos específicos da tarefa necessários como entrada para sua tarefa. Você pode especificar os arquivos que definem seu trabalho como arquivos YAML ou JSON.

O único arquivo necessário é `template.yaml` ou `template.json`. Você também pode incluir os seguintes arquivos:

```
/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
```

Use um pacote de trabalhos para envios de trabalhos personalizados com a CLI do Deadline Cloud e um anexo de trabalho, ou você pode usar uma interface gráfica de envio. Por exemplo, a seguir está a amostra do Blender de GitHub. Para executar a amostra usando o seguinte comando no diretório de amostras do Blender:

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



O painel de configurações específicas do trabalho é gerado a partir das `userInterface` propriedades dos parâmetros do trabalho definidos no modelo do trabalho.

Para enviar um trabalho usando a linha de comando, você pode usar um comando semelhante ao seguinte

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

Ou você pode usar a `deadline.client.api.create_job_from_job_bundle` função no pacote `deadline` Python.

Todos os plug-ins de envio de trabalhos fornecidos com o Deadline Cloud, como o plug-in Autodesk Maya, geram um pacote de trabalhos para seu envio e, em seguida, usam o pacote Python do Deadline Cloud para enviar seu trabalho para o Deadline Cloud. Você pode ver os pacotes de tarefas enviados no diretório de histórico de tarefas da sua estação de trabalho ou usando um remetente. Você pode encontrar seu diretório de histórico de trabalhos com o seguinte comando:

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

Quando seu trabalho está sendo executado em um funcionário do Deadline Cloud, ele tem acesso às variáveis de ambiente que fornecem informações sobre o trabalho. As variáveis de ambiente são:

Nome da variável	Available (Disponível)
DEADLINE_FARM_ID	Todas as ações
DEADLINE_FLEET_ID	Todas as ações
ID_DE_TRABALHO_PRAZO	Todas as ações
ID_DE_DATA_LIMITE	Todas as ações
DEADLINE_JOB_ID	Todas as ações
ID_DA_PASSO_PRAZO	Ações da tarefa
ID_DA_SESSÃO_PRAZO	Todas as ações
ID_DA_TAREFA_PRAZO	Ações da tarefa
ID_DE_DA_AÇÃO DA SESSÃO	Todas as ações

Tópicos

Elementos do modelo de trabalho para pacotes de tarefas

O modelo de trabalho define o ambiente de execução e os processos que são executados como parte de um trabalho do Deadline Cloud. Você pode criar parâmetros em um modelo para que ele possa ser usado para criar trabalhos que diferem somente nos valores de entrada, assim como uma função em uma linguagem de programação.

Quando você envia um trabalho para o Deadline Cloud, ele é executado em qualquer ambiente de fila aplicado à fila. Os ambientes de fila são criados usando a especificação de ambientes externos Open Job Description (OpenJD). Para obter detalhes, consulte o modelo de ambiente no repositório OpenJD. GitHub

Para uma introdução à criação de um trabalho com um modelo de trabalho do OpenJD, consulte Introdução à criação de um trabalho no repositório do GitHub OpenJD. Informações adicionais podem ser encontradas em Como os trabalhos são executados. Há exemplos de modelos de trabalho no diretório do GitHub repositório OpenJD. `samples`

Você pode definir o modelo de trabalho no formato YAML (`template.yaml`) ou no formato JSON (`template.json`). Os exemplos nesta seção são mostrados no formato YAML.

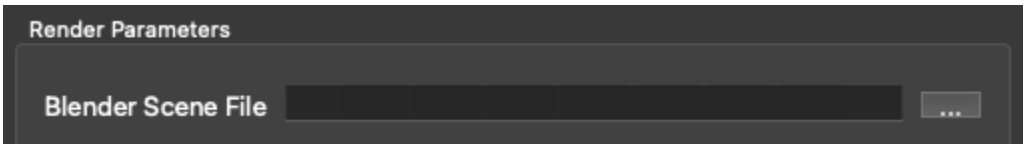
Por exemplo, o modelo de trabalho da `blender_render` amostra define um parâmetro de entrada `BlenderSceneFile` como um caminho de arquivo:

```
- 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.
```

A `userInterface` propriedade define o comportamento das interfaces de usuário geradas automaticamente para a linha de comando usando o `deadline bundle gui-submit` comando e nos plug-ins de envio de tarefas para aplicativos como o Autodesk Maya.

Neste exemplo, o widget de interface do usuário para inserir um valor para o `BlenderSceneFile` parâmetro é uma caixa de diálogo de seleção de arquivos que mostra somente arquivos. `.blend`



Para ver mais exemplos de uso do `userInteface` elemento, consulte a amostra gui_control_showcase no repositório em. deadline-cloud-samples GitHub

As `dataFlow` propriedades `objectType` e controlam o comportamento dos anexos de tarefas quando você envia uma tarefa de um pacote de tarefas. Nesse caso, `objectType: FILE` e `dataFlow:IN` significa que o valor de `BlenderSceneFile` é um arquivo de entrada para anexos do trabalho.

Em contraste, a definição do `OutputDir` parâmetro tem `objectType: DIRECTORY` e`dataFlow: OUT`:

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

O valor do `OutputDir` parâmetro é usado pelos anexos da tarefa como o diretório em que a tarefa grava os arquivos de saída.

Para obter mais informações sobre as `dataFlow` propriedades `objectType` e, consulte JobPathParameterDefinitiona especificação Open Job Description

O restante da amostra do modelo de `blender_render` tarefa define o fluxo de trabalho como uma única etapa, com cada quadro da animação renderizado como uma tarefa separada:

```
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 exemplo, se o valor do `Frames` parâmetro for`1-10`, ele define 10 tarefas. Cada tarefa tem um valor diferente para o `Frame` parâmetro. Para executar uma tarefa:

1. Todas as referências de variáveis na `data` propriedade do arquivo incorporado são expandidas, por exemplo`--render-frame 1`.

2. O conteúdo da `data` propriedade é gravado em um arquivo no diretório de trabalho da sessão no disco.
3. O onRun comando da tarefa é resolvido `bash` *`location of embedded file`* e, em seguida, executado.

Para obter mais informações sobre arquivos incorporados, sessões e locais mapeados por caminhos, consulte Como os trabalhos são executados na especificação Open Job Description.

Há mais exemplos de modelos de trabalho no repositório deadline-cloud-samples/job_bundles, bem como os exemplos de modelos fornecidos com a especificação Open Job Descriptions.

Divisão de tarefas para modelos de trabalho

O agrupamento de tarefas permite agrupar várias tarefas em uma única unidade de trabalho chamada fragmento. Em um trabalho de renderização, por exemplo, isso significa que o Deadline Cloud pode enviar vários quadros juntos em vez de um quadro por invocação de comando. Isso reduz a sobrecarga de iniciar aplicativos para cada tarefa e reduz o tempo de execução total do trabalho. Para obter detalhes, consulte Executando vários quadros ao mesmo tempo no wiki do OpenJD.

O OpenJD suporta extensões que adicionam recursos opcionais aos modelos de trabalho. A fragmentação de tarefas é ativada com a adição da `TASK_CHUNKING` extensão. Para usar o agrupamento, adicione a extensão ao seu modelo de trabalho e use o tipo de parâmetro da `CHUNK[INT]` tarefa. Envie trabalhos fragmentados usando o mesmo `deadline bundle submit` comando. Por exemplo, o modelo de trabalho a seguir renderiza quadros em blocos 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
```

Neste exemplo, o Deadline Cloud divide os 100 quadros em partes como `1-10`11-20, e assim por diante. A `{{Task.Param.Frame}}` variável se expande para uma expressão de intervalo como1-10. Como `rangeConstraint` está definido comoCONTIGUOUS, o intervalo está sempre no `start-end` formato. O script analisa esse intervalo e passa os quadros inicial e final para o Blender usando as `-e` opções `-s` e com. `--render-anim`

A `chunks` propriedade suporta os seguintes campos:

- `defaultTaskCount`— (Obrigatório) Quantas tarefas combinar em um único bloco. O valor máximo é 150.
- `rangeConstraint`— (Obrigatório) Se`CONTIGUOUS`, um pedaço é sempre um intervalo contíguo, como. `1-10` Se`NONCONTIGUOUS`, um pedaço pode ser um conjunto arbitrário, como. `1,3,7-10`
- `targetRuntimeSeconds`— (Opcional) O tempo de execução desejado em segundos para cada bloco. O Deadline Cloud pode ajustar dinamicamente o tamanho do bloco para se aproximar dessa meta depois que alguns trechos forem concluídos.

Para mais exemplos de fragmentação de tarefas, incluindo exemplos básicos e do Blender com partes contíguas e não contíguas, consulte os exemplos de fragmentação de tarefas no repositório de amostras do Deadline Cloud em. GitHub

> **Requisitos de frota gerenciados pelo cliente**
>
> A fragmentação de tarefas requer uma versão compatível do agente de trabalho. Se você usa frotas gerenciadas pelo cliente, certifique-se de que seus agentes de trabalho estejam atualizados antes de enviar trabalhos com fragmentação. As frotas gerenciadas por serviços sempre usam uma versão de agente de trabalho compatível.

> **Baixando a saída para trabalhos fragmentados**
>
> Quando você baixa a saída de uma única tarefa em um trabalho fragmentado, o Deadline Cloud baixa a saída de todo o fragmento. Por exemplo, se os quadros de 1 a 10 foram processados juntos, o download da saída do quadro 3 incluirá todos os quadros de 1 a 10. Esse recurso requer a `deadline-cloud` versão 0.53.3 ou posterior.

Elementos de valores de parâmetros para pacotes de tarefas

Você pode usar o arquivo de parâmetros para definir os valores de alguns dos parâmetros do trabalho no modelo de trabalho ou argumentos da solicitação de CreateJoboperação no pacote de trabalhos para que você não precise definir valores ao enviar um trabalho. A interface do usuário para envio de trabalhos permite que você modifique esses valores.

Você pode definir o modelo de trabalho no formato YAML (`parameter_values.yaml`) ou no formato JSON (`parameter_values.json`). Os exemplos nesta seção são mostrados no formato YAML.

No YAML, o formato do arquivo é:

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

Cada elemento da `parameterValues` lista deve ser um dos seguintes:

- Um parâmetro de trabalho definido no modelo de trabalho.
- Um parâmetro de trabalho definido em um ambiente de fila para a fila para a qual você envia o trabalho.
- Um parâmetro especial passado para a `CreateJob` operação ao criar um trabalho.
  - `deadline:priority`— O valor deve ser um número inteiro. Ele é passado para a `CreateJob` operação como parâmetro de prioridade.
  - `deadline:targetTaskRunStatus`— O valor deve ser uma string. Ele é passado para a `CreateJob` operação como o parâmetro targetTaskRunStatus.
  - `deadline:maxFailedTasksCount`— O valor deve ser um número inteiro. Ele é passado para a `CreateJob` operação como o parâmetro maxFailedTasksCount.
  - `deadline:maxRetriesPerTask`— O valor deve ser um número inteiro. Ele é passado para a `CreateJob` operação como o parâmetro maxRetriesPerTask.
  - `deadline:maxWorkercount`— O valor deve ser um número inteiro. Ele é passado para a `CreateJob` operação como maxWorkerCountparâmetro.

Um modelo de trabalho é sempre um modelo em vez de um trabalho específico a ser executado. Um arquivo de valores de parâmetros permite que um pacote de tarefas atue como um modelo se alguns parâmetros não tiverem valores definidos nesse arquivo ou como um envio de trabalho específico se todos os parâmetros tiverem valores.

Por exemplo, a amostra blender_render não tem um arquivo de parâmetros e seu modelo de trabalho define parâmetros sem valores padrão. Esse modelo deve ser usado como modelo para criar

trabalhos. Depois de criar um trabalho usando esse pacote de trabalhos, o Deadline Cloud grava um novo pacote de trabalhos no diretório do histórico de trabalhos.

Por exemplo, quando você envia um trabalho com o seguinte comando:

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

O novo pacote de tarefas contém um `parameter_values.yaml` arquivo que contém os parâmetros 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
```

Você pode criar o mesmo trabalho com o seguinte comando:

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

> **Note**
>
> O pacote de trabalhos que você envia é salvo no seu diretório de histórico de trabalhos. Você pode encontrar a localização desse diretório com o seguinte comando:
>
> ```
> deadline config get settings.job_history_dir
> ```

Elementos de referências de ativos para pacotes de tarefas

Você pode usar os anexos de trabalho do Deadline Cloud para transferir arquivos entre sua estação de trabalho e o Deadline Cloud. O arquivo de referência do ativo lista os arquivos e diretórios de entrada, bem como os diretórios de saída dos seus anexos. Se você não listar todos os arquivos e diretórios desse arquivo, poderá selecioná-los ao enviar um trabalho com o `deadline bundle gui-submit` comando.

Esse arquivo não tem efeito se você não estiver usando anexos de trabalho.

Você pode definir o modelo de trabalho no formato YAML (`asset_references.yaml`) ou no formato JSON (`asset_references.json`). Os exemplos nesta seção são mostrados no formato YAML.

No YAML, o formato do arquivo é:

```
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
```

Ao selecionar o arquivo de entrada ou saída para carregar no Amazon S3, o Deadline Cloud compara o caminho do arquivo com os caminhos listados em seus perfis de armazenamento. Cada localização do sistema SHARED de arquivos do tipo -type em um perfil de armazenamento abstrai um compartilhamento de arquivos de rede que está montado em suas estações de trabalho e hosts de trabalho. O Deadline Cloud carrega somente arquivos que não estão em um desses compartilhamentos de arquivos.

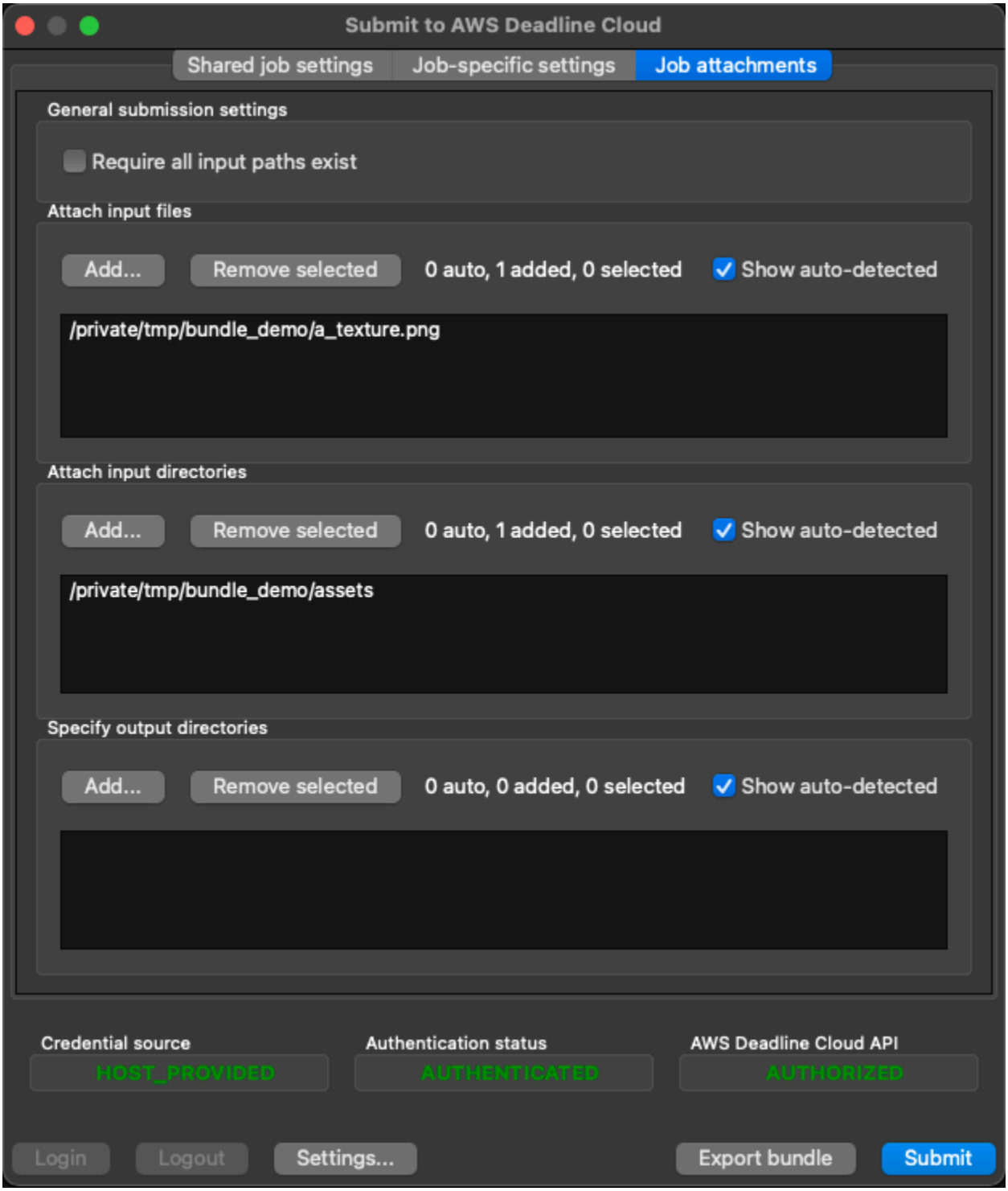
Para obter mais informações sobre como criar e usar perfis de armazenamento, consulte Armazenamento compartilhado no Deadline Cloud no Guia do usuário do AWS Deadline Cloud.

Example- O arquivo de referência do ativo criado pela GUI do Deadline Cloud

Use o comando a seguir para enviar um trabalho usando a amostra blender_render.

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

Adicione alguns arquivos adicionais ao trabalho na guia Anexos do trabalho:



Depois de enviar o trabalho, você pode ver o `asset_references.yaml` arquivo no pacote de trabalhos no diretório do histórico de trabalhos para ver os ativos no arquivo 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: []
```

Usando arquivos em seus trabalhos

Muitos dos trabalhos que você envia para o AWS Deadline Cloud têm arquivos de entrada e saída. Seus arquivos de entrada e diretórios de saída podem estar localizados em uma combinação de sistemas de arquivos compartilhados e unidades locais. Os trabalhos precisam localizar o conteúdo nesses locais. O Deadline Cloud fornece dois recursos, anexos de tarefas e perfis de armazenamento que funcionam juntos para ajudar seus trabalhos a localizar os arquivos de que precisam.

Os anexos de emprego oferecem vários benefícios

- Mova arquivos entre hosts usando o Amazon S3
- Transfira arquivos da sua estação de trabalho para os hosts dos funcionários e vice-versa
- Disponível para trabalhos em filas nas quais você ativa o recurso
- Usado principalmente com frotas gerenciadas por serviços, mas também compatível com frotas gerenciadas pelo cliente.

Use perfis de armazenamento para mapear o layout dos locais compartilhados do sistema de arquivos em sua estação de trabalho e nos hosts de trabalho. Esse mapeamento ajuda seus trabalhos a localizar arquivos e diretórios compartilhados quando suas localizações diferem entre sua estação de trabalho e hosts de trabalho, como configurações multiplataforma com estações de trabalho baseadas e hosts de trabalho Windows baseados. Linux O mapa do perfil de armazenamento da configuração do seu sistema de arquivos também é usado pelos anexos de tarefas para identificar os arquivos que precisam ser transferidos entre os hosts por meio do Amazon S3.

Se você não estiver usando anexos de trabalho e não precisar remapear os locais de arquivos e diretórios entre estações de trabalho e hosts de trabalho, não precisará modelar seus compartilhamentos de arquivos com perfis de armazenamento.

Tópicos

Exemplo de infraestrutura de projeto

Para demonstrar o uso de anexos de tarefas e perfis de armazenamento, configure um ambiente de teste com dois projetos separados. Você pode usar o console do Deadline Cloud para criar os recursos de teste.

1. Se você ainda não o fez, crie uma fazenda de testes. Para criar uma fazenda, siga o procedimento em Criar uma fazenda.
2. Crie duas filas para trabalhos em cada um dos dois projetos. Para criar filas, siga o procedimento em Criar uma fila.

    a. Crie a primeira fila chamada**`Q1`**. Use a configuração a seguir, use os padrões para todos os outros itens.

    - Para anexos de trabalho, escolha Criar um novo bucket do Amazon S3.
    - Selecione Habilitar associação com frotas gerenciadas pelo cliente.
    - Para executar como usuário, insira tanto **`jobuser`** para o usuário quanto para o grupo POSIX.
    - Para a função de serviço de fila, crie uma nova função chamada **`AssetDemoFarm-Q1-Role`**
    - Desmarque a caixa de seleção padrão do ambiente de fila conda.

    b. Crie a segunda fila chamada**`Q2`**. Use a configuração a seguir, use os padrões para todos os outros itens.

    - Para anexos de trabalho, escolha Criar um novo bucket do Amazon S3.
    - Selecione Habilitar associação com frotas gerenciadas pelo cliente.
    - Para executar como usuário, insira tanto **`jobuser`** para o usuário quanto para o grupo POSIX.

    - Para a função de serviço de fila, crie uma nova função chamada **`AssetDemoFarm-Q2-Role`**
    - Desmarque a caixa de seleção padrão do ambiente de fila conda.

3. Crie uma única frota gerenciada pelo cliente que execute os trabalhos nas duas filas. Para criar a frota, siga o procedimento em Criar uma frota gerenciada pelo cliente. Use a seguinte configuração:

    - Para Nome, use**`DemoFleet`**.
    - Para Tipo de frota, escolha Gerenciado pelo cliente
    - Para a função de serviço Fleet, crie uma nova função chamada AssetDemoFarm-Fleet-Role.
    - Não associe a frota a nenhuma fila.

O ambiente de teste pressupõe que há três sistemas de arquivos compartilhados entre hosts usando compartilhamentos de arquivos de rede. Neste exemplo, os locais têm os seguintes nomes:

- `FSCommon`- contém ativos de trabalho de entrada que são comuns aos dois projetos.
- `FS1`- contém ativos de trabalho de entrada e saída para o projeto 1.
- `FS2`- contém ativos de trabalho de entrada e saída para o projeto 2.

O ambiente de teste também pressupõe que haja três estações de trabalho, da seguinte forma:

- `WSAll`- Uma estação de trabalho Linux baseada usada por desenvolvedores para todos os projetos. Os locais do sistema de arquivos compartilhados são:
    - `FSCommon`: `/shared/common`
    - `FS1`: `/shared/projects/project1`
    - `FS2`: `/shared/projects/project2`
- `WS1`- Uma estação de trabalho Windows baseada usada para o projeto 1. Os locais do sistema de arquivos compartilhados são:
    - `FSCommon`: `S:\`
    - `FS1`: `Z:\`
    - `FS2`: Não disponível
- `WS1`- Uma estação de trabalho macOS baseada usada para o projeto 2.Os locais do sistema de arquivos compartilhados são:

- FSCommon: /Volumes/common
- FS1: Não disponível
- FS2: /Volumes/projects/project2

Por fim, defina os locais compartilhados do sistema de arquivos para os trabalhadores da sua frota. Os exemplos a seguir se referem a essa configuração comoWorkerConfig. Os locais compartilhados são:

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

Você não precisa configurar nenhum sistema de arquivos, estações de trabalho ou trabalhadores compartilhados que correspondam a essa configuração. Os locais compartilhados não precisam existir para a demonstração.

Perfis de armazenamento e mapeamento de caminhos

Use perfis de armazenamento para modelar os sistemas de arquivos em sua estação de trabalho e hosts de trabalho. Cada perfil de armazenamento descreve o sistema operacional e o layout do sistema de arquivos de uma das configurações do sistema. Este tópico descreve como usar perfis de armazenamento para modelar as configurações do sistema de arquivos de seus hosts para que o Deadline Cloud possa gerar regras de mapeamento de caminhos para seus trabalhos e como essas regras de mapeamento de caminhos são geradas a partir de seus perfis de armazenamento.

Ao enviar um trabalho para o Deadline Cloud, você pode fornecer um ID de perfil de armazenamento opcional para o trabalho. Esse perfil de armazenamento descreve o sistema de arquivos da estação de trabalho de envio. Ele descreve a configuração original do sistema de arquivos que os caminhos de arquivo no modelo de trabalho usam.

Você também pode associar um perfil de armazenamento a uma frota. O perfil de armazenamento descreve a configuração do sistema de arquivos de todos os hosts de trabalho na frota. Se você tiver trabalhadores com diferentes configurações de sistema de arquivos, esses trabalhadores devem ser atribuídos a uma frota diferente em sua fazenda.

As regras de mapeamento de caminhos descrevem como os caminhos devem ser remapeados, desde a forma como são especificados no trabalho até a localização real do caminho em um host

de trabalho. O Deadline Cloud compara a configuração do sistema de arquivos descrita no perfil de armazenamento de um trabalho com o perfil de armazenamento da frota que está executando o trabalho para derivar essas regras de mapeamento de caminhos.

Tópicos

- Modele locais de sistemas de arquivos compartilhados com perfis de armazenamento
- Configurar perfis de armazenamento para frotas
- Configurar perfis de armazenamento para filas
- Derive regras de mapeamento de caminhos a partir de perfis de armazenamento

Modele locais de sistemas de arquivos compartilhados com perfis de armazenamento

Um perfil de armazenamento modela a configuração do sistema de arquivos de uma de suas configurações de host. Há quatro configurações de host diferentes na infraestrutura do projeto de amostra. Neste exemplo, você cria um perfil de armazenamento separado para cada um. Você pode criar um perfil de armazenamento usando qualquer um dos seguintes:

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation recurso
- Console do AWS

Um perfil de armazenamento é composto por uma lista de localizações do sistema de arquivos, cada uma informando ao Deadline Cloud a localização e o tipo de localização do sistema de arquivos que é relevante para trabalhos enviados ou executados em um host. Um perfil de armazenamento só deve modelar os locais relevantes para os trabalhos. Por exemplo, o `FSCommon` local compartilhado está localizado na estação de trabalho `WS1` em`S:\`, então o local correspondente do sistema de arquivos é:

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

Use os comandos a seguir para criar o perfil de armazenamento para as configurações `WS1` da estação de trabalho `WS3` e a configuração do trabalhador `WorkerConfig` usando o AWS CLIin: `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**
>
> Você deve consultar os locais do sistema de arquivos em seus perfis de armazenamento usando os mesmos valores para a `name` propriedade em todos os perfis de armazenamento em sua fazenda. O Deadline Cloud compara os nomes para determinar se os locais do sistema de arquivos de diferentes perfis de armazenamento se referem ao mesmo local ao gerar regras de mapeamento de caminhos.

Configurar perfis de armazenamento para frotas

Você pode configurar uma frota para incluir um perfil de armazenamento que modele as localizações do sistema de arquivos em todos os trabalhadores da frota. A configuração do sistema de arquivos host de todos os trabalhadores em uma frota deve corresponder ao perfil de armazenamento da frota. Trabalhadores com diferentes configurações de sistema de arquivos devem estar em frotas separadas.

Para definir a configuração da sua frota para usar o perfil `WorkerConfig` de armazenamento, use o AWS CLIem AWS CloudShell:

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

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

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

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

Configurar perfis de armazenamento para filas

A configuração de uma fila inclui uma lista de nomes com distinção entre maiúsculas e minúsculas dos locais do sistema de arquivos compartilhado aos quais os trabalhos enviados à fila exigem acesso. Por exemplo, trabalhos enviados à fila `Q1` exigem locais do sistema de arquivos e. `FSCommon` `FS1` Os trabalhos enviados à fila `Q2` exigem localizações do sistema de arquivos `FSCommon` e. `FS2`

Para definir as configurações da fila para exigir esses locais do sistema de arquivos, use o seguinte 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
```

A configuração de uma fila também inclui uma lista de perfis de armazenamento permitidos que se aplicam aos trabalhos enviados e às frotas associadas a essa fila. Somente perfis de armazenamento que definem os locais do sistema de arquivos para todos os locais necessários do sistema de arquivos para a fila são permitidos na lista de perfis de armazenamento permitidos da fila.

Um trabalho falhará se você o enviar com um perfil de armazenamento que não esteja na lista de perfis de armazenamento permitidos para a fila. Você sempre pode enviar um trabalho sem perfil de armazenamento para uma fila. As configurações da estação de trabalho rotuladas `WSAll` e `WS1` ambas têm os locais necessários do sistema de arquivos (`FSCommoneFS1`) para a fila. `Q1` Eles

precisam ter permissão para enviar trabalhos para a fila. Da mesma forma, as configurações `WSAll` da estação de trabalho `WS2` atendem aos requisitos de fila. `Q2` Eles precisam ter permissão para enviar trabalhos para essa fila. Atualize as duas configurações de fila para permitir que os trabalhos sejam enviados com esses perfis de armazenamento usando o seguinte 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
```

Se você adicionar o perfil `WS2` de armazenamento à lista de perfis de armazenamento permitidos para a fila, `Q1` ele falhará:

```
$ 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
```

Isso ocorre porque o perfil `WS2` de armazenamento não contém uma definição para a localização do sistema de arquivos chamada `FS1` que a fila `Q1` exige.

A associação de uma frota configurada a um perfil de armazenamento que não está na lista de perfis de armazenamento permitidos da fila também falha. Por exemplo:

```
$ 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 corrigir o erro, adicione o perfil de armazenamento nomeado `WorkerConfig` à lista de perfis de armazenamento permitidos para fila `Q1` e filaQ2. Em seguida, associe a frota a essas filas para que os trabalhadores da frota possam executar trabalhos em ambas as filas.

```
# 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
```

Derive regras de mapeamento de caminhos a partir de perfis de armazenamento

As regras de mapeamento de caminhos descrevem como os caminhos devem ser remapeados do trabalho até a localização real do caminho em um host de trabalho. Quando uma tarefa está sendo executada em um trabalhador, o perfil de armazenamento do trabalho é comparado ao perfil de armazenamento da frota do trabalhador para derivar as regras de mapeamento de caminhos para a tarefa.

O Deadline Cloud cria uma regra de mapeamento para cada um dos locais necessários do sistema de arquivos na configuração da fila. Por exemplo, um trabalho enviado com o perfil `WSAll` de armazenamento para a fila `Q1` tem as regras de mapeamento de caminhos:

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

O Deadline Cloud cria regras para `FSComm` os locais do sistema de `FS1` arquivos, mas não para o local do sistema de `FS2` arquivos, embora os perfis `WSAll` e `WorkerConfig` de armazenamento

sejam definidosFS2. Isso ocorre porque a lista Q1 de localizações obrigatórias do sistema de arquivos da fila é`["FSComm", "FS1"]`.

Você pode confirmar as regras de mapeamento de caminhos disponíveis para trabalhos enviados com um perfil de armazenamento específico enviando um trabalho que imprima o arquivo de regras de mapeamento de caminhos do Open Job Description e, em seguida, lendo o registro da sessão após a conclusão do trabalho:

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

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

Se você usar a CLI do Deadline Cloud para enviar trabalhos, sua `settings.storage_profile_id` configuração define o perfil de armazenamento que os trabalhos enviados com a CLI terão. Para enviar trabalhos com o perfil `WSAll` de armazenamento, defina:

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

Para executar um trabalhador gerenciado pelo cliente como se estivesse sendo executado na infraestrutura de amostra, siga o procedimento em Executar o agente de trabalho no Guia do usuário do Deadline Cloud para executar um trabalhador com. AWS CloudShell Se você seguiu essas instruções antes, exclua primeiro `~/demoenv-logs` os `~/demoenv-persist` diretórios e. Além disso, defina os valores das variáveis de `DEV_CMF_ID` ambiente `DEV_FARM_ID` e de ambiente às quais as direções fazem referência da seguinte forma antes de fazer isso:

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

Depois que o trabalho for executado, você poderá ver as regras de mapeamento de caminhos no arquivo de log do trabalho:

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

O log contém mapeamento para os sistemas de `FSComm` arquivos `FS1` e. Reformatada para facilitar a leitura, a entrada do registro tem a seguinte aparência:

```
{
    "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"
        }
    ]
```

Você pode enviar trabalhos com diferentes perfis de armazenamento para ver como as regras de mapeamento de caminhos mudam.

Use anexos de trabalho para compartilhar arquivos

Use anexos de tarefas para disponibilizar arquivos que não estão em diretórios compartilhados para suas tarefas e para capturar os arquivos de saída se eles não estiverem gravados em diretórios compartilhados. Job attachments usa o Amazon S3 para transferir arquivos entre hosts. Os arquivos são armazenados em buckets do S3 e você não precisa fazer upload de um arquivo se o conteúdo não tiver sido alterado.

Você deve usar anexos de trabalho ao executar trabalhos em frotas gerenciadas por serviços, pois os hosts não compartilham os locais do sistema de arquivos. Os anexos de trabalho também são úteis com frotas gerenciadas pelo cliente quando os arquivos de entrada ou saída de um trabalho são armazenados em um sistema de arquivos de rede compartilhado, como quando seu pacote de trabalhos contém scripts shell ou Python.

Quando você envia um pacote de trabalhos com a CLI do Deadline Cloud ou com um remetente do Deadline Cloud, os anexos do trabalho usam o perfil de armazenamento do trabalho e os locais necessários do sistema de arquivos da fila para identificar os arquivos de entrada que não estão em um host de trabalho e devem ser enviados para o Amazon S3 como parte do envio do trabalho. Esses perfis de armazenamento também ajudam o Deadline Cloud a identificar os arquivos de saída nos locais de hospedagem dos trabalhadores que devem ser carregados no Amazon S3 para que estejam disponíveis em sua estação de trabalho.

Os exemplos de anexos de trabalho usam as configurações de fazenda, frota, filas e perfis de armazenamento de e. Exemplo de infraestrutura de projeto Perfis de armazenamento e mapeamento de caminhos Você deve passar por essas seções antes desta.

Nos exemplos a seguir, você usa um pacote de tarefas de amostra como ponto de partida e, em seguida, o modifica para explorar a funcionalidade do anexo de tarefas. Pacotes de tarefas são a melhor maneira de seus trabalhos usarem anexos de trabalho. Eles combinam um modelo de trabalho do Open Job Description em um diretório com arquivos adicionais que listam os arquivos e diretórios exigidos pelos trabalhos que usam o pacote de trabalhos. Para obter mais informações sobre pacotes de tarefas, consulteModelos de Open Job Description (OpenJD) para Deadline Cloud.

Enviando arquivos com um trabalho

Com o Deadline Cloud, você pode permitir que os fluxos de trabalho acessem arquivos de entrada que não estão disponíveis em locais de sistemas de arquivos compartilhados em hosts de trabalho. Os anexos de tarefas permitem que as tarefas de renderização acessem arquivos que residem somente em uma unidade de estação de trabalho local ou em um ambiente de frota gerenciado por

serviços. Ao enviar um pacote de tarefas, você pode incluir listas de arquivos de entrada e diretórios exigidos pelo trabalho. O Deadline Cloud identifica esses arquivos não compartilhados, os carrega da máquina local para o Amazon S3 e os baixa para o host do trabalhador. Ele simplifica o processo de transferência de ativos de entrada para os nós de renderização, garantindo que todos os arquivos necessários estejam acessíveis para execução distribuída de trabalhos.

Você pode especificar os arquivos para trabalhos diretamente no pacote de trabalhos, usar parâmetros no modelo de trabalho que você fornece usando variáveis de ambiente ou um script e usar o `assets_references` arquivo do trabalho. Você pode usar um desses métodos ou uma combinação dos três. Você pode especificar um perfil de armazenamento para o pacote do trabalho para que ele carregue somente os arquivos que foram alterados na estação de trabalho local.

Esta seção usa um exemplo de pacote de tarefas GitHub para demonstrar como o Deadline Cloud identifica os arquivos em seu trabalho para upload, como esses arquivos são organizados no Amazon S3 e como são disponibilizados para os hosts de trabalho que processam seus trabalhos.

Tópicos

Como o Deadline Cloud carrega arquivos para o Amazon S3

Este exemplo mostra como o Deadline Cloud carrega arquivos da sua estação de trabalho ou host de trabalho para o Amazon S3 para que eles possam ser compartilhados. Ele usa um pacote de tarefas de amostra GitHub e a CLI do Deadline Cloud para enviar trabalhos.

Comece clonando o GitHubrepositório de amostras do Deadline Cloud em seu AWS CloudShellambiente e, em seguida, copie o pacote de `job_attachments_devguide` tarefas em seu diretório inicial:

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

Instale a CLI do Deadline Cloud para enviar pacotes de tarefas:

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

O pacote de `job_attachments_devguide` tarefas tem uma única etapa com uma tarefa que executa um script bash shell cujo local do sistema de arquivos é passado como um parâmetro do trabalho. A definição do parâmetro do trabalho é:

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

O `IN` valor da `dataFlow` propriedade informa aos anexos do trabalho que o valor do `ScriptFile` parâmetro é uma entrada para o trabalho. O valor da `default` propriedade é um local relativo ao diretório do pacote de tarefas, mas também pode ser um caminho absoluto. Essa definição de parâmetro declara o `script.sh` arquivo no diretório do pacote de tarefas como um arquivo de entrada necessário para a execução da tarefa.

Em seguida, certifique-se de que a CLI do Deadline Cloud não tenha um perfil de armazenamento configurado e envie o trabalho para a fila: Q1

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

deadline config set settings.storage_profile_id ''

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

A saída da CLI do Deadline Cloud após a execução desse comando é semelhante a:

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

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

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

Quando você envia o trabalho, o Deadline Cloud primeiro faz o hash do `script.sh` arquivo e depois o carrega para o Amazon S3.

O Deadline Cloud trata o bucket S3 como armazenamento endereçável ao conteúdo. Os arquivos são enviados para objetos do S3. O nome do objeto é derivado de um hash do conteúdo do arquivo. Se dois arquivos tiverem conteúdos idênticos, eles terão o mesmo valor de hash, independentemente de onde os arquivos estejam localizados ou do nome que tenham. Esse armazenamento endereçável por conteúdo permite que o Deadline Cloud evite o upload de um arquivo se ele já estiver disponível.

Você pode usar a AWS CLI para ver os objetos que foram enviados para o 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
```

Dois objetos foram enviados para o S3:

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— O conteúdo do`script.sh`. O valor `87cb19095dd5d78fcaf56384ef0e6241` na chave do objeto é o hash do conteúdo do arquivo, e a extensão `xxh128` indica que o valor do hash foi calculado como um xxhash de 128 bits.
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— O objeto manifesto para o envio do trabalho. Os valores `<farm-id>`<queue-id>, e `<guid>` são o identificador da fazenda, o identificador

da fila e um valor hexidecimal aleatório. O valor `a1d221c7fd97b08175b3872a37428e8c` neste exemplo é um valor de hash calculado a partir da string`/home/cloudshell-user/job_attachments_devguide`, o diretório em que `script.sh` está localizado.

O objeto de manifesto contém as informações dos arquivos de entrada em um caminho raiz específico carregado no S3 como parte do envio do trabalho. Baixe esse arquivo de manifesto (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`). Seu conteúdo é semelhante a:

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

Isso indica que o arquivo `script.sh` foi carregado e o hash do conteúdo desse arquivo é`87cb19095dd5d78fcaf56384ef0e6241`. Esse valor de hash corresponde ao valor no nome `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128` do objeto. Ele é usado pelo Deadline Cloud para saber qual objeto baixar para o conteúdo desse arquivo.

O esquema completo desse arquivo está disponível em GitHub.

Ao usar a CreateJob operação, você pode definir a localização dos objetos do manifesto. Você pode usar a GetJoboperação para ver a localização:

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

Como o Deadline Cloud escolhe os arquivos a serem enviados

Os arquivos e diretórios que os anexos de trabalho consideram para serem carregados no Amazon S3 como entradas para seu trabalho são:

- Os valores de todos os parâmetros `PATH` de trabalho do tipo definidos no modelo de trabalho do pacote de tarefas com um `dataFlow` valor de `IN` ou. `INOUT`
- Os arquivos e diretórios listados como entradas no arquivo de referências de ativos do pacote de tarefas.

Se você enviar um trabalho sem perfil de armazenamento, todos os arquivos considerados para envio serão enviados. Se você enviar um trabalho com um perfil de armazenamento, os arquivos não serão enviados para o Amazon S3 se estiverem localizados nos locais do sistema de arquivos do `SHARED` tipo do perfil de armazenamento, que também são locais obrigatórios do sistema de arquivos para a fila. Espera-se que esses locais estejam disponíveis nos hosts de trabalho que executam o trabalho, portanto, não há necessidade de carregá-los para o S3.

Neste exemplo, você cria locais do sistema de `SHARED` arquivos `WSAll` em seu CloudShell ambiente da AWS e, em seguida, adiciona arquivos a esses locais do sistema de arquivos. Use o seguinte 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
```

Em seguida, adicione um arquivo de referências de ativos ao pacote de tarefas que inclua todos os arquivos que você criou como entradas para o trabalho. Use o seguinte 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
```

Em seguida, configure a CLI do Deadline Cloud para enviar trabalhos com o perfil `WSAll` de armazenamento e, em seguida, envie o pacote de trabalhos:

```
# 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/
```

O Deadline Cloud carrega dois arquivos para o Amazon S3 quando você envia o trabalho. Você pode baixar os objetos de manifesto do trabalho do S3 para ver os arquivos enviados:

```
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
```

Neste exemplo, há um único arquivo de manifesto com o seguinte conteúdo:

```
{
    "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
}
```

Use a GetJob operação do manifesto para ver se `rootPath` é "/".

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

O caminho raiz para o conjunto de arquivos de entrada é sempre o subcaminho comum mais longo desses arquivos. Se o seu trabalho foi enviado pelo Windows Instead e há arquivos de entrada sem um subcaminho comum porque eles estavam em unidades diferentes, você verá um caminho raiz separado em cada unidade. Os caminhos em um manifesto são sempre relativos ao caminho raiz do manifesto, então os arquivos de entrada que foram carregados são:

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— O arquivo de script no pacote de tarefas.
- `/shared/projects/project2/file.txt`— O arquivo em um local do sistema de `SHARED` arquivos no perfil `WSAll` de armazenamento que não está na lista de locais necessários do sistema de arquivos para a fila`Q1`.

Os arquivos nos locais do sistema de arquivos `FSCommon` (`/shared/common/file.txt`) e `FS1` (`/shared/projects/project1/file.txt`) não estão na lista. Isso ocorre porque esses locais

do sistema de arquivos estão SHARED no perfil WSAll de armazenamento e ambos estão na lista de locais necessários do sistema de arquivos na filaQ1.

Você pode ver os locais do sistema de arquivos considerados SHARED para um trabalho que é enviado com um perfil de armazenamento específico com a GetStorageProfileForQueue operação. Para consultar o perfil de armazenamento WSAll para fila, Q1 use o seguinte 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
```

Como os trabalhos encontram arquivos de entrada de anexos de trabalhos

Para que um trabalho use os arquivos que o Deadline Cloud carrega para o Amazon S3 usando anexos de trabalho, seu trabalho precisa desses arquivos disponíveis por meio do sistema de arquivos nos hosts do trabalhador. Quando uma sessão do seu trabalho é executada em um host de trabalho, o Deadline Cloud baixa os arquivos de entrada do trabalho em um diretório temporário na unidade local do host de trabalho e adiciona regras de mapeamento de caminhos para cada um dos caminhos raiz do trabalho na localização do sistema de arquivos na unidade local.

Neste exemplo, inicie o agente Deadline Cloud Worker em uma CloudShell guia da AWS. Deixe que todos os trabalhos enviados anteriormente terminem de ser executados e, em seguida, exclua os registros de trabalhos do diretório de registros:

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

O script a seguir modifica o pacote de tarefas para mostrar todos os arquivos no diretório de trabalho temporário da sessão e o conteúdo do arquivo de regras de mapeamento de caminhos e, em seguida, envia uma tarefa com o pacote 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/
```

Você pode ver o registro da execução do trabalho após ele ter sido executado pelo trabalhador em seu AWS CloudShell ambiente:

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

O registro mostra que a primeira coisa que ocorre na sessão é que os dois arquivos de entrada do trabalho são baixados para o trabalhador:

```
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.
```

A seguir está a saída de `script.sh` run by the job:

- Os arquivos de entrada enviados quando o trabalho foi enviado estão localizados em um diretório cujo nome começa com “assetroot” no diretório temporário da sessão.
- Os caminhos dos arquivos de entrada foram realocados em relação ao diretório “assetroot” em vez de em relação ao caminho raiz do manifesto de entrada () do trabalho. `"/"`
- O arquivo de regras de mapeamento de caminhos contém uma regra adicional que é remapeada `"/"` para o caminho absoluto do diretório “assetroot”.

Por exemplo:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
```

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

> **Note**
>
> Se o trabalho que você enviar tiver vários manifestos com caminhos raiz diferentes, há um diretório com o nome “assetroot” diferente para cada um dos caminhos raiz.

Se precisar referenciar a localização do sistema de arquivos realocado de um dos seus arquivos de entrada, diretórios ou localizações do sistema de arquivos, você pode processar o arquivo de regras de mapeamento de caminhos em sua tarefa e realizar o remapeamento sozinho, ou adicionar um parâmetro de PATH tipo de tarefa ao modelo de tarefa em seu pacote de tarefas e passar o valor que você precisa remapear como o valor desse parâmetro. Por exemplo, o exemplo a seguir modifica o pacote de tarefas para ter um desses parâmetros de tarefa e, em seguida, envia uma tarefa com a localização `/shared/projects/project2` do sistema de arquivos como seu 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
```

O arquivo de log para a execução desse trabalho contém sua saída:

```
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
```

Obtendo arquivos de saída de um trabalho

Este exemplo mostra como o Deadline Cloud identifica os arquivos de saída que seus trabalhos geram, decide se deseja fazer o upload desses arquivos para o Amazon S3 e como você pode obter esses arquivos de saída em sua estação de trabalho.

Use o pacote de `job_attachments_devguide_output` tarefas em vez do pacote de `job_attachments_devguide` tarefas para este exemplo. Comece fazendo uma cópia do pacote em seu AWS CloudShell ambiente a partir do seu clone do repositório de amostras GitHub do Deadline Cloud:

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

A diferença importante entre esse pacote de tarefas e o pacote de `job_attachments_devguide` tarefas é a adição de um novo parâmetro de tarefa no modelo de tarefa:

```
...
parameterDefinitions:
...
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
```

```
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

A `dataFlow` propriedade do parâmetro tem o valor`OUT`. O Deadline Cloud usa o valor dos parâmetros do `dataFlow` trabalho com um valor de `OUT` ou `INOUT` como resultados do seu trabalho. Se a localização do sistema de arquivos passada como um valor para esses tipos de parâmetros de trabalho for remapeada para uma localização do sistema de arquivos local no trabalhador que executa o trabalho, o Deadline Cloud procurará novos arquivos no local e os enviará para o Amazon S3 como resultados do trabalho.

Para ver como isso funciona, primeiro inicie o agente do Deadline Cloud Worker em uma AWS CloudShell guia. Permita que todos os trabalhos enviados anteriormente terminem de ser executados. Em seguida, exclua os registros de tarefas do diretório de registros:

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

Em seguida, envie um trabalho com esse pacote de trabalhos. Depois que o trabalhador estiver executando suas CloudShell execuções, veja os 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
```

O registro mostra que um arquivo foi detectado como saída e carregado no 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.
```

O registro também mostra que o Deadline Cloud criou um novo objeto de manifesto no bucket do Amazon S3 configurado para uso por anexos de trabalho na fila. Q1 O nome do objeto manifesto é derivado do farm, da fila, do trabalho, da etapa, da tarefa, do carimbo de data/hora e dos `sessionaction` identificadores da tarefa que gerou a saída. Baixe esse arquivo de manifesto para ver onde o Deadline Cloud colocou os arquivos de saída para essa tarefa:

```
# 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 .
```

O manifesto se parece com:

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

```
  ],
  "totalSize": 117
}
```

Isso mostra que o conteúdo do arquivo de saída é salvo no Amazon S3 da mesma forma que os arquivos de entrada do trabalho são salvos. Semelhante aos arquivos de entrada, o arquivo de saída é armazenado no S3 com um nome de objeto contendo o hash do arquivo e o prefixo. `DeadlineCloud/Data`

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

Você pode baixar a saída de um trabalho para sua estação de trabalho usando o monitor do Deadline Cloud ou a CLI do Deadline Cloud:

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

O valor do parâmetro do `OutputDir` trabalho no trabalho enviado é`./output_dir`, portanto, a saída é baixada para um diretório chamado `output_dir` dentro do diretório do pacote de trabalhos. Se você especificou um caminho absoluto ou um local relativo diferente como valor para`OutputDir`, os arquivos de saída seriam baixados para esse local.

```
$ 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)
```

Usando arquivos de uma etapa em uma etapa dependente

Este exemplo mostra como uma etapa em uma tarefa pode acessar as saídas de uma etapa da qual ela depende na mesma tarefa.

Para disponibilizar as saídas de uma etapa para outra, o Deadline Cloud adiciona ações adicionais a uma sessão para baixar essas saídas antes de executar tarefas na sessão. Você diz a ele de quais etapas fazer o download das saídas declarando essas etapas como dependências da etapa que precisa usar as saídas.

Use o pacote de `job_attachments_devguide_output` tarefas para este exemplo. Comece fazendo uma cópia em seu AWS CloudShell ambiente a partir do seu clone do GitHub repositório de amostras do Deadline Cloud. Modifique-a para adicionar uma etapa dependente que só é executada após a etapa existente e usa a saída dessa etapa:

```
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
```

O trabalho criado com esse pacote de trabalhos modificado é executado como duas sessões separadas, uma para a tarefa na etapa “Etapa” e outra para a tarefa na etapa "DependentStep”.

Primeiro, inicie o agente Deadline Cloud Worker em uma CloudShell guia. Deixe que todos os trabalhos enviados anteriormente terminem de ser executados e, em seguida, exclua os registros do trabalho do diretório de registros:

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

Em seguida, envie um trabalho usando o pacote de `job_attachments_devguide_output` trabalhos modificado. Espere até que ele termine de ser executado no trabalhador em seu CloudShell ambiente. Veja os registros das duas sessões:

```
# 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-*
```

No registro da sessão da tarefa na etapa denominada`DependentStep`, há duas ações de download separadas executadas:

```
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.
```

A primeira ação baixa o `script.sh` arquivo usado pela etapa chamada “Etapa”. A segunda ação baixa as saídas dessa etapa. O Deadline Cloud determina quais arquivos baixar usando o manifesto de saída gerado por essa etapa como um manifesto de entrada.

Mais tarde, no mesmo registro, você pode ver a saída da etapa chamada "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
```

Crie limites de recursos para trabalhos

Os trabalhos enviados ao Deadline Cloud podem depender de recursos compartilhados entre vários trabalhos. Por exemplo, uma fazenda pode ter mais trabalhadores do que licenças flutuantes para um recurso específico. Ou um servidor de arquivos compartilhado pode ser capaz de fornecer dados apenas para um número limitado de trabalhadores ao mesmo tempo. Em alguns casos, um ou mais trabalhos podem reivindicar todos esses recursos, causando erros devido à indisponibilidade de recursos quando novos trabalhadores são contratados.

Para ajudar a resolver isso, você pode usar limites para esses recursos restritos. O Deadline Cloud considera a disponibilidade de recursos restritos e usa essas informações para garantir que os recursos estejam disponíveis à medida que novos funcionários são iniciados, para que os trabalhos tenham uma probabilidade menor de falhar devido à indisponibilidade de recursos.

Os limites são criados para toda a fazenda. Os trabalhos enviados a uma fila só podem adquirir limites associados à fila. Se você especificar um limite para um trabalho que não esteja associado à fila, o trabalho não será compatível e não será executado.

Para usar um limite, você

- Crie um limite
- Associar um limite e uma fila
- Envie um trabalho que exija limites

> **Note**
>
> Se você executar um trabalho que tenha recursos restringidos em uma fila que não esteja associada a um limite, esse trabalho poderá consumir todos os recursos. Se você tiver um recurso restrito, certifique-se de que todas as etapas em trabalhos em filas que usam o recurso estejam associadas a um limite.

Para limites definidos em uma fazenda, associados a uma fila e especificados em um trabalho, uma das quatro coisas pode acontecer:

- Se você criar um limite, associá-lo a uma fila e especificar o limite no modelo de um trabalho, o trabalho será executado e usará somente os recursos definidos no limite.
- Se você criar um limite, especificá-lo em um modelo de trabalho, mas não associar o limite a uma fila, o trabalho será marcado como incompatível e não será executado.
- Se você criar um limite, não o associar a uma fila e não especificar o limite no modelo de um trabalho, o trabalho será executado, mas não usará o limite.
- Se você não usar nenhum limite, o trabalho será executado.

Se você associar um limite a várias filas, as filas compartilharão os recursos restringidos pelo limite. Por exemplo, se você criar um limite de 100 e uma fila estiver usando 60 recursos, outras filas poderão usar somente 40 recursos. Quando um recurso é liberado, ele pode ser usado por uma tarefa de qualquer fila.

O Deadline Cloud fornece duas AWS CloudFormation métricas para ajudar você a monitorar os recursos fornecidos por um limite. Você pode monitorar o número atual de recursos em uso e o número máximo de recursos disponíveis no limite. Para obter mais informações, consulte Métricas de limite de recursos no Guia do desenvolvedor do Deadline Cloud.

Você aplica um limite a uma etapa do trabalho em um modelo de trabalho. Quando você especifica o nome da exigência de quantidade de um limite na `amounts` seção `hostRequirements` de uma etapa e um limite com o mesmo `amountRequirementName` é associado à fila do trabalho, as tarefas agendadas para essa etapa são restringidas pelo limite do recurso.

Se uma etapa exigir um recurso limitado por um limite atingido, as tarefas dessa etapa não serão realizadas por funcionários adicionais.

Você pode aplicar mais de um limite a uma etapa do trabalho. Por exemplo, se a etapa usar duas licenças de software diferentes, você poderá aplicar um limite separado para cada licença. Se uma etapa exigir dois limites e o limite de um dos recursos for atingido, as tarefas dessa etapa não serão realizadas por outros trabalhadores até que os recursos estejam disponíveis.

Interrompendo e excluindo limites

Quando você interrompe ou exclui a associação entre uma fila e um limite, um trabalho usando o limite interrompe o agendamento de tarefas a partir de etapas que exigem esse limite e bloqueia a criação de novas sessões para uma etapa.

As tarefas que estão no estado PRONTO permanecem prontas e as tarefas são retomadas automaticamente com a associação entre a fila e o limite se tornam ativas novamente. Você não precisa recolocar nenhum trabalho na fila.

Ao interromper ou excluir a associação entre uma fila e um limite, você tem duas opções sobre como interromper a execução de tarefas:

- Pare e cancele tarefas — os trabalhadores com sessões que atingiram o limite cancelam todas as tarefas.
- Pare e conclua as tarefas em execução — os trabalhadores com sessões que atingiram o limite concluem suas tarefas.

Quando você exclui um limite usando o console, os trabalhadores primeiro param de executar tarefas imediatamente ou, eventualmente, quando elas são concluídas. Quando a associação é excluída, acontece o seguinte:

- As etapas que exigem o limite estão marcadas como não compatíveis.
- Todo o trabalho contendo essas etapas é cancelado, incluindo etapas que não exigem o limite.
- O trabalho está marcado como não compatível.

Se a fila associada ao limite tiver uma frota associada com uma capacidade de frota que corresponda ao nome da exigência de quantidade do limite, essa frota continuará processando trabalhos com o limite especificado.

Crie um limite

Você cria um limite usando o console do Deadline Cloud ou a CreateLimit operação na API Deadline Cloud. Os limites são definidos para uma fazenda, mas associados às filas. Depois de criar um limite, você pode associá-lo a uma ou mais filas.

Para criar um limite

1. No painel do console do Deadline Cloud (console do Deadline Cloud), selecione a fazenda para a qual você deseja criar uma fila.
2. Escolha a fazenda à qual adicionar o limite, escolha a guia Limites e, em seguida, escolha Criar limite.
3. Forneça os detalhes do limite. O nome do requisito de valor é o nome usado no modelo de trabalho para identificar o limite. Ele deve começar com o prefixo **`amount.`** seguido pelo nome do valor. O nome do requisito de quantia deve ser exclusivo nas filas associadas ao limite.
4. Se você escolher Definir um valor máximo, esse será o número total de recursos permitidos por esse limite. Se você escolher Sem quantidade máxima, o uso de recursos não será limitado. Mesmo quando o uso de recursos não é limitado, a CloudWatch métrica da `CurrentCount` Amazon é emitida para que você possa acompanhar o uso. Para obter mais informações, consulte CloudWatchas métricas no Guia do desenvolvedor do Deadline Cloud.
5. Se você já conhece as filas que devem usar o limite, você pode escolhê-las agora. Você não precisa associar uma fila para criar um limite.
6. Escolha Criar limite.

Associar um limite e uma fila

Depois de criar um limite, você pode associar uma ou mais filas ao limite. Somente as filas associadas a um limite usam os valores especificados no limite.

Você cria uma associação com uma fila usando o console do Deadline Cloud ou a CreateQueueLimitAssociation operação na API do Deadline Cloud.

Para associar uma fila a um limite

1. No painel do console do Deadline Cloud (console do Deadline Cloud), selecione a fazenda em que você deseja associar um limite a uma fila.
2. Escolha a guia Limites, escolha o limite ao qual associar uma fila e escolha Editar limite.

3. Na seção Associar filas, escolha as filas a serem associadas ao limite.
4. Escolha Salvar alterações.

Envie um trabalho que exija limites

Você aplica um limite especificando-o como um requisito de host para o trabalho ou etapa do trabalho. Se você não especificar um limite em uma etapa e essa etapa usar um recurso associado, o uso da etapa não será contabilizado no limite quando os trabalhos forem agendados.

Alguns remetentes do Deadline Cloud permitem que você defina um requisito de anfitrião. Você pode especificar o nome do requisito de valor do limite no remetente para aplicar o limite.

Se o remetente não aceitar a adição de requisitos de hospedagem, você também pode aplicar um limite editando o modelo de trabalho para o trabalho.

Para aplicar um limite a uma etapa do trabalho no pacote de tarefas

1. Abra o modelo de trabalho para o trabalho usando um editor de texto. O modelo de trabalho está localizado no diretório do pacote de tarefas do trabalho. Para obter mais informações, consulte Pacotes de tarefas no Guia do desenvolvedor do Deadline Cloud.
2. Encontre a definição da etapa à qual aplicar o limite.
3. Adicione o seguinte à definição da etapa. *amount.name*Substitua pelo nome do valor exigido do seu limite. Para uso típico, você deve definir o `min` valor como 1.

   YAML

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

   JSON

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

```
    }
}
```

Você pode adicionar vários limites a uma etapa do trabalho da seguinte maneira. Substitua *amount.name_1* e *amount.name_2* pelos nomes dos requisitos de valor de seus limites.

YAML

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

JSON

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

4. Salve as alterações no modelo de trabalho.

Como enviar uma vaga para o Deadline Cloud

Há muitas maneiras diferentes de enviar trabalhos para o AWS Deadline Cloud. Esta seção descreve algumas maneiras pelas quais você pode enviar trabalhos usando as ferramentas fornecidas pelo Deadline Cloud ou criando suas próprias ferramentas personalizadas para suas cargas de trabalho.

- De um terminal — para quando você está desenvolvendo um pacote de tarefas pela primeira vez ou quando os usuários que enviam uma tarefa se sentem confortáveis usando a linha de comando

- A partir de um script — para personalizar e automatizar cargas de trabalho
- De um aplicativo — para quando o trabalho do usuário está em um aplicativo ou quando o contexto de um aplicativo é importante.

Os exemplos a seguir usam a biblioteca `deadline` Python e a ferramenta de linha de `deadline` comando. Ambos estão disponíveis PyPie hospedados em GitHub.

Tópicos

- Envie um trabalho para o Deadline Cloud a partir de um terminal
- Envie um trabalho para o Deadline Cloud usando um script
- Envie uma vaga dentro de uma candidatura

Envie um trabalho para o Deadline Cloud a partir de um terminal

Usando apenas um pacote de tarefas e a CLI do Deadline Cloud, você ou seus usuários mais técnicos podem rapidamente escrever pacotes de tarefas para testar o envio de um trabalho. Use o comando a seguir para enviar um pacote de tarefas:

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

Se você enviar um pacote de tarefas com parâmetros que não tenham padrões no pacote, você poderá especificá-los com a opção /. `-p --parameter`

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

Para obter uma lista completa das opções disponíveis, execute o comando help:

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

Envie um trabalho para o Deadline Cloud usando uma GUI

A CLI do Deadline Cloud também vem com uma interface gráfica de usuário que permite que os usuários vejam os parâmetros que devem fornecer antes de enviar um trabalho. Se seus usuários preferirem não interagir com a linha de comando, você pode escrever um atalho na área de trabalho que abre uma caixa de diálogo para enviar um pacote de tarefas específico:

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

Use a `--browse` opção can para que o usuário possa selecionar um pacote de tarefas:

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

Para obter uma lista completa das opções disponíveis, execute o comando help:

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

Envie um trabalho para o Deadline Cloud usando um script

Para automatizar o envio de trabalhos para o Deadline Cloud, você pode criá-los usando ferramentas como bash, Powershell e arquivos em lote.

Você pode adicionar funcionalidades como preencher parâmetros de trabalho a partir de variáveis de ambiente ou outros aplicativos. Você também pode enviar vários trabalhos em uma linha ou criar um script para a criação de um pacote de trabalhos para envio.

Envie um trabalho usando Python

O Deadline Cloud também tem uma biblioteca Python de código aberto para interagir com o serviço. O código-fonte está disponível em GitHub.

A biblioteca está disponível em pypi via pip (). `pip install deadline` É a mesma biblioteca usada pela ferramenta CLI do 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 criar uma caixa de diálogo como o `deadline bundle gui-submit` comando, você pode usar a `show_job_bundle_submitter` função do `deadline.client.ui.job_bundle_submitter`.

O exemplo a seguir inicia um aplicativo Qt e mostra o remetente do pacote de tarefas:

```
# 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 criar seu próprio diálogo, você pode usar a `SubmitJobToDeadlineDialog` classe em `deadline.client.ui.dialogs.submit_job_to_deadline_dialog`. Você pode transmitir valores, incorporar sua própria guia específica do trabalho e determinar como o pacote de trabalhos é criado (ou transmitido).

Envie uma vaga dentro de uma candidatura

Para facilitar o envio de trabalhos pelos usuários, você pode usar os tempos de execução de scripts ou os sistemas de plug-ins fornecidos por um aplicativo. Os usuários têm uma interface familiar e você pode criar ferramentas poderosas que os auxiliam no envio de uma carga de trabalho.

Incorporar pacotes de tarefas em um aplicativo

Este exemplo demonstra o envio de pacotes de tarefas que você disponibiliza no aplicativo.

Para dar ao usuário acesso a esses pacotes de tarefas, crie um script incorporado em um item de menu que inicie a CLI do Deadline Cloud.

O script a seguir permite que o usuário selecione o pacote de tarefas:

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

Em vez disso, para usar um pacote de tarefas específico em um item de menu, use o seguinte:

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

Isso abre uma caixa de diálogo na qual o usuário pode modificar os parâmetros, entradas e saídas do trabalho e, em seguida, enviar o trabalho. Você pode ter itens de menu diferentes para pacotes de tarefas diferentes para um usuário enviar em uma inscrição.

Se o trabalho enviado com um pacote de trabalhos contiver parâmetros e referências de ativos semelhantes em todos os envios, você poderá preencher os valores padrão no pacote de trabalhos subjacente.

Obtenha informações de um aplicativo

Para extrair informações de um aplicativo para que os usuários não precisem adicioná-las manualmente ao envio, você pode integrar o Deadline Cloud ao aplicativo para que seus usuários possam enviar trabalhos usando uma interface familiar sem precisar sair do aplicativo ou usar ferramentas de linha de comando.

Se seu aplicativo tiver um tempo de execução de script compatível com Python e pyside/pyqt, você poderá usar os componentes da GUI da biblioteca de cliente do Deadline Cloud para criar uma interface de usuário. Por exemplo, consulte Integração do Deadline Cloud para Maya em GitHub.

A biblioteca cliente do Deadline Cloud fornece operações que fazem o seguinte para ajudar você a fornecer uma experiência de usuário forte e integrada:

- Extraia parâmetros de ambiente de fila, parâmetros de trabalho e referências de ativos a partir de variáveis de ambiente e chamando o SDK do aplicativo.
- Defina os parâmetros no pacote de tarefas. Para evitar a modificação do pacote original, você deve fazer uma cópia do pacote e enviar a cópia.

Se você usar o `deadline bundle gui-submit` comando para enviar o pacote de tarefas, deverá programaticamente os `asset_references.yaml` arquivos `parameter_values.yaml` e para transmitir as informações do aplicativo. Para obter mais informações sobre esses arquivos, consulteModelos de Open Job Description (OpenJD) para Deadline Cloud.

Se você precisar de controles mais complexos do que os oferecidos pelo OpenJD, precisar abstrair o trabalho do usuário ou quiser que a integração corresponda ao estilo visual do aplicativo, você pode escrever sua própria caixa de diálogo que chama a biblioteca cliente do Deadline Cloud para enviar o trabalho.

Agende trabalhos no Deadline Cloud

Depois de criar um trabalho, AWS o Deadline Cloud o programa para ser processado em uma ou mais frotas associadas a uma fila. A frota que processa uma tarefa específica é escolhida com base na configuração de agendamento, nos recursos configurados para a frota e nos requisitos do host de uma etapa específica.

As seções a seguir fornecem detalhes do processo de agendamento de um trabalho.

Configurações de agendamento

Você pode configurar como o Deadline Cloud agenda trabalhos em uma fila definindo uma configuração de agendamento na fila. A configuração de agendamento controla como os trabalhadores são distribuídos entre os trabalhos.

Você pode definir a configuração de agendamento usando o console do Deadline Cloud ou ligando para o CreateQueueou UpdateQueue APIs.

Há três configurações de agendamento disponíveis:

- Prioridade, first-in-first-out (`priorityFifo`) — Agenda a tarefa de maior prioridade e a primeira a ser enviada primeiro (padrão).
- Prioritário, equilibrado (`priorityBalanced`) — Distribui os trabalhadores uniformemente entre os trabalhos com a maior prioridade.
- Ponderado, balanceado (`weightedBalanced`) — Usa uma fórmula ponderada para determinar como os trabalhadores são distribuídos entre os trabalhos.

Em todas as configurações de agendamento, as tarefas em andamento são executadas até a conclusão antes que uma nova decisão de agendamento seja tomada. Se você alterar a configuração de agendamento enquanto as tarefas estiverem em execução, a alteração se aplicará somente quando os trabalhadores forem designados em seguida. As tarefas em execução não são interrompidas nem reatribuídas.

Prioridade, first-in-first-out

Priority, first-in-first-out (`priorityFifo`) é a configuração de agendamento padrão para novas filas. O Deadline Cloud atribui primeiro aos trabalhadores o trabalho de maior prioridade. Quando vários trabalhos compartilham a mesma prioridade, o trabalho mais antigo (enviado mais cedo) recebe primeiro todos os trabalhadores disponíveis.

Use FIFO prioritário quando quiser uma ordenação estrita de trabalhos. Essa configuração é apropriada quando os trabalhos devem ser concluídos um por vez na ordem em que foram enviados, como estágios sequenciais do pipeline ou processamento em lote, em que cada trabalho deve ser concluído antes do início do próximo.

Essa configuração não tem parâmetros adicionais.

Prioritário, equilibrado

Priority, balanced (`priorityBalanced`) distribui os trabalhadores uniformemente em todos os trabalhos no nível de prioridade mais alto. Quando existe apenas um trabalho com a prioridade mais alta, o Deadline Cloud atribui todos os trabalhadores a esse trabalho. Quando vários trabalhos compartilham a maior prioridade, os trabalhadores são divididos igualmente entre eles. Se os trabalhadores não puderem ser divididos uniformemente, os trabalhadores extras serão distribuídos entre os empregos de maior prioridade.

Use o balanceamento de prioridade quando vários artistas ou usuários enviarem trabalhos com a mesma prioridade e cada usuário precisar de feedback imediato. Essa configuração garante que nenhum trabalho monopolize todos os trabalhadores disponíveis, de modo que todos os usuários recebam trabalhadores logo após o envio.

Se um trabalho tiver menos tarefas restantes do que sua parcela de trabalhadores, os trabalhadores excedentes serão redistribuídos para outros empregos no mesmo nível de prioridade. Se todos os trabalhos de maior prioridade forem totalmente alocados, os trabalhadores excedentes se transformarão em empregos no próximo nível de prioridade mais alta.

Essa configuração tem o seguinte parâmetro:

`renderingTaskBuffer`

Controla a aderência do trabalhador. Um trabalhador muda de seu trabalho atual para outro trabalho com a mesma prioridade somente se a diferença nas tarefas de renderização exceder o

`renderingTaskBuffer` valor. Um valor mais alto mantém os trabalhadores em seus empregos atuais por mais tempo, reduzindo a mudança de contexto. O valor padrão é 1.

Ponderado, equilibrado

Weighted, balanced (`weightedBalanced`) usa uma fórmula para calcular um peso para cada trabalho. O Deadline Cloud atribui primeiro aos trabalhadores o trabalho de maior peso. Se vários trabalhos tiverem o mesmo peso, os trabalhadores serão distribuídos entre eles.

Use a balança ponderada quando precisar de um controle refinado sobre como os trabalhadores são distribuídos em trabalhos com prioridades, taxas de erro e tempos de envio variados. Essa configuração é apropriada para ambientes complexos de render farm em que você deseja ajustar o equilíbrio entre a prioridade do trabalho, a idade do trabalho, o tratamento de erros e a fidelidade do trabalhador.

O peso de cada trabalho é calculado da seguinte forma:

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

O `renderingTaskBuffer` componente é aplicado somente se o trabalhador estiver trabalhando atualmente no trabalho. Geralmente, `renderingTaskWeight` é definido como um valor negativo para que os trabalhos com trabalhadores designados recebam um peso menor, colocando outros trabalhos na frente da fila. Geralmente, também `errorWeight` é negativo, de modo que os trabalhos com erros não são priorizados. Você pode usar substituições de agendamento para trabalhos de prioridade mínima e máxima.

Essa configuração tem os seguintes parâmetros:

`priorityWeight`

O peso aplicado à prioridade de um trabalho. Um valor positivo significa que os trabalhos de maior prioridade são agendados primeiro. O valor padrão é `100.0`. Alcance: `0` até`10000`.

`errorWeight`

O peso aplicado à contagem de erros de uma tarefa. Um valor negativo significa que trabalhos sem erros são agendados primeiro. O valor padrão é `-10.0`. Alcance: `-10000` até`10000`.

`submissionTimeWeight`

O peso aplicado ao tempo de envio de um trabalho (em segundos). Um valor positivo significa que os trabalhos enviados anteriormente são agendados primeiro. O valor padrão é `3.0`. Alcance: `0` até`10000`.

`renderingTaskWeight`

O peso aplicado ao número de tarefas atualmente renderizadas para um trabalho. Um valor negativo significa que trabalhos com menos trabalhadores serão programados em seguida. O valor padrão é `-100.0`. Alcance: `-10000` até`10000`.

`renderingTaskBuffer`

O número de tarefas de renderização antes que o peso da tarefa de renderização entre em vigor. Um valor positivo mantém os trabalhadores em seus empregos atuais. O valor padrão é `1`. Alcance: `0` até`1000`.

`maxPriorityOverride`

Opcional. Quando definido como`alwaysScheduleFirst`, os trabalhos com prioridade máxima (100) são sempre programados antes dos outros trabalhos, independentemente da fórmula ponderada. Quando vários trabalhos têm a prioridade máxima, os empates são quebrados usando a fórmula ponderada padrão. Quando a substituição está ausente, os trabalhos de prioridade máxima usam a fórmula ponderada padrão sem tratamento especial.

`minPriorityOverride`

Opcional. Quando definido como`alwaysScheduleLast`, os trabalhos com prioridade mínima (0) são sempre programados após outros trabalhos, independentemente da fórmula ponderada. Quando vários trabalhos têm a prioridade mínima, os empates são quebrados usando a fórmula ponderada padrão. Quando a substituição está ausente, os trabalhos de prioridade mínima usam a fórmula ponderada padrão sem tratamento especial.

Determine a compatibilidade da frota

Depois de criar um trabalho, o Deadline Cloud verifica os requisitos do host para cada etapa do trabalho em relação aos recursos das frotas associadas à fila para a qual o trabalho foi enviado. Se uma frota atender aos requisitos do anfitrião, o trabalho será colocado no `READY` estado.

Se alguma etapa do trabalho tiver requisitos que não possam ser atendidos por uma frota associada à fila, o status da etapa será definido como. `NOT_COMPATIBLE` Além disso, as demais etapas do trabalho são canceladas.

As capacidades de uma frota são definidas no nível da frota. Mesmo que um trabalhador em uma frota atenda aos requisitos do trabalho, ele não receberá tarefas do trabalho se sua frota não atender aos requisitos do trabalho.

O modelo de tarefa a seguir tem uma etapa que especifica os requisitos de host para a etapa:

```
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
```

Esse trabalho pode ser programado para uma frota com os seguintes recursos:

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

Esse trabalho não pode ser programado para uma frota com nenhum dos seguintes recursos:

```
{
    "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.
```

Dimensionamento da frota

Quando um trabalho é atribuído a uma frota compatível com gerenciamento de serviços, a frota é escalada automaticamente. O número de trabalhadores na frota muda com base no número de tarefas disponíveis para a frota executar.

Quando um trabalho é atribuído a uma frota gerenciada pelo cliente, os trabalhadores podem já existir ou podem ser criados usando o escalonamento automático baseado em eventos. Para obter mais informações, consulte Use EventBridge para lidar com eventos de auto scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Sessões

As tarefas em um trabalho são divididas em uma ou mais sessões. Os trabalhadores executam as sessões para configurar o ambiente, executar as tarefas e, em seguida, destruir o ambiente. Cada sessão é composta por uma ou mais ações que um trabalhador deve realizar.

Quando um trabalhador conclui as ações da seção, ações adicionais da sessão podem ser enviadas ao trabalhador. O funcionário reutiliza os ambientes e os anexos de trabalho existentes na sessão para concluir as tarefas com mais eficiência.

Em trabalhadores de frotas gerenciados por serviços, os diretórios de sessão são excluídos após o término da sessão, mas outros diretórios são mantidos entre as sessões. Esse comportamento permite que você implemente estratégias de armazenamento em cache para dados que podem ser reutilizados em várias sessões. Para armazenar dados em cache entre as sessões, armazene-os no diretório inicial do usuário que está executando o trabalho. Por exemplo, os pacotes conda são armazenados em cache no diretório inicial do usuário do trabalho em `C:\Users\job-user\.conda-pkgs` on Windows workers e `/home/job-user/.conda-pkgs` on Linux workers. Esses dados permanecem disponíveis até que o trabalhador seja desligado.

Os anexos de trabalho são criados pelo remetente e você usa como parte do pacote de trabalhos da CLI do Deadline Cloud. Você também pode criar anexos de trabalho usando a `--attachments` opção do comando. `create-job` AWS CLI Os ambientes são definidos em dois lugares: ambientes de fila anexados a uma fila específica e ambientes de tarefas e etapas definidos no modelo de trabalho.

Há quatro tipos de ação de sessão:

- `syncInputJobAttachments`— Faz o download dos anexos do trabalho de entrada para o trabalhador.
- `envEnter`— Executa as `onEnter` ações para um ambiente.
- `taskRun`— Executa as `onRun` ações de uma tarefa.
- `envExit`— Executa as `onExit` ações para um ambiente.

O modelo de trabalho a seguir tem um ambiente de etapas. Ele tem uma `onEnter` definição para configurar o ambiente de etapas, uma `onRun` definição que define a tarefa a ser executada e uma `onExit` definição para derrubar o ambiente de etapas. As sessões criadas para esse trabalho incluirão uma `envEnter` ação, uma ou mais `taskRun` ações e, em seguida, uma `envExit` ação.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
      actions:
        onEnter:
          command: MayaAdaptor
          args:
          - daemon
          - start
          - --init-data
          - file://{{Env.File.initData}}
        onExit:
          command: MayaAdaptor
          args:
          - daemon
          - stop
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      range: 1-5
      type: INT
  script:
    embeddedFiles:
    - name: runData
      filename: run-data.yaml
      type: TEXT
      data: |
        frame: {{Task.Param.Frame}}
    actions:
      onRun:
```

```
        command: MayaAdaptor
        args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

Pipelining de ações de sessão

O pipeline de ações de sessão permite que um agendador pré-atribua várias ações de sessão a um trabalhador. O trabalhador pode então executar essas ações sequencialmente, reduzindo ou eliminando o tempo ocioso entre as tarefas.

Para criar uma atribuição inicial, o agendador cria uma sessão com uma tarefa, o trabalhador conclui a tarefa e, em seguida, o agendador analisa a duração da tarefa para determinar as atribuições futuras.

Para que o agendador seja eficaz, existem regras de duração da tarefa. Para tarefas com menos de um minuto, o agendador usa um padrão de crescimento de potência de 2. Por exemplo, para uma tarefa de 1 segundo, o agendador atribui 2 novas tarefas, depois 4 e depois 8. Para tarefas com mais de um minuto, o agendador atribui apenas uma nova tarefa e o pipeline permanece desativado.

Para calcular o tamanho do pipeline, o programador faz o seguinte:

- Usa a duração média das tarefas concluídas
- Visa manter o trabalhador ocupado por um minuto
- Considera somente tarefas dentro da mesma sessão
- Não compartilha dados de duração entre os trabalhadores

Com o pipeline de ações da sessão, os trabalhadores iniciam novas tarefas imediatamente e não há tempo de espera entre as solicitações do agendador. Ele também fornece maior eficiência do trabalhador e melhor distribuição de tarefas para processos de longa execução.

Além disso, se houver um novo trabalho de maior prioridade disponível, o trabalhador concluirá todo o trabalho anteriormente atribuído antes que a sessão atual termine e uma nova sessão de um trabalho de maior prioridade seja atribuída.

Dependências de etapas

O Deadline Cloud suporta a definição de dependências entre as etapas para que uma etapa espere até que outra seja concluída antes de começar. Você pode definir mais de uma dependência para uma etapa. Uma etapa com uma dependência não é agendada até que todas as dependências estejam concluídas.

Se o modelo de trabalho definir uma dependência circular, o trabalho será rejeitado e o status do trabalho será definido como. `CREATE_FAILED`

O modelo de trabalho a seguir cria um trabalho com duas etapas. `StepB`depende de`StepA`. `StepB`só é executado após `StepA` ser concluído com sucesso.

Depois que o trabalho é criado, `StepA` está no `READY` estado e `StepB` está no `PENDING` estado. Depois de `StepA` terminar, `StepB` se muda para o `READY` estado. Se `StepA` falhar ou `StepA` for cancelado, `StepB` passa para o `CANCELED` estado.

Você pode definir uma dependência em várias etapas. Por exemplo, `StepC` depende de ambos `StepA` e`StepB`, `StepC` não começará até que as outras duas etapas terminem.

As dependências de etapas têm as seguintes restrições:

- Dependências por etapa — Uma etapa pode depender de no máximo 128 outras etapas.
- Consumidores por etapa — No máximo 32 outras etapas podem depender de uma única etapa.

```
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 um trabalho no Deadline Cloud

Você pode usar os seguintes `update` comandos AWS Command Line Interface (AWS CLI) para modificar a configuração de um trabalho ou definir o status de destino de um trabalho, etapa ou tarefa:

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

Nos exemplos de `update` comandos a seguir, substitua cada um *`user input placeholder`* por suas próprias informações.

Example— Solicitar um trabalho

Todas as tarefas na tarefa mudam para o `READY` status, a menos que haja dependências de etapas. As etapas com dependências mudam para uma `READY` ou à `PENDING` medida que são restauradas.

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

Example— Cancelar um trabalho

Todas as tarefas no trabalho que não têm o status `SUCCEEDED` ou `FAILED` estão marcadas`CANCELED`.

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

Example— Marcar que um trabalho falhou

Todas as tarefas no trabalho que têm o status permanecem `SUCCEEDED` inalteradas. Todas as outras tarefas estão marcadas`FAILED`.

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

Example— Marque um trabalho bem-sucedido

Todas as tarefas do trabalho são transferidas para o `SUCCEEDED` estado.

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

Example— Suspender um emprego

As tarefas no trabalho no `FAILED` estado `SUCCEEDEDCANCELED`, ou não mudam. Todas as outras tarefas estão marcadas`SUSPENDED`.

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

Example— Mudar a prioridade de um trabalho

Atualiza a prioridade de um trabalho em uma fila para alterar a ordem em que ele está agendado. Os trabalhos de maior prioridade geralmente são agendados primeiro.

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

Example— Alterar o número de tarefas com falha permitidas

Atualiza o número máximo de tarefas com falha que o trabalho pode ter antes que as tarefas restantes sejam canceladas.

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

Example— Alterar o número de novas tentativas de tarefas permitidas

Atualiza o número máximo de tentativas de uma tarefa antes que a tarefa falhe. Uma tarefa que tenha atingido o número máximo de novas tentativas não pode ser colocada novamente na fila até que esse valor seja aumentado.

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

```
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example— Arquivar um trabalho

Atualiza o status do ciclo de vida do trabalho para. ARCHIVED Os trabalhos arquivados não podem ser agendados nem modificados. Você só pode arquivar um trabalho que esteja no SUSPENDED estado FAILEDCANCELED,SUCCEEDED,, ou.

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

Example— Alterar o nome de um emprego

Atualiza o nome de exibição de um trabalho. O nome do trabalho pode ter até 128 caracteres.

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

Example— Alterar a descrição de um trabalho

Atualiza a descrição de um trabalho. A descrição pode ter até 2048 caracteres. Para remover a descrição existente, passe uma string vazia.

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

Example— Recoloque uma etapa na fila

Todas as tarefas na etapa mudam para o READY estado, a menos que haja dependências de etapas. As tarefas em etapas com dependências mudam para READY ouPENDING, e a tarefa é restaurada.

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

Example— Cancelar uma etapa

Todas as tarefas na etapa que não têm o status `SUCCEEDED` ou `FAILED` estão 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 uma etapa que falhou

Todas as tarefas na etapa que têm o status permanecem `SUCCEEDED` inalteradas. Todas as outras tarefas estão marcadas`FAILED`.

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

Example— Marque uma etapa bem-sucedida

Todas as tarefas na etapa estão 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 uma etapa

As tarefas na etapa do `FAILED` estado `SUCCEEDED``CANCELED`,, ou não mudam. Todas as outras tarefas estão marcadas`SUSPENDED`.

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

Example— Alterar o status de uma tarefa

Quando você usa o comando da CLI do `update-task` Deadline Cloud, a tarefa muda para o status 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
```

Crie e use frotas gerenciadas pelo cliente do Deadline Cloud

Ao criar uma frota gerenciada pelo cliente (CMF), você tem controle total sobre seu pipeline de processamento. Você define a rede e o ambiente de software para cada trabalhador. O Deadline Cloud atua como repositório e agendador de seus trabalhos.

Um trabalhador pode ser uma instância do Amazon Elastic Compute Cloud (Amazon EC2), um trabalhador em uma instalação de co-location ou um trabalhador local. Cada funcionário deve executar o agente de trabalho do Deadline Cloud. Todos os trabalhadores devem ter acesso ao endpoint do serviço Deadline Cloud.

Os tópicos a seguir mostram como criar um CMF básico usando instâncias do Amazon EC2.

Tópicos

Crie uma frota gerenciada pelo cliente

Para criar uma frota gerenciada pelo cliente (CMF), conclua as etapas a seguir.

Deadline Cloud console

Para usar o console do Deadline Cloud para criar uma frota gerenciada pelo cliente

1. Abra o console do Deadline Cloud.
2. Selecione Fazendas. Uma lista das fazendas disponíveis é exibida.
3. Selecione o nome da Fazenda na qual você deseja trabalhar.

4. Selecione a guia Frotas e, em seguida, escolha Criar frota.
5. Insira um nome para sua frota.
6. (Opcional) Insira uma descrição para sua frota.
7. Selecione Gerenciado pelo cliente para o tipo de frota.
8. Selecione o acesso ao serviço da sua frota.

    a. Recomendamos usar a opção Criar e usar uma nova função de serviço para cada frota para um controle de permissões mais granular. Essa opção é selecionada por padrão.

    b. Você também pode usar uma função de serviço existente selecionando Escolher uma função de serviço.
9. Revise suas seleções e escolha Avançar.
10. Selecione um sistema operacional para sua frota. Todos os trabalhadores de uma frota devem ter um sistema operacional comum.
11. Selecione a arquitetura da CPU do host.
12. Selecione os recursos mínimos e máximos de vCPU e hardware de memória para atender às demandas de carga de trabalho de suas frotas.
13. Selecione um tipo de Auto Scaling. Para obter mais informações, consulte Usar EventBridge para lidar com eventos do Auto Scaling.

    - Sem escalabilidade: você está criando uma frota local e quer optar por não participar do Deadline Cloud Auto Scaling.
    - Recomendações de escalabilidade: Você está criando uma frota do Amazon Elastic Compute Cloud (Amazon EC2).
14. (Opcional) Selecione a seta para expandir a seção Adicionar recursos.
15. (Opcional) Marque a caixa de seleção Adicionar capacidade de GPU - Opcional e, em seguida, insira o mínimo, o máximo GPUs e a memória.
16. Revise suas seleções e escolha Avançar.
17. (Opcional) Defina os recursos personalizados do trabalhador e escolha Avançar.
18. Usando o menu suspenso, selecione uma ou mais filas para associar à frota.

> **Note**
>
> Recomendamos associar uma frota somente a filas que estejam todas no mesmo limite de confiança. Essa recomendação garante um forte limite de segurança entre a execução de trabalhos no mesmo trabalhador.

19. Revise as associações de filas e escolha Avançar.
20. (Opcional) Para o ambiente de fila padrão do Conda, criaremos um ambiente para sua fila que instalará pacotes conda solicitados por jobs.

> **Note**
>
> O ambiente conda queue é usado para instalar pacotes conda solicitados por jobs. Normalmente, você deve desmarcar o ambiente conda queue nas filas associadas a, CMFs pois CMFs não terá os comandos conda necessários instalados por padrão.

21. (Opcional) Adicione tags ao seu CMF. Para obter mais informações, consulte Como marcar seus AWS recursos.
22. Revise a configuração da sua frota e faça as alterações, depois escolha Criar frota.
23. Selecione a guia Frotas e anote a ID da frota.

AWS CLI

Para usar o AWS CLI para criar uma frota gerenciada pelo cliente

1. Abra um terminal.
2. Crie `fleet-trust-policy.json` em um novo editor.

   a. Adicione a seguinte política do IAM, substituindo o *ITALICIZED* texto pelo ID da sua AWS conta e pelo ID do farm do 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. Salve as alterações.

3. Criar `fleet-policy.json`.

   a. Adicione a seguinte política do 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. Salve as alterações.

4. Adicione uma função do IAM para os trabalhadores da sua frota usarem.

```
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. Criar `create-fleet-request.json`.

   a. Adicione a seguinte política do IAM, substituindo o texto em ITÁLICO pelos valores do CMF.

> **Note**
>
> Você pode encontrar o *ROLE_ARN* no `create-cmf-fleet.json`.
> Para o *OS_FAMILY*, você deve escolher um dos `linux`, `macos` ou `windows`.

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
}
```

b. Salve as alterações.

6. Crie sua frota.

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

Instalação e configuração do host de trabalho

Um host de trabalho se refere a uma máquina host que executa um funcionário do Deadline Cloud. Esta seção explica como configurar o host de trabalho e configurá-lo de acordo com suas necessidades específicas. Cada host de trabalho executa um programa chamado agente de trabalho. O agente do trabalhador é responsável por:

- Gerenciando o ciclo de vida do trabalhador.
- Sincronizando o trabalho atribuído, seu progresso e resultados.
- Monitorando o trabalho em execução.
- Encaminhando registros para destinos configurados.

Recomendamos que você use o agente de trabalho do Deadline Cloud fornecido. O agente de trabalho é de código aberto e incentivamos solicitações de recursos, mas você também pode desenvolver e personalizar para atender às suas necessidades.

Para concluir as tarefas nas seções a seguir, você precisa do seguinte:

Linux

- Uma instância Linux baseada no Amazon Elastic Compute Cloud (Amazon EC2). Recomendamos o Amazon Linux 2023.
- `sudo`privilégios
- Python 3.9 ou posterior

Windows

- Uma instância Windows baseada no Amazon Elastic Compute Cloud (Amazon EC2). Nós recomendamosWindows Server 2022.
- Acesso do administrador ao host do trabalhador
- Python 3.9 ou superior instalado para todos os usuários

Crie e configure um ambiente virtual Python

Você pode criar um ambiente virtual Python Linux se tiver instalado o Python 3.9 ou superior e o colocado no seu. PATH

> **Note**
>
> AtivadoWindows, os arquivos do agente devem ser instalados no diretório global de pacotes de sites do Python. Atualmente, não há suporte para ambientes virtuais Python.

Para criar e ativar um ambiente virtual Python

1. Abra um terminal como `root` usuário (ou use`sudo`/`su`).
2. Crie e ative um ambiente virtual Python.

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

Instale o agente Deadline Cloud Worker

Depois de configurar seu Python e criar um ambiente virtualLinux, instale os pacotes Python do agente Deadline Cloud Worker.

Para instalar os pacotes Python do agente de trabalho

Linux

1. Abra um terminal como `root` usuário (ou use`sudo`/`su`).
2. Baixe e instale os pacotes do agente Deadline Cloud Worker do PyPI:

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

Windows

1. Abra um prompt de comando ou PowerShell terminal do administrador.

2. Baixe e instale os pacotes do agente Deadline Cloud Worker do PyPI:

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

Quando seu host de Windows trabalho exige nomes de caminhos longos (maiores que 250 caracteres), você deve habilitar nomes de caminhos longos da seguinte forma:

Para habilitar caminhos longos para anfitriões de Windows trabalhadores

1. Certifique-se de que a chave de registro de caminho longo esteja habilitada. Para obter mais informações, consulte Configuração do registro para habilitar caminhos de log no site da Microsoft.
2. Instale o Windows SDK para aplicativos C++ x86 para desktop. Para obter mais informações, consulte WindowsSDK no Windows Dev Center.
3. Abra o local de instalação do Python em seu ambiente em que o agente de trabalho está instalado. O padrão é `C:\Program Files\Python311`. Há um arquivo executável chamado`pythonservice.exe`.
4. Crie um novo arquivo chamado `pythonservice.exe.manifest` no mesmo local. Adicione o seguinte:

```
<?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 um prompt de comando e execute o comando a seguir no local do arquivo de manifesto que você criou:

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

Você deve ver uma saída semelhante a:

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

O trabalhador agora pode acessar caminhos longos. Para limpar, remova o `pythonservice.exe.manifest` arquivo e desinstale o SDK.

Configurar o agente Deadline Cloud Worker

Você pode definir as configurações do agente Deadline Cloud Worker de três maneiras. Recomendamos que você use a configuração do sistema operacional executando a `install-deadline-worker` ferramenta.

O agente de trabalho não oferece suporte à execução como usuário de domínio no Windows. Para executar um trabalho como usuário de domínio, você pode especificar uma conta de usuário de domínio ao configurar um usuário de fila para executar trabalhos. Para obter mais informações, consulte a etapa 7 em Filas do Deadline Cloud no Guia do usuário do AWS Deadline Cloud.

Argumentos da linha de comando — Você pode especificar argumentos ao executar o agente de trabalho do Deadline Cloud na linha de comando. Algumas configurações não estão disponíveis por meio de argumentos de linha de comando. Para ver todos os argumentos de linha de comando disponíveis, digite`deadline-worker-agent --help`.

Variáveis de ambiente — Você pode configurar o agente de trabalho do Deadline Cloud definindo a variável de ambiente começando com`DEADLINE_WORKER_`. Por exemplo, para ver todos os argumentos de linha de comando disponíveis, você pode usar `export DEADLINE_WORKER_VERBOSE=true` para definir a saída do agente de trabalho como detalhada. Para obter mais exemplos e informações, consulte `/etc/amazon/deadline/worker.toml.example` on Linux or `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` onWindows.

Arquivo de configuração — Quando você instala o agente de trabalho, ele cria um arquivo de configuração localizado em `/etc/amazon/deadline/worker.toml` on Linux ou `C:\ProgramData\Amazon\Deadline\Config\worker.toml` onWindows. O agente de trabalho carrega esse arquivo de configuração quando ele é iniciado. Você pode usar o arquivo de configuração de exemplo (`/etc/amazon/deadline/worker.toml.example`ativado Linux ou `C:`

`\ProgramData\Amazon\Deadline\Config\worker.toml.example` ativadoWindows) para adaptar o arquivo de configuração padrão do agente de trabalho às suas necessidades específicas.

Por fim, recomendamos que você ative o desligamento automático do agente de trabalho depois que o software for implantado e funcionar conforme o esperado. Isso permite que a frota de trabalhadores aumente quando necessário e seja encerrada quando um trabalho for concluído. O escalonamento automático ajuda a garantir que você esteja usando apenas os recursos necessários. Para permitir que uma instância iniciada pelo grupo de auto scaling seja encerrada, você deve adicioná-la `shutdown_on_stop=true` ao arquivo de `worker.toml` configuração.

Para ativar o desligamento automático

Como **`root`** usuário:

- Instale o agente de trabalho com parâmetros**`--allow-shutdown`**.

  Linux

  Insira:

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

  Windows

  Insira:

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

Crie usuários e grupos de trabalho

Esta seção descreve o relacionamento necessário de usuário e grupo entre o usuário do agente e o `jobRunAsUser` definido em suas filas.

O agente do Deadline Cloud Worker deve ser executado como um usuário dedicado específico do agente no host. Você deve configurar a `jobRunAsUser` propriedade das filas do Deadline Cloud para que os trabalhadores executem os trabalhos de fila como um usuário e grupo específicos do sistema operacional. Essa configuração significa que você pode controlar as permissões compartilhadas do sistema de arquivos que seus trabalhos têm. Ele também fornece um importante limite de segurança entre seus trabalhos e o usuário do agente de trabalho.

Linuxusuários e grupos de trabalho

Para configurar um usuário de agente de trabalho local e `jobRunAsUser` garantir que você atenda aos seguintes requisitos. Se você estiver usando um Linux Pluggable Authentication Module (PAM), como Active Directory ou LDAP, seu procedimento pode ser diferente.

O usuário do agente de trabalho e o `jobRunAsUser` grupo compartilhado são definidos quando você instala o agente de trabalho. Os padrões são `deadline-worker-agent` e`deadline-job-users`, mas você pode alterá-los ao instalar o agente de trabalho.

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

Os comandos devem ser executados como usuário root.

- Cada um `jobRunAsUser` deve ter um grupo primário correspondente. Criar um usuário com o `adduser` comando geralmente cria um grupo primário correspondente.

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

- O grupo primário do `jobRunAsUser` é um grupo secundário para o usuário do agente de trabalho. O grupo compartilhado permite que o agente de trabalho disponibilize arquivos para a tarefa enquanto ela está sendo executada.

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

- `jobRunAsUser`Deve ser membro do grupo de trabalho compartilhado.

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

- Eles não `jobRunAsUser` devem pertencer ao grupo principal do usuário do agente de trabalho. Arquivos confidenciais gravados pelo agente de trabalho pertencem ao grupo principal do agente.

Se `jobRunAsUser` a fizer parte desse grupo, os arquivos do agente de trabalho poderão estar acessíveis aos trabalhos em execução no trabalhador.

- O padrão Região da AWS deve corresponder à região da fazenda à qual o trabalhador pertence. Isso deve ser aplicado a todas as `jobRunAsUser` contas do trabalhador.

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

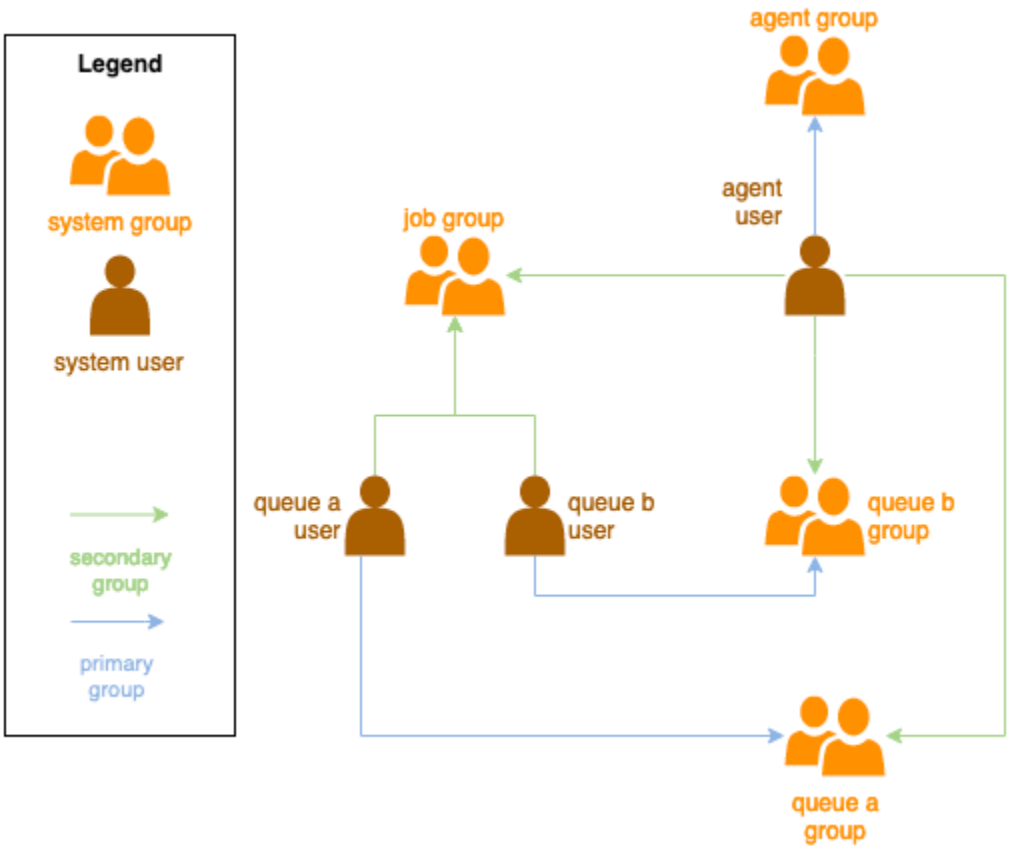
- O usuário do agente de trabalho deve ser capaz de executar `sudo` comandos como `jobRunAsUser` o. Execute o comando a seguir para abrir um editor e criar uma nova regra sudoers:

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

Adicione o seguinte ao arquivo:

```
# 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
```

O diagrama a seguir ilustra a relação entre o usuário agente e os `jobRunAsUser` usuários e grupos das filas associadas à frota.



Usuários do Windows

Para usar um Windows usuário como `ojobRunAsUser`, ele deve atender aos seguintes requisitos:

- Todos os `jobRunAsUser` usuários da fila devem existir.
- Suas senhas devem corresponder ao valor do segredo especificado no `JobRunAsUser` campo da fila. Para obter instruções, consulte a etapa 7 em Filas do Deadline Cloud no Guia do usuário do AWS Deadline Cloud.
- O usuário-agente deve ser capaz de fazer login como esses usuários.

Protegendo seu host de trabalho

Ao configurar seu host de trabalho, siga as melhores práticas de segurança para proteger informações confidenciais e manter os controles de acesso adequados.

Configurando as permissões da pasta de registro

O agente de trabalho grava arquivos de log que podem conter informações confidenciais dos scripts de configuração do host e da execução do trabalho. O `install-deadline-worker` comando cria o diretório de log com permissões seguras. Se você precisar criar o diretório manualmente antes da instalação, use os procedimentos a seguir para combinar as permissões usadas pelas frotas gerenciadas por serviços:

Linux

Para configurar as permissões do diretório de registros em Linux

1. Crie o diretório de log:

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

2. Defina o proprietário e o grupo como o usuário do agente de trabalho:

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

3. Defina as permissões para 750:

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

Essas permissões garantem que somente o usuário e o grupo do agente de trabalho possam acessar os arquivos de log, impedindo que usuários do trabalho e outros usuários não autorizados leiam informações potencialmente confidenciais.

Windows

Para configurar as permissões do diretório de registros em Windows

1. Abra um PowerShell terminal de administrador.
2. Crie o diretório de log:

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

3. Configure ACLs restrito para permitir somente o usuário do agente de trabalho e os 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
```

Esses comandos restringem o acesso ao diretório de log somente ao usuário do agente de trabalho e ao grupo Administradores, impedindo que usuários do trabalho e outros usuários não autorizados leiam informações potencialmente confidenciais.

Gerencie o acesso aos segredos dos usuários do Windows trabalho

Ao configurar uma fila com um Windows`jobRunAsUser`, você deve especificar um segredo do AWS Secrets Manager. Espera-se que o valor desse segredo seja um objeto codificado em JSON do formato:

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

Para que os trabalhadores executem trabalhos conforme a fila está configurada`jobRunAsUser`, a função do IAM da frota deve ter permissões para obter o valor do segredo. Se o segredo for criptografado usando uma chave KMS gerenciada pelo cliente, a função do IAM da frota também deverá ter permissões para descriptografar usando a chave KMS.

É altamente recomendável seguir o princípio do menor privilégio para esses segredos. Isso significa que o acesso para buscar o valor secreto do `jobRunAsUser` → `windows` → de uma fila `passwordArn` deve ser:

- concedido a uma função de frota quando uma associação fila-frota é criada entre a frota e a fila

- revogado de uma função de frota quando uma associação fila-frota é excluída entre a frota e a fila

Além disso, o AWS segredo do Secrets Manager contendo a `jobRunAsUser` senha deve ser excluído quando não estiver mais sendo usado.

Conceder acesso a uma senha secreta

As frotas do Deadline Cloud exigem acesso à `jobRunAsUser` senha armazenada no segredo da senha da fila quando a fila e a frota estão associadas. Recomendamos usar a política de recursos do AWS Secrets Manager para conceder acesso às funções da frota. Se você seguir rigorosamente essa diretriz, será mais fácil determinar quais funções da frota têm acesso ao segredo.

Para conceder acesso ao segredo

1. Abra o console do AWS Secret Manager para ver o segredo.
2. Na seção Permissões de recursos, adicione uma declaração de política no 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"
    }
  ]
}
```

Revogar o acesso a uma senha secreta

Quando uma frota não precisar mais acessar uma fila, remova o acesso à senha secreta da fila`jobRunAsUser`. Recomendamos usar a política de recursos do AWS Secrets Manager para conceder acesso às funções da frota. Se você seguir rigorosamente essa diretriz, será mais fácil determinar quais funções da frota têm acesso ao segredo.

Para revogar o acesso ao segredo

1. Abra o console do AWS Secret Manager para ver o segredo.
2. Na seção Permissões de recursos, remova a declaração de política do formulário:

   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"
    }
  ]
}
```

Instale e configure o software necessário para trabalhos

Depois de configurar o agente de trabalho do Deadline Cloud, você pode preparar o host do trabalhador com qualquer software necessário para executar trabalhos.

Quando você envia um trabalho para uma fila com um associado`jobRunAsUser`, o trabalho é executado como esse usuário. Quando um trabalho é enviado com comandos que não são um caminho absoluto, esse comando deve ser encontrado no PATH do usuário.

No Linux, você pode especificar o PATH para um usuário em uma das seguintes opções:

- deles ~/.bashrc ou ~/.bash_profile
- arquivos de configuração do sistema, como /etc/profile.d/* e /etc/profile
- scripts de inicialização do shell:/etc/bashrc.

No Windows, você pode especificar o PATH para um usuário em uma das seguintes opções:

- suas variáveis de ambiente específicas do usuário
- as variáveis de ambiente de todo o sistema

Instale adaptadores de ferramentas de criação de conteúdo digital

O Deadline Cloud fornece OpenJobDescription adaptadores para o uso de aplicativos populares de criação de conteúdo digital (DCC). Para usar esses adaptadores em uma frota gerenciada pelo cliente, você deve instalar o software DCC e os adaptadores de aplicativos. Em seguida, certifique-se de que os programas executáveis do software estejam disponíveis no caminho de busca do sistema (por exemplo, na variável de PATH ambiente).

Para instalar adaptadores DCC em uma frota gerenciada pelo cliente

1. Abra o terminal a.
    a. No Linux, abra um terminal como root usuário (ou usesudo/su)
    b. AtivadoWindows, abra um prompt de comando ou PowerShell terminal do administrador.
2. Instale os pacotes do adaptador Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

Configurando credenciais AWS

A fase inicial do ciclo de vida do trabalhador é a inicialização. Nessa fase, o software do agente de trabalho cria um trabalhador em sua frota e obtém AWS credenciais da função de sua frota para operações futuras.

AWS credentials for Amazon EC2

Para criar uma função do IAM para o Amazon EC2 com permissões de host do Deadline Cloud Worker

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Funções no painel de navegação e escolha Criar função.
3. Selecione o serviço da AWS .
4. Selecione EC2 como serviço ou caso de uso e, em seguida, selecione Avançar.
5. Para conceder as permissões necessárias, anexe a política `AWSDeadlineCloud-WorkerHost` AWS gerenciada.

On-premises AWS credentials

Seus funcionários locais usam credenciais para acessar o Deadline Cloud. Para obter o acesso mais seguro, recomendamos usar o IAM Roles Anywhere para autenticar seus trabalhadores. Para obter mais informações, consulte IAM Roles Anywhere.

Para testar, você pode usar as chaves de acesso do usuário do IAM como AWS credenciais. Recomendamos que você defina uma expiração para o usuário do IAM incluindo uma política embutida restritiva.

> **Important**
>
> Tenha atenção aos seguintes avisos:
>
> - NÃO use as credenciais raiz da sua conta para acessar AWS recursos. Estas credenciais fornecem acesso ilimitado à conta e são difíceis de revogar.
> - NÃO coloque chaves de acesso literais ou informações de credenciais em arquivos de aplicações. Se colocar, criará um risco de exposição acidental das credenciais se, por exemplo, fizer upload do projeto em um repositório público.
> - NÃO inclua arquivos que contenham credenciais em sua área de projeto.
> - Proteja suas chaves de acesso. Não as forneça a terceiros não autorizados, mesmo que seja para ajudar a localizar os identificadores da sua conta. Ao fazer isso, você pode dar a alguém acesso permanente à conta.

- Lembre-se de que todas as credenciais armazenadas no arquivo de AWS credenciais compartilhado são armazenadas em texto sem formatação.

Para obter mais detalhes, consulte Práticas recomendadas para gerenciar chaves de AWS acesso na Referência AWS geral.

Criar um usuário do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Usuários e selecione Criar usuário.
3. Dê um nome ao usuário. Desmarque a caixa de seleção Fornecer acesso ao usuário ao Console de gerenciamento da AWS e escolha Avançar.
4. Selecione Anexar políticas diretamente.
5. Na lista de políticas de permissão, escolha a AWSDeadlineCloud-WorkerHostpolítica e, em seguida, escolha Avançar.
6. Revise os detalhes do usuário e escolha Criar usuário.

Restringir o acesso do usuário a uma janela de tempo limitado

Todas as chaves de acesso do usuário do IAM que você criar serão credenciais de longo prazo. Para garantir que essas credenciais expirem caso sejam usadas incorretamente, você pode estabelecer um limite temporal para essas credenciais criando uma política em linha que especifica uma data após a qual as chaves não serão mais válidas.

1. Abra o usuário do IAM que você acabou de criar. Na guia Permissões, escolha Adicionar permissões e, em seguida, escolha Criar política embutida.
2. No editor JSON, especifique as permissões a seguir. Para usar essa política, substitua o valor do `aws:CurrentTime` carimbo de data/hora no exemplo de política por sua própria hora e data.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Criar uma chave de acesso

1. Na página de detalhes do usuário, selecione a guia Credenciais de segurança. Na seção Chaves de acesso, escolha Criar chave de acesso.
2. Indique que você deseja usar a chave para Outro, escolha Avançar e, em seguida, escolha Criar chave de acesso.
3. Na página Recuperar chaves de acesso, escolha Mostrar para revelar o valor da chave de acesso secreta do usuário. Você pode copiar as credenciais ou baixar um arquivo .csv.

Armazene as chaves de acesso do usuário

- Armazene as chaves de acesso do usuário no arquivo de AWS credenciais do usuário agente no sistema host do trabalhador:
  - LinuxAtivado, o arquivo está localizado em `~/.aws/credentials`
  - WindowsAtivado, o arquivo está localizado em `%USERPROFILE\.aws\credentials`

  Substitua as seguintes teclas:

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> Quando você não precisar mais desse usuário do IAM, recomendamos que você o remova para se alinhar às melhores práticas AWS de segurança. Recomendamos que você exija que seus usuários humanos usem credenciais temporárias Centro de Identidade do AWS IAMdurante o acesso AWS.

Fluxo de dados do host de trabalhadores para frotas gerenciadas pelo cliente

Este tópico descreve as conexões de rede que os hosts de trabalho do AWS Deadline Cloud (Deadline Cloud) fazem durante a operação, incluindo os endpoints contatados, os protocolos usados e os dados transmitidos. Essas informações se aplicam aos trabalhadores da frota gerenciada pelo cliente (CMF), incluindo instâncias do Amazon Elastic Compute Cloud (Amazon EC2) e trabalhadores locais. Use essas informações para configurar regras de firewall, criar endpoints de VPC, realizar auditorias de segurança ou planejar políticas de rede para seus hosts de trabalho. Para obter informações sobre redes de frotas gerenciadas por serviços, consulte. Inter-network privacidade no trânsito

Toda a comunicação com o trabalhador é somente externa. Os hosts de trabalho iniciam todas as conexões — você não precisa permitir nenhuma conexão de entrada. Todas as conexões usam HTTPS (TLS 1.2 ou posterior) pela porta 443.

Esse tópico inclui as seguintes seções:

- the section called “Endpoints e protocolos”
- the section called “Operações de API usadas pelos trabalhadores”
- the section called “Outros dados transmitidos”
- the section called “Opções de conectividade privada”

Endpoints e protocolos

A tabela a seguir lista os endpoints AWS de serviço aos quais os hosts de trabalho se conectam durante a operação. Para obter a lista completa de endpoints regionais para cada serviço, consulte os endpoints e cotas do serviço na AWS Referência geral.

Referência de endpoint do host do trabalhador

AWS serviço	Endpoint	Porta/Protocolo	Finalidade	Obrigatório
Deadline Cloud (agendamento)	`scheduling.deadline. [Region].amazonaws.com`	443/ HTTPS	Registro de trabalhadores, pesquisa de tarefas, atualizações de status, troca de credenciais, recuperação de entidades de trabalho. Consulte the section called “Operações de API usadas pelos trabalhadores”.	Sempre
Amazon CloudWatch Logs (CloudWatch registros)	`logs.[Region].amazonas.com`	443/ HTTPS	Agente de trabalho e entrega do registro da sessão.	Sempre
Amazon Simple Storage Service (Amazon S3)	`s3.[Region].amazonaws.com`	443/ HTTPS	Upload e download de anexos de trabalho.	Se estiver usando anexos de trabalho

Se seus trabalhos usarem outros AWS serviços, talvez você também precise permitir conexões de saída para esses endpoints de serviço.

Operações de API usadas pelos trabalhadores

Todas as operações de API a seguir usam o `scheduling.deadline.[Region].amazonaws.com` endpoint. Para ver os esquemas completos de solicitação e resposta de cada operação, consulte a Referência da Deadline Cloud API.

Fase de bootstrap

Quando um host de trabalhadores é iniciado, o agente do trabalhador se registra na frota. As credenciais de bootstrap exigem as permissões na política `AWSDeadlineCloud-WorkerHost` AWS gerenciada ou permissões personalizadas equivalentes. A fase de bootstrap usa as seguintes operações de API:

- `CreateWorker`— Registra o trabalhador na frota. Envia o nome do host e os endereços IP. Recebe uma identificação de trabalhador.
- `AssumeFleetRoleForWorker`— Obtém as credenciais da função da frota. Recebe AWS credenciais temporárias que o agente de trabalho usa para operações subsequentes.

Fase operacional

Após o bootstrap, o agente de trabalho pesquisa as sessões de trabalho e processos. A função de frota exige as permissões na política `AWSDeadlineCloud-FleetWorker` AWS gerenciada, ou permissões personalizadas equivalentes, e usa as seguintes operações de API:

- `UpdateWorker`— Atualiza o status do trabalhador, por exemplo, `STOPPED` durante o desligamento.
- `UpdateWorkerSchedule`— Pesquisas para tarefas de trabalho. Envia atualizações do status das ações da sessão, incluindo status de conclusão, porcentagem de progresso, mensagem de progresso e hashes do manifesto de saída. Recebe sessões atribuídas (ID do trabalho, ID da fila, ações da sessão, configuração do log), solicitações de cancelamento, status do trabalhador desejado e o intervalo de atualização.
- `BatchGetJobEntity`— Busca detalhes do trabalho atribuído. Envia identificadores de entidades de trabalho. Recebe detalhes do trabalho, detalhes do ambiente e detalhes do anexo do trabalho.
- `AssumeFleetRoleForWorker`— Atualiza periodicamente as credenciais da função da frota.
- `AssumeQueueRoleForWorker`— Obtém credenciais de função de fila com escopo definido para uma fila específica. O trabalhador usa essas credenciais para acessar os anexos do trabalho no Amazon S3.

Outros dados transmitidos

Além das operações da API de agendamento do Deadline Cloud, os hosts de trabalho transmitem os seguintes dados para outros AWS serviços:

Dados de registro

O agente de trabalho envia registros do agente de trabalho e registros de sessão (stdout e stderr dos processos de trabalho) para o CloudWatch Logs usando a operação de API. `PutLogEvents`

Anexos de trabalho

Os trabalhadores transferem arquivos de entrada e saída por meio do Amazon S3 usando operações `GetObject` de `PutObject` API. O trabalhador usa credenciais de função de fila obtidas `AssumeQueueRoleForWorker` por meio desse acesso.

Telemetria (opcional)

O agente do trabalhador envia métricas operacionais, como relatórios de falhas. Você pode optar por não receber a coleta de telemetria. Para obter mais informações, consulte Rejeitar.

Opções de conectividade privada

Você pode usar AWS PrivateLink para manter o tráfego entre os hosts de trabalho do CMF e o Deadline Cloud em sua VPC, sem atravessar a Internet pública. Para trabalhadores locais, você pode combinar AWS PrivateLink com AWS Direct Connect (Direct Connect) ou uma conexão VPN. Para obter mais informações, consulte Acesso AWS Deadline Cloud usando um endpoint de interface (AWS PrivateLink).

Teste a configuração do seu host de trabalho

Depois de instalar o agente de trabalho, instalar o software necessário para processar seus trabalhos e configurar AWS as credenciais do agente de trabalho, você deve testar se a instalação pode processar seus trabalhos antes de criar um AMI para sua frota. Você deve testar o seguinte:

- O agente de trabalho do Deadline Cloud está configurado corretamente para ser executado como um serviço do sistema.
- Que o trabalhador pesquise a fila associada para trabalhar.
- Que o trabalhador processe com sucesso os trabalhos enviados para a fila associada à frota.

Depois de testar a configuração e processar com êxito os trabalhos representativos, você pode usar o trabalhador configurado para criar um AMI para EC2 funcionários da Amazon ou como modelo para seus funcionários locais.

> **Note**
>
> Se você estiver testando a configuração do host do trabalhador de uma frota de escalonamento automático, talvez tenha dificuldade em testar seu trabalhador nas seguintes situações:
>
> - Se não houver trabalho na fila, o Deadline Cloud interrompe o agente do trabalhador logo após o início do trabalhador.
> - Se o agente de trabalho estiver configurado para desligar o host ao parar, o agente desligará a máquina se não houver trabalho na fila.
>
> Para evitar esses problemas, use uma frota temporária que não seja escalonada automaticamente para configurar e testar seus funcionários. Depois de testar o trabalhador anfitrião, certifique-se de definir a ID correta da frota antes de preparar um AMI.

Para testar sua configuração de host de trabalho

1. Execute o agente de trabalho iniciando o serviço do sistema operacional.

   Linux

   Em um shell raiz, execute o seguinte comando:

   ```
   systemctl start deadline-worker
   ```

   Windows

   A partir de um prompt de comando do administrador ou PowerShell terminal, digite o seguinte comando:

   ```
   sc.exe start DeadlineWorker
   ```

2. Monitore o trabalhador para garantir que ele comece e faça pesquisas para trabalhar.

   Linux

   Em um shell raiz, execute o seguinte comando:

```
systemctl status deadline-worker
```

O comando deve retornar uma resposta como:

```
Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
```

Se a resposta não for assim, inspecione o arquivo de log usando o seguinte comando:

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

A partir de um prompt de comando do administrador ou PowerShell terminal, digite o seguinte comando:

```
sc.exe query DeadlineWorker
```

O comando deve retornar uma resposta como:

```
STATE   : 4 RUNNING
```

Se a resposta não contiver`RUNNING`, inspecione o arquivo de registro do trabalhador. Aberto e administrador PowerShell solicite e execute o seguinte comando:

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. Envie trabalhos para a fila associada à sua frota. Os trabalhos devem ser representativos dos trabalhos que a frota processa.
4. Monitore o progresso do trabalho usando o monitor ou a CLI do Deadline Cloud. Se um trabalho falhar, verifique os registros da sessão e do trabalhador.
5. Atualize a configuração do host do trabalhador conforme necessário até que os trabalhos sejam concluídos com êxito.
6. Quando os trabalhos de teste forem bem-sucedidos, você poderá parar o trabalhador:

Linux

Em um shell raiz, execute o seguinte comando:

```
systemctl stop deadline-worker
```

Windows

A partir de um prompt de comando do administrador ou PowerShell terminal, digite o seguinte comando:

```
sc.exe stop DeadlineWorker
```

Crie um Amazon Machine Image

Para criar um Amazon Machine Image (AMI) para usar em uma frota gerenciada pelo cliente (CMF EC2) da Amazon Elastic Compute Cloud (Amazon), conclua as tarefas nesta seção. Você deve criar uma EC2 instância da Amazon antes de continuar. Para obter mais informações, consulte Inicie sua instância no Guia EC2 do usuário da Amazon para instâncias Linux.

> **Important**
>
> Criando um AMI cria um instantâneo dos volumes anexados à EC2 instância da Amazon. Qualquer software instalado na instância persiste, portanto, as instâncias, que são reutilizadas quando você executa instâncias a partir do AMI. Recomendamos adotar uma estratégia de correção e atualizar regularmente qualquer nova AMI com software atualizado antes de aplicar em sua frota.

Prepare a EC2 instância da Amazon

Antes de criar um AMI, você deve excluir o estado do trabalhador. O estado do trabalhador persiste entre os lançamentos do agente de trabalho. Se esse estado persistir no AMI, então todas as instâncias lançadas a partir dele compartilharão o mesmo estado.

Também recomendamos que você exclua todos os arquivos de log existentes. Os arquivos de log podem permanecer em uma EC2 instância da Amazon quando você prepara a AMI. A

exclusão desses arquivos minimiza a confusão ao diagnosticar possíveis problemas nas frotas de trabalhadores que usam a AMI.

Você também deve habilitar o serviço de sistema do agente de trabalho para que o agente do Deadline Cloud Worker seja iniciado quando a Amazon EC2 for iniciada.

Por fim, recomendamos que você ative o desligamento automático do agente de trabalho. Isso permite que a frota de trabalhadores aumente quando necessário e seja encerrada quando o trabalho de renderização for concluído. Esse escalonamento automático ajuda a garantir que você use os recursos somente conforme necessário.

Para preparar a EC2 instância da Amazon

1. Abra o EC2 console da Amazon.
2. Inicie uma EC2 instância da Amazon. Para obter mais informações, consulte Executar sua instância.
3. Configure o host para se conectar ao seu provedor de identidade (IdP) e, em seguida, monte qualquer sistema de arquivos compartilhado necessário.
4. Siga os tutoriais paraInstale o agente Deadline Cloud Worker, em seguidaConfigurar agente de trabalho, e. Crie usuários e grupos de trabalho
5. Se você estiver preparando um AMI com base no Amazon Linux 2023 para executar software compatível com a plataforma de referência VFX, você precisa atualizar vários requisitos. Para obter mais informações, consulte Compatibilidade da plataforma de referência VFX no Guia do usuário do AWS Deadline Cloud.
6. Abra um terminal.
    a. No Linux, abra um terminal como `root` usuário (ou use`sudo`/`su`)
    b. Ativado Windows, abra um prompt de comando ou PowerShell terminal do administrador.
7. Certifique-se de que o serviço de trabalho não esteja em execução e configurado para iniciar na inicialização:
    a. No Linux, execute

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

    b. Ativado Windows, corra

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. Exclua o estado do trabalhador.

   a. No Linux, execute

   ```
   rm -rf /var/lib/deadline/*
   ```

   b. Ativado Windows, corra

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
   ```

9. Exclua os arquivos de log.

   a. No Linux, execute

   ```
   rm -rf /var/log/amazon/deadline/*
   ```

   b. Ativado Windows, corra

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
   ```

10. Ativado Windows, é recomendável executar o aplicativo Amazon EC2 Launch Settings encontrado no menu Iniciar para concluir a preparação final do host e o desligamento da instância.

> **Note**
>
> Você DEVE escolher Desligar sem Sysprep e nunca escolher Desligar com Sysprep. Desligar com o Sysprep fará com que todos os usuários locais se tornem inutilizáveis. Para obter mais informações, consulte a seção Antes de começar do tópico Criar uma AMI personalizada do Guia do usuário para instâncias do Windows.

Construa o AMI

Para construir o AMI

1. Abra o EC2 console da Amazon.
2. Selecione Instâncias no painel de navegação e, em seguida, selecione sua instância.
3. Escolha Estado da instância e, em seguida, Parar instância.
4. Depois que a instância for interrompida, escolha Ações.
5. Escolha Imagem e modelos e, em seguida, Criar imagem.
6. Insira o nome da imagem.
7. (Opcional) Insira uma descrição para sua imagem.
8. Escolha Criar imagem.

Crie uma infraestrutura de frota com um grupo Amazon EC2 Auto Scaling

Esta seção explica como criar uma frota do Amazon EC2 Auto Scaling.

Use o modelo CloudFormation YAML abaixo para criar um grupo do Amazon EC2 Auto Scaling (Auto Scaling), uma Amazon Virtual Private Cloud (Amazon VPC) com duas sub-redes, um perfil de instância e uma função de acesso à instância. Eles são necessários para iniciar a instância usando o Auto Scaling nas sub-redes.

Você deve revisar e atualizar a lista de tipos de instância para atender às suas necessidades de renderização.

Para obter uma explicação completa dos recursos e parâmetros usados no modelo CloudFormation YAML, consulte a referência do tipo de recurso do Deadline Cloud no Guia do AWS CloudFormation usuário.

Para criar uma frota do Amazon EC2 Auto Scaling

1. Use o exemplo a seguir para criar um CloudFormation modelo que define os `AMIId` parâmetros `FarmIDFleetID`, e. Salve o modelo em um `.YAML` arquivo no seu computador 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. Abra o CloudFormation console em https://console.aws.amazon.com/cloudformation.

   Use o CloudFormation console para criar uma pilha usando as instruções para carregar o arquivo de modelo que você criou. Para obter mais informações, consulte Criação de uma pilha no CloudFormation console no Guia do AWS CloudFormation usuário.

> **Note**
>
> - As credenciais da função do IAM que estão anexadas à instância do Amazon EC2 do seu trabalhador estão disponíveis para todos os processos em execução nesse trabalhador, o que inclui trabalhos. O trabalhador deve ter o mínimo de privilégios para operar: `deadline:CreateWorker` e `deadline:AssumeFleetRoleForWorker.`
> - O agente de trabalho obtém as credenciais para a função de fila e as configura para uso na execução de trabalhos. A função do perfil da instância do Amazon EC2 não deve incluir as permissões necessárias para seus trabalhos.

Escale automaticamente sua frota do Amazon EC2 com o recurso de recomendação de escala Deadline Cloud

O Deadline Cloud utiliza um grupo do Amazon EC2 Auto Scaling (Auto Scaling) para escalar automaticamente a frota gerenciada pelo cliente (CMF) do Amazon EC2. Você precisa configurar o modo de frota e implantar a infraestrutura necessária em sua conta para que sua frota seja dimensionada automaticamente. A infraestrutura que você implantou funcionará para todas as frotas, então você só precisa configurá-la uma vez.

O fluxo de trabalho básico é: você configura seu modo de frota para escalar automaticamente e, em seguida, o Deadline Cloud enviará um EventBridge evento para essa frota sempre que o tamanho recomendado da frota mudar (um evento contém o ID da frota, o tamanho recomendado da frota e outros metadados). Você terá uma EventBridge regra para filtrar os eventos relevantes e ter um Lambda para consumi-los. O Lambda se integrará ao Amazon EC2 Auto `AutoScalingGroup` Scaling para escalar automaticamente a frota do Amazon EC2.

Defina o modo de frota como **EVENT_BASED_AUTO_SCALING**

Configure seu modo de frota para`EVENT_BASED_AUTO_SCALING`. Você pode usar o console para fazer isso ou usar o AWS CLI para chamar diretamente a `UpdateFleet` API `CreateFleet` ou. Depois que o modo é configurado, o Deadline Cloud começa a enviar EventBridge eventos sempre que o tamanho recomendado da frota mudar.

- Exemplo de `UpdateFleet` comando:

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- Exemplo de `CreateFleet` comando:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

A seguir está um exemplo de `configuration.json` uso nos comandos da CLI acima (`--configuration file://configuration.json`.

- Para ativar o Auto Scaling em sua frota, você deve definir o modo como. `EVENT_BASED_AUTO_SCALING`
- Esses `workerCapabilities` são os valores padrão atribuídos ao CMF quando você o criou. Você pode alterar esses valores se precisar aumentar os recursos disponíveis para seu CMF.

Depois de configurar o modo de frota, o Deadline Cloud começa a emitir eventos de recomendação de tamanho de frota para essa frota.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
```

```
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

Implante a pilha de Auto Scaling usando o modelo CloudFormation

Você pode configurar uma EventBridge regra para filtrar eventos, um Lambda para consumir os eventos e controlar o Auto Scaling e uma fila SQS para armazenar eventos não processados. Use o CloudFormation modelo a seguir para implantar tudo em uma pilha. Depois de implantar os recursos com sucesso, você pode enviar um trabalho e a frota aumentará automaticamente.

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
              "source": "aws.deadline",
              "account": "111122223333",
              "time": "2017-12-22T18:43:48Z",
              "region": "us-west-1",
              "resources": [],
              "detail": {
                  "farmId": "farm-12345678900000000000000000000000",
```

```
                "fleetId": "fleet-12345678900000000000000000000000",
                "oldFleetSize": 1,
                "newFleetSize": 5,
            }
        }
        """

        import json
        import boto3
        import logging

        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        auto_scaling_client = boto3.client("autoscaling")

        def lambda_handler(event, context):
            logger.info(event)
            event_detail = event["detail"]
            fleet_id = event_detail["fleetId"]
            desired_capacity = event_detail["newFleetSize"]

            asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
            auto_scaling_client.set_desired_capacity(
                AutoScalingGroupName=asg_name,
                DesiredCapacity=desired_capacity,
                HonorCooldown=False,
            )

            return {
                'statusCode': 200,
                'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
            }
     Handler: index.lambda_handler
     Role: !GetAtt
       - AutoScalingLambdaServiceRole
       - Arn
     Runtime: python3.11
   DependsOn:
     - AutoScalingLambdaServiceRoleDefaultPolicy
     - AutoScalingLambdaServiceRole
 AutoScalingEventRule:
   Type: 'AWS::Events::Rule'
```

```
    Properties:
      EventPattern:
        source:
          - aws.deadline
        detail-type:
          - Fleet Size Recommendation Change
      State: ENABLED
      Targets:
        - Arn: !GetAtt
            - AutoScalingLambda
            - Arn
          DeadLetterConfig:
            Arn: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
          Id: Target0
          RetryPolicy:
            MaximumRetryAttempts: 15
  AutoScalingEventRuleTargetPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !GetAtt
        - AutoScalingLambda
        - Arn
      Principal: events.amazonaws.com
      SourceArn: !GetAtt
        - AutoScalingEventRule
        - Arn
  AutoScalingLambdaServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - !Join
          - ''
          - - 'arn:'
            - !Ref 'AWS::Partition'
```

```
          - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  AutoScalingLambdaServiceRoleDefaultPolicy:
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
      Roles:
        - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

Realize uma verificação de integridade da frota

Depois de criar sua frota, você deve criar uma verificação de saúde personalizada para garantir que sua frota permaneça íntegra e livre de instâncias paralisadas para ajudar a evitar custos desnecessários. Consulte Implantação de uma verificação de integridade da frota do Deadline Cloud. GitHub Uma verificação de integridade pode reduzir o risco de uma alteração acidental em seu Amazon Machine Image modelo de lançamento ou configuração de rede ser executada sem ser detectada.

Configure e use frotas gerenciadas por serviços do Deadline Cloud

Uma frota gerenciada por serviços (SMF) é uma coleção de trabalhadores gerenciada pela Deadline Cloud. Um SMF elimina a necessidade de gerenciar o dimensionamento da frota para atender às demandas de processamento ou reduzir o tamanho da frota após a conclusão da tarefa.

Quando um SMF é associado a uma fila usando o ambiente padrão de filas conda, o Deadline Cloud configura os trabalhadores da frota com o pacote de software apropriado. Para aplicativos de parceiros compatíveis, consulte Ambiente padrão de fila conda no Guia do usuário do AWS Deadline Cloud.

Na maioria dos casos, você não precisa alterar um SMF para processar suas cargas de trabalho. No entanto, algumas situações podem exigir que você faça alterações em suas frotas.

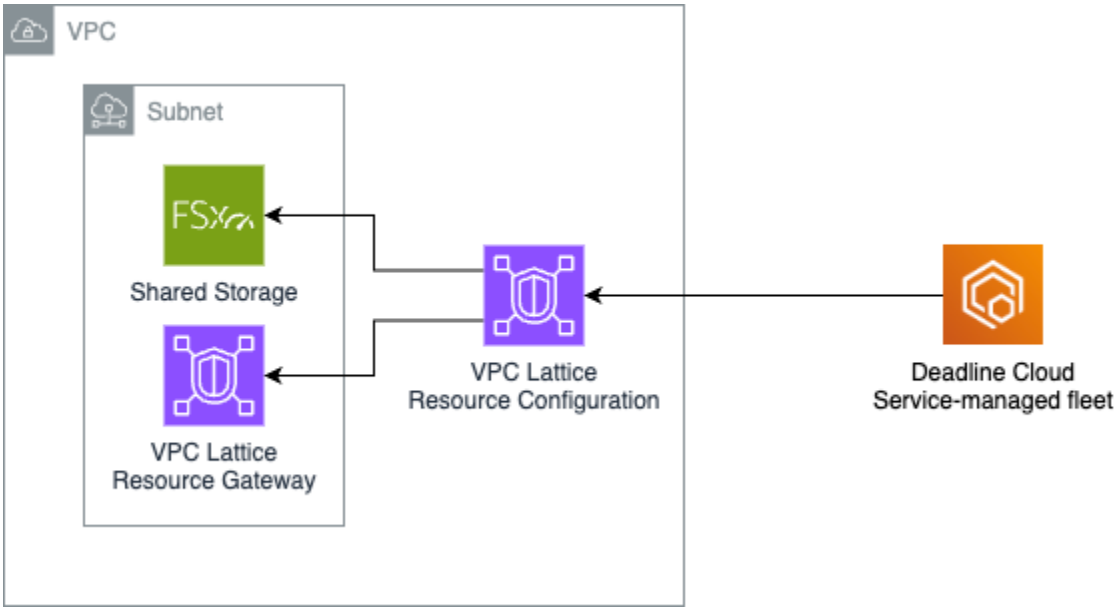
> **Note**
>
> Para instalar software personalizado em trabalhadores usando scripts de configuração do host, consulteExecute scripts de configuração do host com privilégios de administrador.

Tópicos

- Conecte recursos de VPC ao seu SMF com endpoints de recursos de VPC
- Use anexos de trabalho com frotas gerenciadas por serviços

Conecte recursos de VPC ao seu SMF com endpoints de recursos de VPC

Com os endpoints de recursos do Amazon VPC para frotas gerenciadas por serviços (SMF) do Deadline Cloud, você pode conectar seus recursos de VPC, como sistemas de arquivos de rede (NFS), servidores de licenças e bancos de dados, com seus funcionários do Deadline Cloud. Esse recurso permite que você aproveite a plataforma totalmente gerenciada do Deadline Cloud enquanto se integra à sua infraestrutura existente em uma VPC.



> **Tip**
>
> Para obter um CloudFormation modelo de referência que configura um FSx cluster da Amazon e o conecta a uma frota gerenciada por serviços, consulte smf_vpc_fsx no repositório de amostras do Deadline Cloud em. GitHub

Como funcionam os endpoints de recursos de VPC

Os endpoints de recursos de VPC usam o VPC Lattice para criar uma conexão segura entre seus funcionários do Deadline Cloud SMF e os recursos em sua VPC. A conexão é unidirecional, o que significa que os trabalhadores podem estabelecer uma conexão com os recursos em sua VPC e transferir dados para frente e para trás, mas os recursos em sua VPC não podem estabelecer uma conexão com um trabalhador.

Quando você conecta um recurso de VPC a uma frota gerenciada por serviços do Deadline Cloud, seus funcionários podem acessar recursos em sua VPC usando um nome de domínio privado. Além disso, o tráfego flui dos trabalhadores para seus recursos de VPC por meio do VPC Lattice, e os recursos em sua VPC veem o tráfego proveniente do gateway de recursos do VPC Lattice.

Para saber mais, consulte o guia do usuário do VPC Lattice.

Pré-requisitos

Antes de conectar os recursos de VPC à sua frota gerenciada por serviços do Deadline Cloud, verifique se você tem o seguinte:

- Uma AWS conta com uma VPC contendo recursos que você deseja conectar.
- Permissões do IAM para criar e gerenciar recursos do VPC Lattice.
- Uma fazenda Deadline Cloud com pelo menos uma frota gerenciada por serviços.
- Recursos de VPC que você deseja tornar acessíveis (NFSFSx, servidores de licenças etc.).

Configurar um endpoint de recursos de VPC

Para configurar um endpoint de recursos de VPC, você precisa criar recursos no VPC Lattice e AWS RAM, em seguida, conectar esses recursos à sua frota no Deadline Cloud. Para configurar um endpoint de recursos de VPC para seu SMF, conclua as etapas a seguir.

1. Para criar um gateway de recursos no VPC Lattice, consulte Criar um gateway de recursos no guia do usuário do VPC Lattice.
2. Para criar uma configuração de recursos no VPC Lattice, consulte Criar uma configuração de recursos no guia do usuário do VPC Lattice.
3. Para compartilhar o recurso com sua frota do Deadline Cloud, crie um compartilhamento de recursos em AWS RAM. Consulte Criação de um compartilhamento de recursos para obter instruções.

   Ao criar um compartilhamento de recursos, para Diretores, selecione Principal de serviço no menu suspenso e, em seguida, insira. **`fleets.deadline.amazonaws.com`**
4. Para conectar a configuração do recurso à sua frota do Deadline Cloud, conclua as etapas a seguir.

   a. Se ainda não o fez, abra o console do Deadline Cloud.

   b. No painel de navegação, escolha Fazendas e, em seguida, selecione sua fazenda.

   c. Escolha a guia Frotas e, em seguida, selecione sua frota.

   d. Escolha a guia Configurations (Configurações).

   e. Em VPC resource endpoints, escolha Editar.

   f. Selecione a configuração do recurso que você criou e escolha Salvar alterações.

Acessando seus recursos de VPC

Depois de conectar seu recurso de VPC à sua frota, os trabalhadores podem acessá-lo usando um nome de domínio privado no seguinte formato: `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Esse domínio é privado e só pode ser acessado por trabalhadores (não pela Internet ou pela sua estação de trabalho).

Para montar ou configurar o acesso ao recurso de VPC em seus trabalhadores, use um script de configuração de host. Os scripts de configuração do host são executados com privilégios de administrador quando os trabalhadores iniciam, permitindo que você monte sistemas de arquivos, defina configurações de rede ou execute outras tarefas de configuração.

Autenticação e segurança

Para recursos que exigem autenticação, armazene credenciais com segurança no AWS Secrets Manager, acesse segredos dos scripts de configuração do host ou scripts de trabalho e implemente as permissões apropriadas do sistema de arquivos para controlar o acesso. Considere as implicações de segurança ao compartilhar recursos em várias frotas. Por exemplo, se duas frotas estiverem conectadas ao mesmo armazenamento compartilhado, os trabalhos executados em uma frota poderão acessar ativos criados na outra frota.

Considerações técnicas

Ao usar endpoints de recursos de VPC, considere o seguinte:

- As conexões só podem ser iniciadas de trabalhadores para recursos de VPC, não de recursos de VPC para trabalhadores.
- Uma vez estabelecida, a conexão persiste até ser redefinida, mesmo que a configuração do recurso seja desconectada.
- A conexão VPC Lattice gerencia conexões entre zonas de disponibilidade automaticamente, sem custos adicionais. Seu gateway de recursos deve compartilhar uma zona de disponibilidade com seu recurso de VPC, por isso recomendamos configurar o gateway de recursos para abranger todas as zonas de disponibilidade.
- O tráfego que passa pelo endpoint de recursos da VPC usa Network Address Translation (NAT), que não é compatível com todos os casos de uso. Por exemplo, o Microsoft Active Directory (AD) não pode se conectar via NAT.

Para obter mais informações sobre as cotas do VPC Lattice, consulte Cotas para o VPC Lattice.

Solução de problemas

Se você encontrar problemas com endpoints de recursos de VPC, verifique o seguinte.

- Se você receber uma mensagem de erro como “mount.nfs: acesso negado pelo servidor durante a montagem”, talvez seja necessário atualizar a configuração do cliente do seu volume NFS.
- Verifique sua configuração de recursos testando em uma instância do Amazon EC2 ou AWS CloudShell em sua VPC.
- Teste sua conexão com o Deadline Cloud com tarefas simples de CLI. Para obter mais informações, consulte exemplos do Deadline Cloud em GitHub.
- Verifique as configurações no grupo de segurança do gateway de recursos se você tiver falhas de conexão.
- Ative os registros de acesso à VPC para monitorar as conexões.

Use anexos de trabalho com frotas gerenciadas por serviços

Os anexos do trabalho transferem arquivos entre sua estação de trabalho e os trabalhadores do Deadline Cloud usando o Amazon Simple Storage Service (Amazon S3). Você pode usar anexos de tarefas sozinhos ou em conjunto com o armazenamento compartilhado para anexar dados auxiliares a tarefas que não são compartilhadas com outras tarefas, como scripts de tarefas, arquivos de configuração ou ativos de projetos armazenados localmente.

Para obter informações sobre como os anexos de trabalho funcionam, consulte Anexos de trabalhos no Guia do usuário do Deadline Cloud. Para obter detalhes sobre a especificação de arquivos de entrada e saída em pacotes de tarefas, consulte. Use anexos de trabalho para compartilhar arquivos

Escolha um modo de sistema de arquivos

Ao enviar um trabalho com anexos, você pode escolher como os trabalhadores carregam arquivos do Amazon S3 definindo a propriedade: `fileSystem`

- COPIADO (padrão) — baixa todos os arquivos para o disco local antes do início das tarefas. Melhor quando cada tarefa precisa da maioria dos arquivos de entrada.

- VIRTUAL — monta um sistema de arquivos virtual que baixa arquivos sob demanda. Melhor quando as tarefas precisam apenas de um subconjunto de arquivos de entrada. Disponível somente para trabalhadores Linux SMF.

> **Important**
>
> O cache no modo VIRTUAL pode aumentar o consumo de memória e não é otimizado para todas as cargas de trabalho. Recomendamos que você teste sua carga de trabalho antes de executar trabalhos de produção.

Para obter informações detalhadas sobre como configurar o modo de sistema de arquivos, consulte Sistema de arquivos virtual no Guia do usuário do Deadline Cloud.

Otimize o desempenho da transferência

A velocidade de sincronização de arquivos do Amazon S3 com os funcionários do SMF depende da configuração de volume do Amazon Elastic Block Store (Amazon EBS) da sua frota. Por padrão, os trabalhadores do SMF usam volumes gp3 do Amazon EBS com configurações básicas de desempenho. Para cargas de trabalho com arquivos de entrada grandes ou muitos arquivos pequenos, você pode melhorar as velocidades de transferência aumentando a taxa de transferência e as configurações de IOPS do Amazon EBS. Você pode atualizar essas configurações usando o AWS Command Line Interface (AWS CLI).

Throughput (MiB/s)

A taxa na qual os dados podem ser lidos ou gravados no volume. O padrão é 125 MiB/s, maximum is 1,000 MiB/s para volumes gp3. Aumento para grandes transferências sequenciais de arquivos.

IOPS

Operações de entrada/saída por segundo. O padrão é 3.000 IOPS, o máximo é 16.000 IOPS para volumes gp3. Aumente ao transferir muitos arquivos pequenos.

> **Note**
>
> Aumentar a taxa de transferência e o IOPS do Amazon EBS aumenta o custo da frota. Para obter informações sobre preços, consulte Preços do Deadline Cloud.

Para atualizar as configurações do Amazon EBS em uma frota existente usando o AWS CLI

- Execute este 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"}
    }
  }'
```

Baixe os resultados do trabalho

Depois que seu trabalho for concluído, baixe os arquivos de saída usando a CLI do Deadline Cloud AWS ou o monitor do Deadline Cloud (monitor do Deadline Cloud):

```
deadline job download-output --job-id job-0123456789abcdef0
```

Para baixar automaticamente as saídas à medida que os trabalhos são concluídos, consulte Downloads automáticos no Guia do usuário do Deadline Cloud.

Implemente e configure software personalizado para trabalhadores

AWS O Deadline Cloud fornece vários métodos para implantar e configurar software, plug-ins e ferramentas personalizados em seus funcionários. O método escolhido depende de seus requisitos, como se você precisa de privilégios de administrador, com que frequência o software muda e se o software deve estar disponível para todos os trabalhos ou somente trabalhos específicos.

Escolha um método de implantação

Use a tabela a seguir para escolher o método de implantação certo para seu caso de uso.

Critérios	Ambiente de filas	Script de configuração do host	Pacote conda personalizado
Privilégios de administrador necessários	Não	Sim	Não
Quando é executado	Início da sessão	Startup de trabalhadores	Início da sessão
Escopo	Por fila ou trabalho	Todos os trabalhadores da frota	Por fila ou trabalho
Pode ser controlado pelo envio de trabalhos	Sim	Não	Sim
Complexidade da configuração	Baixo	Médio	Alto
Melhor para	Plugins, scripts e variáveis de ambiente simples	Drivers de sistema, Docker, suportes de armazenamento	Aplicativos complexos com dependências

Guia de decisão rápida:

- Precisa de privilégios de administrador ou root? Use um script de configuração do host.

- Plugin ou script simples sem direitos de administrador? Use um ambiente de fila.
- Aplicativo complexo com necessidades de controle de versão? Crie um pacote conda personalizado.

Configurar trabalhos usando ambientes de fila

AWS O Deadline Cloud usa ambientes de fila para configurar o software em seus trabalhadores. Um ambiente permite que você execute tarefas demoradas, como configurar e desmontar, uma vez para todas as tarefas em uma sessão. Ele define as ações a serem executadas em um trabalhador ao iniciar ou interromper uma sessão. Você pode configurar um ambiente para uma fila, trabalhos que são executados na fila e as etapas individuais para um trabalho.

Você define ambientes como ambientes de fila ou ambientes de trabalho. Crie ambientes de fila com o console Deadline Cloud ou com a CreateQueueEnvironment operação Deadline: e defina ambientes de trabalho nos modelos de trabalho dos trabalhos que você envia. Eles seguem a especificação Open Job Description (OpenJD) para ambientes. Para obter detalhes, consulte a <Environment>especificação do OpenJDhttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment em. GitHub

Além de um `name` e`description`, cada ambiente contém dois campos que definem o ambiente no host. Eles são:

- `script`— A ação tomada quando esse ambiente é executado em um trabalhador.
- `variables`— Um conjunto de name/value pares de variáveis de ambiente que são definidos ao entrar no ambiente.

Você deve definir pelo menos um dos `script` ou`variables`.

Você pode definir mais de um ambiente em seu modelo de trabalho. Cada ambiente é aplicado na ordem em que estão listados no modelo. Você pode usar isso para ajudar a gerenciar a complexidade de seus ambientes.

O ambiente de fila padrão do Deadline Cloud usa o gerenciador de pacotes conda para carregar software no ambiente, mas você pode usar outros gerenciadores de pacotes. O ambiente padrão define dois parâmetros para especificar o software que deve ser carregado. Essas variáveis são definidas pelos remetentes fornecidas pelo Deadline Cloud, embora você possa defini-las em seus próprios scripts e aplicativos que usam o ambiente padrão. Eles são:

- `CondaPackages`— Uma lista separada por espaço do pacote conda corresponde às especificações a serem instaladas para o trabalho. Por exemplo, o remetente do Blender adicionaria quadros `blender=3.6` para renderizar no Blender 3.6.
- `CondaChannels`— Uma lista separada por espaços de canais conda para instalar pacotes. Para frotas gerenciadas por serviços, os pacotes são instalados a partir do canal. `deadline-cloud` Você pode adicionar outros canais.

Controle o ambiente de trabalho com ambientes de fila do OpenJD

Você pode definir ambientes personalizados para seus trabalhos de renderização usando ambientes de fila. Um ambiente de fila é um modelo que controla as variáveis de ambiente, mapeamentos de arquivos e outras configurações para trabalhos executados em uma fila específica. Ele permite que você adapte o ambiente de execução dos trabalhos enviados a uma fila de acordo com os requisitos de suas cargas de trabalho. AWS O Deadline Cloud fornece três níveis agrupados nos quais você pode aplicar ambientes Open Job Description (OpenJD): fila, tarefa e etapa. Ao definir ambientes de filas, você pode garantir um desempenho consistente e otimizado para diferentes tipos de trabalhos, agilizar a alocação de recursos e simplificar o gerenciamento de filas.

O ambiente de fila é um modelo que você anexa a uma fila em sua AWS conta a partir do console de AWS gerenciamento ou usando o. AWS CLI Você pode criar um ambiente para uma fila ou criar vários ambientes de fila aplicados para criar o ambiente de execução. Essa abordagem permite criar e testar um ambiente em etapas para ajudar a garantir que ele funcione corretamente para seus trabalhos.

Os ambientes de trabalho e etapas são definidos no modelo de trabalho que você usa para criar um trabalho na sua fila. A sintaxe do OpenJD é a mesma nessas diferentes formas de ambientes. Nesta seção, mostraremos eles dentro dos modelos de trabalho.

Tópicos

- Definir variáveis de ambiente em um ambiente de fila
- Definir o caminho em um ambiente de fila
- Execute um processo daemon em segundo plano a partir do ambiente de fila

Definir variáveis de ambiente em um ambiente de fila

Muitos aplicativos e estruturas usam variáveis de ambiente para controlar as configurações de recursos, os níveis de registro e a configuração de exibição. Você pode usar ambientes Open Job

Description (OpenJD) para definir variáveis de ambiente que cada comando de tarefa dentro de seu escopo herda.

Escopo da variável de ambiente

AWS O Deadline Cloud aplica variáveis de ambiente de ambientes de fila que você anexa a uma fila. Em um modelo de trabalho, você também pode definir variáveis de ambiente nos níveis de tarefa e etapa usando ambientes OpenJD. Variáveis definidas em um escopo mais restrito substituem variáveis com o mesmo nome de um escopo mais amplo.

- Ambiente de fila — Um modelo que você anexa a uma fila no Deadline Cloud. As variáveis se aplicam a todos os trabalhos enviados à fila. Você pode definir variáveis com um `variables` mapa para valores fixos ou usar scripts para valores dinâmicos.
- Ambiente de trabalho — Definido abaixo `jobEnvironments` em um modelo de trabalho. As variáveis se aplicam a todas as etapas e tarefas do trabalho. Uma variável no nível do trabalho substitui uma variável no nível da fila com o mesmo nome.
- Ambiente de etapas — definido abaixo `stepEnvironments` em um modelo de trabalho. As variáveis se aplicam somente às tarefas dessa etapa. Uma variável de nível de etapa substitui uma variável de nível de trabalho ou de fila com o mesmo nome.

Configurando variáveis em um ambiente de fila

Você pode definir variáveis de ambiente em um ambiente de fila usando um `variables` mapa para valores fixos ou usando um `script` com uma `onEnter` ação para valores dinâmicos.

O modelo de ambiente de fila a seguir usa um `variables` mapa para definir a `QT_QPA_PLATFORM` variável`offscreen`, o que permite que aplicativos que usam o Qt Framework sejam executados em hosts de trabalho sem uma exibição interativa.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Para valores dinâmicos, como modificar `PATH` ou ativar ambientes virtuais, use um script que imprima linhas no formato `openjd_env: VAR=value` em stdout. O `openjd_env:` prefixo é obrigatório. Use`echo`,`export`, ou outros mecanismos de shell sem o prefixo não propaga variáveis para trabalhos e tarefas.

O modelo de ambiente de fila a seguir define a `QT_QPA_PLATFORM` variável usando um 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 anexar um ambiente de fila à sua fila, use o console do Deadline Cloud ou o. AWS CLI Para obter mais informações, consulte Criar um ambiente de fila no Guia do usuário do AWS Deadline Cloud. O AWS CLI comando a seguir cria um ambiente de fila a partir de um arquivo de modelo.

```
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 exemplos mais complexos, como criar e ativar ambientes virtuais conda, consulte os exemplos do ambiente de filas do Deadline Cloud em. GitHub

Definindo variáveis em um modelo de trabalho

Em um modelo de trabalho, adicione um `variables` mapa a uma `stepEnvironments` entrada `jobEnvironments` or. Cada entrada é um par de valores-chave em que a chave é o nome da variável e o valor é o valor da variável.

O modelo de trabalho a seguir define a variável de `QT_QPA_PLATFORM` ambiente como`offscreen`, o que permite que aplicativos que usam o Qt Framework sejam executados em hosts de trabalho sem uma exibição interativa.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

Você pode definir várias variáveis em uma única definição de ambiente.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
    QT_QPA_PLATFORM: offscreen
```

Você pode referenciar os parâmetros do trabalho em valores variáveis usando a `{{Param.ParameterName}}` sintaxe.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Para substituir uma variável de nível de trabalho para uma etapa específica, defina uma `stepEnvironments` entrada com o mesmo nome de variável. O exemplo a seguir define `JOB_PROJECT_ID` no nível do trabalho com o valor e`project-12`, em seguida, substitui o valor no nível da etapa com. `step-project-12` As tarefas na etapa usam o valor no nível da etapa.

```
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
```

Experimente: Executando a amostra da variável de ambiente

O repositório de amostras do Deadline Cloud inclui um pacote de tarefas que demonstra a configuração e a visualização de variáveis de ambiente. O modelo de trabalho de amostra define variáveis nos níveis do trabalho e da etapa e, em seguida, executa uma tarefa que imprime o resultado mesclado. Use o procedimento a seguir para executar a amostra e inspecionar os resultados.

Pré-requisitos

1. Se você não tiver um farm do Deadline Cloud com uma fila e uma frota Linux associada, siga a experiência de integração guiada no console do Deadline Cloud para criar um com as configurações padrão.
2. Se você não tiver a CLI do Deadline Cloud e o monitor do AWS Deadline Cloud em sua estação de trabalho, siga as etapas em Configurar os remetentes do Deadline Cloud.
3. Use `git` para clonar o GitHubrepositório de amostras do Deadline Cloud.

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

Executar o exemplo

1. Use a CLI do Deadline Cloud para enviar a `job_env_vars` amostra.

```
deadline bundle submit job_env_vars
```

2. No monitor do Deadline Cloud, selecione o novo trabalho para monitorar seu progresso. Depois que a Linux frota associada à fila tiver um trabalhador disponível, o trabalho será concluído em alguns segundos. Selecione a tarefa e escolha Exibir registros no menu superior direito do painel de tarefas.

Comparando as ações da sessão com suas definições

A visualização do registro mostra três ações da sessão. Abra o arquivo job_env_vars/template.yaml em um editor de texto para comparar cada ação com sua definição no modelo de trabalho.

1. Selecione a ação Iniciar JobEnv sessão. A saída do log mostra as variáveis de ambiente no nível do trabalho que estão sendo definidas.

```
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
```

   As linhas a seguir do modelo de trabalho definem esse ambiente.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. Selecione a ação Iniciar StepEnv sessão. A saída do log mostra as variáveis em nível de etapa, incluindo as substituídas. `JOB_PROJECT_ID`

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   As linhas a seguir do modelo de trabalho definem esse ambiente.

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Selecione a ação Tarefa executar sessão. A saída do log mostra as variáveis de ambiente mescladas disponíveis para a tarefa. Observe que `JOB_PROJECT_ID` usa o valor em nível de etapa. `step-project-12`

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
```

```
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

Definir o caminho em um ambiente de fila

Use ambientes OpenJD para fornecer novos comandos em um ambiente. Primeiro, você cria um diretório contendo arquivos de script e, em seguida, adiciona esse diretório às variáveis de PATH ambiente para que os executáveis em seu script possam executá-los sem precisar especificar o caminho do diretório a cada vez. A lista de variáveis em uma definição de ambiente não fornece uma maneira de modificar a variável, então você faz isso executando um script em vez disso. Depois que o script configura e modifica oPATH, ele exporta a variável para o tempo de execução do OpenJD com o comando. `echo "openjd_env: PATH=$PATH"`

Pré-requisitos

Execute as etapas a seguir para executar o pacote de tarefas de amostra com variáveis de ambiente do repositório github de amostras do Deadline Cloud.

1. Se você não tiver um farm do Deadline Cloud com uma fila e uma frota Linux associada, siga a experiência de integração guiada no console do Deadline Cloud para criar um com as configurações padrão.
2. Se você não tiver a CLI do Deadline Cloud e o monitor do Deadline Cloud em sua estação de trabalho, siga as etapas em Configurar os remetentes do Deadline Cloud no guia do usuário.
3. Use `git` para clonar o GitHubrepositório de amostras do Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Execute a amostra do caminho

1. Use a CLI do Deadline Cloud para enviar a `job_env_with_new_command` amostra.

```
 $ deadline bundle submit job_env_with_new_command
 Submitting to Queue: MySampleQueue
```

```
...
```

2. No monitor do Deadline Cloud, você verá o novo trabalho e poderá monitorar seu progresso. Quando a Linux frota associada à fila tiver um trabalhador disponível para executar a tarefa do trabalho, o trabalho será concluído em alguns segundos. Selecione a tarefa e, em seguida, escolha a opção Exibir registros no menu superior direito do painel de tarefas.

   À direita estão duas ações de sessão, Iniciar RandomSleepCommand e Executar tarefa. O visualizador de registros no centro da janela corresponde à ação de sessão selecionada à direita.

Compare as ações da sessão com suas definições

Nesta seção, você usa o monitor Deadline Cloud para comparar as ações da sessão com o local em que elas estão definidas no modelo de trabalho. Ela continua a partir da seção anterior.

Abra o arquivo job_env_with_new_command/template.yaml em um editor de texto. Compare as ações da sessão com o local em que elas estão definidas no modelo de trabalho.

1. Selecione a ação Iniciar RandomSleepCommand sessão no monitor do Deadline Cloud. Você verá a saída do log da seguinte forma.

```
 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.
```

As linhas a seguir do modelo de trabalho especificaram essa ação.

```
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. Selecione a ação Iniciar StepEnv sessão no monitor do Deadline Cloud. Você vê a saída do log da seguinte forma.

```
 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. As linhas a seguir do modelo de trabalho especificaram essa ação.

```
 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
```

Execute um processo daemon em segundo plano a partir do ambiente de fila

Em muitos casos de uso de renderização, carregar os dados do aplicativo e da cena pode levar um tempo significativo. Se um trabalho os recarregar para cada quadro, ele passará a maior parte do tempo sobrecarregando. Geralmente, é possível carregar o aplicativo uma vez como um processo daemon em segundo plano, fazer com que ele carregue os dados da cena e, em seguida, envie comandos via comunicação entre processos (IPC) para realizar as renderizações.

Muitas das integrações de código aberto do Deadline Cloud usam esse padrão. O projeto Open Job Description fornece uma biblioteca de tempo de execução de adaptadores com padrões robustos de IPC em todos os sistemas operacionais compatíveis.

Para demonstrar esse padrão, há um pacote de tarefas de amostra independente que usa Python e código bash para implementar um daemon em segundo plano e o IPC para tarefas de comunicação com ele. O daemon é implementado em Python e escuta um SIGUSR1 sinal POSIX para saber quando processar uma tarefa. Os detalhes da tarefa são passados para o daemon em um arquivo JSON específico e os resultados da execução da tarefa são retornados como outro arquivo JSON.

Pré-requisitos

Execute as etapas a seguir para executar o pacote de tarefas de amostra com um processo daemon do repositório github de amostras do Deadline Cloud.

1. Se você não tiver um farm do Deadline Cloud com uma fila e uma frota Linux associada, siga a experiência de integração guiada no console do Deadline Cloud para criar um com as configurações padrão.

2. Se você não tiver a CLI do Deadline Cloud e o monitor do Deadline Cloud em sua estação de trabalho, siga as etapas em Configurar os remetentes do Deadline Cloud no guia do usuário.
3. Use `git` para clonar o GitHubrepositório de amostras do Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Execute a amostra do daemon

1. Use a CLI do Deadline Cloud para enviar a `job_env_daemon_process` amostra.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. No aplicativo de monitoramento Deadline Cloud, você verá o novo trabalho e poderá monitorar seu progresso. Quando a Linux frota associada à fila tiver um trabalhador disponível para executar a tarefa, ela será concluída em cerca de um minuto. Com uma das tarefas selecionadas, escolha a opção Exibir registros no menu superior direito do painel de tarefas.

   À direita, há duas ações de sessão, Iniciar DaemonProcess e Executar tarefa. O visualizador de registros no centro da janela corresponde à ação de sessão selecionada à direita.

   Selecione a opção Exibir registros de todas as tarefas. A linha do tempo mostra o restante das tarefas executadas como parte da sessão e a `Shut down DaemonProcess` ação que saiu do ambiente.

Exibir os registros do daemon

1. Nesta seção, você usa o monitor Deadline Cloud para comparar as ações da sessão com o local em que elas estão definidas no modelo de trabalho. Ela continua a partir da seção anterior.

   Abra o arquivo job_env_daemon_process/template.yaml em um editor de texto. Compare as ações da sessão com o local em que elas estão definidas no modelo de trabalho.

2. Selecione a ação da `Launch DaemonProcess` sessão no monitor do Deadline Cloud. Você verá a saída do log da seguinte forma.

```
 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
```

As linhas a seguir do modelo de trabalho especificaram essa ação.

```
  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. Selecione uma das ações de sessão Tarefa Executar: N no monitor do Deadline Cloud. Você verá a saída do log da seguinte forma.

```
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 ------------------------------------------------
```

As seguintes linhas do modelo de trabalho são as que especificaram essa ação. ```etapas:

```
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
```

Forneça inscrições para seus empregos

Você pode usar um ambiente de fila para carregar aplicativos e processar seus trabalhos. Ao criar uma frota gerenciada por serviços usando o console do Deadline Cloud, você tem a opção de criar um ambiente de fila que usa o gerenciador de pacotes conda para carregar aplicativos.

Se quiser usar um gerenciador de pacotes diferente, você pode criar um ambiente de fila para esse gerenciador. Para obter um exemplo de uso do Rez, consulteUse um gerenciador de pacotes diferente.

O Deadline Cloud fornece um canal conda para carregar uma seleção de aplicativos de renderização em seu ambiente. Eles apoiam os remetentes que o Deadline Cloud fornece para aplicativos de criação de conteúdo digital.

Você também pode carregar software para o conda-forge usar em seus trabalhos. Os exemplos a seguir mostram modelos de trabalho usando o ambiente de fila fornecido pelo Deadline Cloud para carregar aplicativos antes de executar o trabalho.

Tópicos

- Obtendo um aplicativo de um canal conda
- Use um gerenciador de pacotes diferente

Obtendo um aplicativo de um canal conda

Você pode criar um ambiente de fila personalizado para seus funcionários do Deadline Cloud que instala o software de sua escolha. Esse exemplo de ambiente de fila tem o mesmo comportamento do ambiente usado pelo console para frotas gerenciadas por serviços. Ele executa o conda diretamente para criar o ambiente.

O ambiente cria um novo ambiente virtual conda para cada sessão do Deadline Cloud que é executada em um trabalhador e, em seguida, exclui o ambiente quando termina.

O Conda armazena em cache os pacotes baixados para que eles não precisem ser baixados novamente, mas cada sessão deve vincular todos os pacotes ao ambiente.

O ambiente define três scripts que são executados quando o Deadline Cloud inicia uma sessão em um trabalhador. O primeiro script é executado quando a `onEnter` ação é chamada. Ele chama os outros dois para configurar variáveis de ambiente. Quando o script termina de ser executado, o ambiente conda fica disponível com todas as variáveis de ambiente especificadas definidas.

Para ver a versão mais recente do exemplo, consulte conda_queue_env_console_equivalent.yaml no repositório em. deadline-cloud-samples GitHub

Se você quiser usar um aplicativo que não está disponível no canal conda, você pode criar um canal conda no Amazon S3 e, em seguida, criar seus próprios pacotes para esse aplicativo. Para saber mais, consulte Crie um canal conda usando o S3.

Obtenha bibliotecas de código aberto do conda-forge

Esta seção descreve como usar bibliotecas de código aberto do `conda-forge` canal. O exemplo a seguir é um modelo de trabalho que usa o `polars` pacote Python.

O trabalho define os `CondaChannels` parâmetros `CondaPackages` e definidos no ambiente de fila que informam ao Deadline Cloud onde obter o pacote.

A seção do modelo de trabalho que define os parâmetros é:

```
- 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 a versão mais recente do exemplo completo do modelo de trabalho, consulte stage_1_self_contained_template/template.yaml. Para a versão mais recente do ambiente de filas que carrega os pacotes conda, consulte conda_queue_env_console_equivalent.yaml no repositório em. deadline-cloud-samples GitHub

Obtenha a Blender partir do canal Deadline-Cloud

O exemplo a seguir mostra um modelo de trabalho que Blender vem do canal `deadline-cloud` conda. Esse canal oferece suporte aos remetentes que o Deadline Cloud fornece para software de criação de conteúdo digital, embora você possa usar o mesmo canal para carregar o software para seu próprio uso.

Para ver uma lista do software fornecido pelo `deadline-cloud` canal, consulte Ambiente de fila padrão no Guia do usuário do AWS Deadline Cloud.

Esse trabalho define o `CondaPackages` parâmetro definido no ambiente de fila para fazer com que o Deadline Cloud seja carregado Blender no ambiente.

A seção do modelo de trabalho que define o parâmetro é:

```
- 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 a versão mais recente do exemplo completo do modelo de trabalho, consulte blender_render/template.yaml. Para a versão mais recente do ambiente de filas que carrega os pacotes conda, consulte conda_queue_env_console_equivalent.yaml no repositório em. deadline-cloud-samplesGitHub

Use um gerenciador de pacotes diferente

O gerenciador de pacotes padrão do Deadline Cloud é o conda. Se você precisar usar um gerenciador de pacotes diferente, comoRez, você pode criar um ambiente de fila personalizado que contém scripts que usam seu gerenciador de pacotes em vez disso.

Esse exemplo de ambiente de fila fornece o mesmo comportamento do ambiente usado pelo console para frotas gerenciadas por serviços. Ele substitui o gerenciador de pacotes conda por. Rez

O ambiente define três scripts que são executados quando o Deadline Cloud inicia uma sessão em um trabalhador. O primeiro script é executado quando a `onEnter` ação é chamada. Ele chama os outros dois para configurar variáveis de ambiente. Quando a execução do script termina, o Rez ambiente fica disponível com todas as variáveis de ambiente especificadas definidas.

O exemplo pressupõe que você tenha uma frota gerenciada pelo cliente que usa um sistema de arquivos compartilhado para os pacotes Rez.

Para ver a versão mais recente do exemplo, consulte rez_queue_env.yaml no repositório em. deadline-cloud-samplesGitHub

Crie um canal conda usando o S3

Se seus trabalhos precisarem executar aplicativos não disponíveis nos `conda-forge`canais `deadline-cloud`or, você pode hospedar um canal conda personalizado para servir seus próprios pacotes. Quando você cria uma fila no console do AWS Deadline Cloud (Deadline Cloud), o console adiciona um ambiente de fila conda por padrão. Para disponibilizar seus pacotes para trabalhos, adicione o canal personalizado ao ambiente de fila.

Um canal conda é um conteúdo hospedado estático que você pode hospedar de várias maneiras, inclusive em um sistema de arquivos ou em um bucket do Amazon Simple Storage Service (Amazon S3). Se seu farm do Deadline Cloud usa um sistema de arquivos compartilhado para ativos, você pode usar qualquer caminho nele como nome de canal. Você pode hospedar o canal em um bucket do Amazon S3 para um acesso mais amplo usando permissões AWS Identity and Access Management (IAM).

Você pode criar e testar pacotes localmente e depois publicá-los em um canal. Construir pacotes localmente é uma maneira fácil de começar a iterar em receitas de compilação de pacotes sem configuração de infraestrutura. Você também pode usar uma fila de criação de pacotes do Deadline Cloud para criar pacotes e publicá-los em um canal. Uma fila de criação de pacotes simplifica a

manutenção de pacotes para vários sistemas operacionais e configurações de aceleradores. Você pode atualizar versões e enviar conjuntos completos de compilações de pacotes de qualquer lugar.

Você pode configurar canais para seu estúdio e sua fazenda do Deadline Cloud de várias maneiras. Você pode ter um canal Amazon S3 e configurar todas as suas estações de trabalho e hosts de farm para usá-lo. Você também pode ter mais de um canal e configurar o espelhamento com AWS DataSync (DataSync). Por exemplo, sua fila de criação de pacotes do Deadline Cloud pode ser publicada em um canal do Amazon S3 que é espelhado localmente para estações de trabalho e hosts agrícolas locais.

Tópicos

Crie e teste pacotes localmente

Antes de publicar pacotes no Amazon S3 ou configurar a CI/CD automação em seu farm Deadline Cloud, você pode criar e testar pacotes conda em sua estação de trabalho usando um canal de sistema de arquivos local. Essa abordagem permite que você itere rapidamente as receitas localmente e verifique os pacotes.

O `rattler-build publish` comando cria uma receita, copia o pacote resultante em um canal e indexa o canal em uma única etapa. Quando você direciona um diretório do sistema de arquivos local, `rattler-build` cria e inicializa o canal automaticamente se o diretório não existir.

As instruções a seguir usam a receita de amostra Blender 4.5 do repositório de amostras do Deadline Cloud emGitHub. Você pode substituir uma receita diferente do repositório de amostras ou usar sua própria receita.

Pré-requisitos

Antes de começar, instale as seguintes ferramentas em sua estação de trabalho:

- pixi — Um gerenciador de pacotes que você usa para instalar `rattler-build` e testar pacotes. Instale o pixi a partir do pixi.sh.
- rattler-build — A ferramenta de criação de pacotes usada pelas receitas do Deadline Cloud conda. Depois de instalar o pixi, execute o seguinte comando para instalar`rattler-build`.

```
pixi global install rattler-build
```

- git — Necessário para clonar o repositório de amostras. WindowsAtivado, o git for Windows também fornece um `bash` shell, exigido por algumas das receitas de Windows amostra.

Criando e publicando um pacote em um canal local

Neste procedimento, você clona o repositório de amostras do Deadline Cloud e o usa `rattler-build publish` para criar e publicar o pacote em um canal de sistema de arquivos local.

> **Note**
>
> Aplicativos grandes podem exigir dezenas de GB de espaço livre em disco para o arquivamento de origem, arquivos extraídos e saída de compilação. Certifique-se de usar um disco com espaço disponível suficiente para a saída da compilação do pacote.

Para criar e publicar um pacote em um canal local

1. Clone o repositório de amostras do Deadline Cloud.

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

2. Mude para o diretório `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Execute o comando a seguir para criar a receita Blender 4.5 e publicar o pacote em um diretório de canal local.

   Em Linux emacOS, execute o comando a seguir.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Em Windows (cmd), execute o comando a seguir.

```
rattler-build publish blender-4.5/recipe/recipe.yaml ^
    --to file://%USERPROFILE%/my-conda-channel ^
    --build-number=+1
```

O `rattler-build publish` comando executa as seguintes ações:

- Cria o pacote a partir da receita.
- Cria o diretório do canal se o diretório não existir.
- Copia o arquivo do pacote para o canal.
- Indexa o canal para que os gerenciadores de pacotes possam encontrar o pacote.

Se a receita do pacote depender de pacotes de um canal específico, como conda-forge, adicione `-c conda-forge` ao comando.

Sobre números de compilação

A `--build-number=+1` opção seleciona automaticamente o próximo número de compilação com base no que já existe no canal de destino. A melhor prática é nunca sobrescrever um pacote em um canal. Sempre crie com um novo número de compilação se, de outra forma, o pacote tivesse o mesmo nome de arquivo. O uso `--build-number=+1` consegue isso quando você cria um canal de produção ou um canal de preparação que espelha a produção.

Se quiser controlar o número da compilação diretamente, você pode defini-lo com um valor específico, como`--build-number=7`. Se você omitir a opção, `rattler-build` usa o número de compilação definido no `recipe.yaml` arquivo.

Para obter mais informações sobre`rattler-build publish`, consulte a documentação de publicação do rattler-build.

Compilações de depuração

Se uma compilação falhar, `rattler-build` preserva o diretório de compilação para que você possa investigar. Execute o comando a seguir para abrir um shell interativo no ambiente de compilação com todas as variáveis de ambiente configuradas como estavam durante a compilação.

```
rattler-build debug shell
```

No shell de depuração, você pode modificar arquivos, executar comandos de compilação individuais e adicionar dependências para isolar o problema. Para obter mais informações, consulte Depuração de compilações na documentação do rattler-build.

Testando o pacote

Depois de criar e publicar o pacote, crie um projeto pixi temporário. Use o projeto para instalar o pacote a partir do canal local e verificar se ele funciona corretamente.

Para testar o pacote

1. Crie um diretório de teste temporário e inicialize um projeto pixi com o canal local.

   LinuxAtivado emacOS, execute os seguintes comandos.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://$HOME/my-conda-channel
   ```

   Em Windows (cmd), execute os seguintes comandos.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://%USERPROFILE%/my-conda-channel
   ```

2. Adicione o pacote ao projeto.

   ```
   pixi add blender=4.5
   ```

3. Verifique se o pacote funciona corretamente.

   ```
   pixi run blender --version
   ```

O `pixi run` comando ativa o ambiente conda para o diretório do projeto e executa o comando especificado dentro dele. O ambiente persiste no diretório do projeto, então você pode usar o mesmo `pixi run` comando em outros terminais.

Quando estiver satisfeito com o pacote, você pode publicá-lo em um canal conda do Amazon S3 para que os funcionários do Deadline Cloud possam instalar o pacote. Consulte Publicar pacotes em um canal conda do S3.

Removendo pacotes do canal

Evite remover pacotes dos canais que você usa para produção, porque os arquivos de bloqueio fazem referência a pacotes específicos por hash. A remoção de um pacote impede a recriação de ambientes a partir desses arquivos de bloqueio. Para canais de desenvolvimento e teste, você pode remover um pacote específico excluindo o `.conda` arquivo do diretório do canal e depois reindexando o canal. Primeiro, instale`rattler-index`.

```
pixi global install rattler-index
```

Em seguida, exclua o arquivo do pacote e reindexe o canal.

LinuxAtivado emacOS, execute os seguintes comandos.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

Em Windows (cmd), execute os seguintes comandos.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

Os arquivos do Package são armazenados em subdiretórios específicos da plataforma, como, ou. `linux-64` `win-64` `osx-arm64` Liste o conteúdo desses subdiretórios para encontrar o nome exato do arquivo do pacote que você deseja remover.

Limpeza

Após o teste, você pode remover o projeto de teste e o canal local.

Para limpar os recursos de teste

1. Remova o diretório do projeto de teste.

   Em Linux emacOS, execute o comando a seguir.

   ```
   rm -rf package-test-env
   ```

   Em Windows (cmd), execute o comando a seguir.

   ```
   rmdir /s /q package-test-env
   ```

2. Remova o diretório local do canal conda.

   Em Linux emacOS, execute o comando a seguir.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Em Windows (cmd), execute o comando a seguir.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Opcional) Remova o diretório `rattler-build` de saída que contém o arquivo do pacote criado.

   Em Linux emacOS, execute o comando a seguir.

   ```
   rm -rf deadline-cloud-samples/conda_recipes/output
   ```

   Em Windows (cmd), execute o comando a seguir.

   ```
   rmdir /s /q deadline-cloud-samples\conda_recipes\output
   ```

Publique pacotes em um canal conda do Amazon S3

Você pode publicar pacotes conda em um bucket do Amazon Simple Storage Service (Amazon S3) para que os funcionários do AWS Deadline Cloud (Deadline Cloud) possam instalá-los para executar trabalhos. O `rattler-build publish` comando funciona com o Amazon S3 da mesma forma

que com um canal de sistema de arquivos local. O comando pode criar uma receita e publicar o resultado, ou publicar um arquivo de pacote que você já criou. Em ambos os casos, o comando carrega o pacote no bucket e indexa o canal em uma única etapa.

O `rattler-build publish` comando é autenticado AWS usando a cadeia de credenciais padrão, portanto, ele usa sua AWS configuração como qualquer AWS ferramenta. Para obter mais informações sobre a configuração de credenciais, consulte Configuração e configurações do arquivo de credenciais no Guia do Usuário AWS Command Line Interface (AWS CLI).

Pré-requisitos

Antes de publicar pacotes no Amazon S3, preencha os seguintes pré-requisitos:

- pixi e rattler-build — Instale o pixi a partir do pixi.sh e, em seguida, instale. `rattler-build`

```
pixi global install rattler-build
```

- git — Necessário para clonar o repositório de amostras. WindowsAtivado, o git for Windows também fornece um `bash` shell, exigido por algumas das receitas de Windows amostra.
- Bucket Amazon S3 — Um bucket Amazon S3 para usar como canal conda. Você pode usar o repositório de anexos de tarefas do seu farm do Deadline Cloud ou criar um intervalo separado.
- AWS credenciais — Configure as credenciais em sua estação de trabalho usando o `aws configure` comando ou o comando. `aws login` Para obter mais informações, consulte Configuração do AWS CLI no Guia do usuário do AWS Command Line Interface .
- Permissões do IAM — (opcional) Para reduzir o escopo das permissões que suas credenciais têm, você pode usar uma política AWS Identity and Access Management (IAM) que concede somente as seguintes permissões no bucket do Amazon S3 e no prefixo do canal que você usa (por exemplo,): `/Conda/*`
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:DeleteObject`
  - `s3:ListBucket`
  - `s3:GetBucketLocation`

Publicar um pacote em um canal do Amazon S3

Use `rattler-build publish` com um `s3://` destino para publicar um pacote em seu canal conda do Amazon S3. Se o canal não existir no bucket, `rattler-build` inicializará o canal automaticamente. Antes de começar, verifique se você concluiu os pré-requisitos.

O exemplo a seguir publica a receita de amostra Blender 4.5 do repositório de amostras do Deadline Cloud em. GitHub Você pode substituir uma receita diferente do repositório de amostras ou usar sua própria receita.

> **Note**
>
> Aplicativos grandes podem exigir dezenas de GB de espaço livre em disco para o arquivamento de origem, arquivos extraídos e saída de compilação. Certifique-se de usar um disco com espaço disponível suficiente para a saída da compilação do pacote.

Para publicar um pacote em um canal do Amazon S3

1. Clone o repositório de amostras do Deadline Cloud.

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

2. Mude para o diretório `conda_recipes`.

   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Execute o comando a seguir. Substitua *`amzn-s3-demo-bucket`* pelo nome do seu bucket.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
   Conda/Default --build-number=+1
   ```

   O `/Conda/Default` prefixo organiza o canal dentro do bucket. Você pode usar um prefixo diferente, mas o prefixo deve ser consistente em todos os comandos e configurações de fila que fazem referência ao canal.

> **Sobre números de compilação**
>
> A `--build-number=+1` opção seleciona automaticamente o próximo número de compilação com base no que já existe no canal de destino. A melhor prática é nunca sobrescrever um pacote em um canal. Sempre crie com um novo número de compilação se, de outra forma, o pacote tivesse o mesmo nome de arquivo. O uso `--build-number=+1` consegue isso quando você cria um canal de produção ou um canal de preparação que espelha a produção.
>
> Se quiser controlar o número da compilação diretamente, você pode defini-lo com um valor específico, como`--build-number=7`. Se você omitir a opção, `rattler-build` usa o número de compilação definido no `recipe.yaml` arquivo.

Se a receita do pacote depender de pacotes de um canal específico, como conda-forge, adicione `-c conda-forge` ao comando.

Você também pode publicar um arquivo de pacote que você já criou, por exemplo, um `.conda` arquivo de uma compilação local. Substitua *`amzn-s3-demo-bucket`* pelo nome do seu bucket.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Inicializando ou reindexando um canal

Quando você usa `rattler-build publish` para publicar um pacote, o comando inicializa o canal automaticamente se o canal ainda não existir. Na maioria dos casos, você não precisa inicializar ou reindexar o canal manualmente.

Talvez seja necessário inicializar ou reindexar manualmente um canal nas seguintes situações:

- Você quer criar um canal vazio antes de publicar qualquer pacote, por exemplo, para verificar se o ambiente de filas do Deadline Cloud pode se conectar ao canal.
- Você carregou ou excluiu `.conda` arquivos diretamente com as ferramentas do Amazon S3 em vez de usar`rattler-build publish`, e o índice do canal está desatualizado.

Inicializando um canal vazio

Para inicializar um canal vazio, crie um `repodata.json` arquivo e faça o upload para o `noarch` subdiretório do prefixo do canal. Substitua *amzn-s3-demo-bucket* pelo nome do seu 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
```

O `/Conda/Default` prefixo deve corresponder ao prefixo do canal que seu ambiente de fila usa. Depois de inicializar o canal, você pode publicar pacotes no canal usando`rattler-build publish`.

Reindexando um canal

Se o índice do canal estiver desatualizado, use `rattler-index` para reconstruir o índice a partir dos arquivos do pacote no canal. Primeiro, instale`rattler-index`.

```
pixi global install rattler-index
```

Em seguida, reindexe o canal. Substitua *amzn-s3-demo-bucket* pelo nome do seu bucket.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

Testando o pacote

Depois de publicar o pacote, crie um projeto pixi temporário para verificar se o pacote funciona corretamente. O projeto instala o pacote a partir do canal Amazon S3.

Para testar o pacote

1. Crie um diretório de teste temporário e inicialize um projeto pixi com o canal Amazon S3. Substitua *amzn-s3-demo-bucket* pelo nome do seu bucket.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. Adicione o pacote ao projeto.

```
pixi add blender=4.5
```

3. Verifique se o pacote funciona corretamente.

```
pixi run blender --version
```

O `pixi run` comando ativa o ambiente conda para o diretório do projeto e executa o comando especificado dentro dele. O ambiente persiste no diretório do projeto, então você pode usar o mesmo `pixi run` comando em outros terminais.

Removendo pacotes do canal

Evite remover pacotes dos canais que você usa para produção, porque os arquivos de bloqueio fazem referência a pacotes específicos por hash. A remoção de um pacote impede a recriação de ambientes a partir desses arquivos de bloqueio. Para canais de desenvolvimento e teste, você pode remover um pacote específico excluindo o `.conda` arquivo do bucket e depois reindexando o canal.

Exclua o arquivo do pacote e reindexe o canal. Substitua *`amzn-s3-demo-bucket`* pelo nome do seu bucket.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

Depois de excluir o arquivo, reindexe o canal para atualizar os metadados do canal. Para obter instruções, consulte Reindexação de um canal.

Os arquivos do Package são armazenados em subdiretórios específicos da plataforma, como, ou. `linux-64` `win-64` `osx-arm64` Para listar os pacotes em um subdiretório, execute o comando a seguir.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

Limpeza

Após o teste, remova o diretório do projeto de teste.

Para limpar os recursos de teste

- Remova o diretório do projeto de teste.

Em Linux emacOS, execute o comando a seguir.

```
rm -rf package-test-env
```

Em Windows (cmd), execute o comando a seguir.

```
rmdir /s /q package-test-env
```

Compilações de depuração

Se uma compilação falhar, `rattler-build` preserva o diretório de compilação para que você possa investigar. Execute o comando a seguir para abrir um shell interativo no ambiente de compilação com todas as variáveis de ambiente configuradas como estavam durante a compilação.

```
rattler-build debug shell
```

No shell de depuração, você pode modificar arquivos, executar comandos de compilação individuais e adicionar dependências para isolar o problema. Para obter mais informações, consulte Depuração de compilações na documentação do rattler-build.

Construindo pacotes para outras plataformas

O `rattler-build publish` comando cria pacotes para o sistema operacional da estação de trabalho em que o comando é executado. Se sua frota do Deadline Cloud usa um sistema operacional diferente da sua estação de trabalho ou se seu pacote tem outros requisitos de hospedagem, você tem as seguintes opções:

- Execute `rattler-build publish` em um host que corresponda ao sistema operacional de destino. Por exemplo, use uma instância do Amazon Elastic Compute Cloud (Amazon EC2) em Linux execução para criar pacotes para uma frota. Linux
- Use uma fila de criação de pacotes do Deadline Cloud para automatizar compilações na plataforma de destino. Consulte Criar uma fila de criação de pacotes.
- (Avançado) Use a compilação cruzada para criar pacotes para uma plataforma diferente da sua estação de trabalho. Para obter mais informações, consulte Compilação cruzada na documentação do rattler-build.

Próximas etapas

Depois de publicar pacotes em seu canal conda do Amazon S3, configure suas filas do Deadline Cloud para usar o canal:

- Configure permissões de fila de produção para pacotes conda personalizados — Conceda às suas filas de produção acesso somente de leitura ao canal conda do Amazon S3.
- Adicionar um canal conda a um ambiente de fila — Configure o ambiente de fila para instalar pacotes do canal conda do Amazon S3.

Configurar permissões de fila de produção para pacotes conda personalizados

Sua fila de produção precisa de permissões somente de leitura para o `/Conda` prefixo no bucket S3 da fila. Abra a página AWS Identity and Access Management (IAM) da função associada à fila de produção e modifique a política com o seguinte:

1. Abra o console do Deadline Cloud e navegue até a página de detalhes da fila de criação de pacotes.
2. Escolha a função de serviço de fila e, em seguida, escolha Editar fila.
3. Role até a seção Função de serviço de fila e escolha Visualizar essa função no console do IAM.
4. Na lista de políticas de permissão, escolha a AmazonDeadlineCloudQueuePolicypara sua fila.
5. Na guia Permissões, escolha Editar.
6. Adicione uma nova seção à função de serviço de fila, como a seguir. *111122223333*Substitua *amzn-s3-demo-bucket* e por seu próprio bucket e conta.

```
{
   "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"
    }
   }
  },
```

Adicionar um canal conda a um ambiente de fila

Para usar o canal conda do S3, você precisa adicionar a localização do `s3://amzn-s3-demo-bucket/Conda/Default` canal ao `CondaChannels` parâmetro dos trabalhos que você envia para o Deadline Cloud. Os remetentes fornecidos com o Deadline Cloud fornecem campos para especificar canais e pacotes personalizados de conda.

Você pode evitar a modificação de cada trabalho editando o ambiente da fila conda para sua fila de produção. Use o procedimento a seguir:

1. Abra o console do Deadline Cloud e navegue até a página de detalhes da fila de produção.
2. Escolha a guia de ambientes.
3. Selecione o ambiente da fila Conda e, em seguida, escolha Editar.
4. Escolha o editor JSON e, em seguida, no script, encontre a definição do parâmetro para`CondaChannels`.
5. Edite a linha `default: "deadline-cloud"` para que ela comece com o canal conda S3 recém-criado:

   ```
   default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
   ```

Por padrão, frotas gerenciadas por serviços permitem prioridade de canal flexível para conda. Para um trabalho solicitando `blender=4.5` se Blender 4.5 está no novo canal e no `deadline-cloud` canal, o pacote será retirado do canal que estiver primeiro na lista de canais. Se uma versão de pacote especificada não for encontrada no primeiro canal, os canais subsequentes serão verificados em ordem para a versão do pacote.

Para frotas gerenciadas pelo cliente, você pode habilitar o uso de pacotes conda usando uma das amostras do ambiente conda queue no repositório de amostras do Deadline Cloud. GitHub

Crie um pacote conda para um aplicativo ou plug-in

Um pacote conda é um arquivo compactado de software escrito em qualquer idioma. O Conda oferece suporte a uma variedade de combinações de sistemas operacionais e arquiteturas, para que você possa empacotar aplicativos completosBlender, comoMaya, e Nuke junto com bibliotecas para Python e outras linguagens. Para obter mais informações sobre pacotes conda, consulte Pacotes na documentação do conda.

Para usar um pacote conda, você o instala em um ambiente virtual. Um ambiente virtual conda tem um diretório de prefixo onde os pacotes são instalados. A instalação de um pacote usa vinculação direta ou revinculação de arquivos quando há suporte, portanto, a criação de vários ambientes com os mesmos pacotes não usa espaço adicional em disco significativo. Para usar um ambiente virtual, você o ativa para definir variáveis de ambiente. A ativação executa scripts fornecidos pelos pacotes, dando a cada pacote a oportunidade de modificar o PATH ou outras variáveis de ambiente. Os pacotes Conda normalmente contêm aplicativos ou bibliotecas, mas a ativação flexível significa que eles também podem apontar para aplicativos instalados em um sistema de arquivos compartilhado.

A criação de um pacote personalizado envolve três etapas: uma receita contém as instruções de construção, um pacote é o artefato (`.conda`ou `.tar.bz2` arquivo) construído e um canal hospeda pacotes para instalação. O `rattler-build publish` comando executa todas as três etapas: ele pode criar uma receita em um pacote e publicá-la em um canal, ou pode usar um artefato de pacote diretamente para publicá-la.

A comunidade conda-forge mantém receitas de pacotes para um amplo conjunto de software de código aberto e hospeda artefatos de pacotes no canal. `conda-forge` Você pode configurar sua fila para incluir `conda-forge` como fonte de pacote e, em seguida, criar pacotes personalizados que dependam dos pacotes conda-forge para serem executados. PoisLinux, o conda-forge hospeda uma cadeia de ferramentas de compilação completa, incluindo suporte a CUDA, com opções consistentes de compilação e vinculação selecionadas. Você pode usar pacotes conda-forge como dependências em suas próprias receitas ou instalá-los junto com seus pacotes personalizados no mesmo ambiente.

Você pode combinar um aplicativo inteiro, incluindo dependências, em um pacote conda. Os pacotes que o Deadline Cloud fornece no canal de nuvem de prazos para frotas gerenciadas por serviços usam essa abordagem de reempacotamento binário. Isso organiza os mesmos arquivos de uma instalação para se adequar ao ambiente virtual conda.

> **Note**
>
> Aplicativos grandes podem exigir dezenas de GB de espaço livre em disco para o arquivamento de origem, arquivos extraídos e saída de compilação. Certifique-se de usar um disco com espaço disponível suficiente para a saída da compilação do pacote.

Package um aplicativo

Ao reempacotar um aplicativo para conda, há dois objetivos:

- A maioria dos arquivos do aplicativo deve ser separada da estrutura primária do ambiente virtual conda. Os ambientes podem então misturar o aplicativo com pacotes de outras fontes, como conda-forge.
- Quando um ambiente virtual conda é ativado, o aplicativo deve estar disponível na variável de ambiente PATH.

Para reempacotar um aplicativo para conda

1. Escreva receitas de compilação do conda que instalem o aplicativo em um subdiretório como. `$CONDA_PREFIX/opt/<application-name>` Isso o separa dos diretórios de prefixo padrão, como e. `bin` `lib`
2. Adicione links simbólicos ou scripts de inicialização `$CONDA_PREFIX/bin` para executar os binários do aplicativo.

   Como alternativa, crie scripts activate.d que o `conda activate` comando executará para adicionar os diretórios binários do aplicativo ao PATH. AtivadoWindows, onde os links simbólicos não são suportados em todos os lugares em que os ambientes podem ser criados, use scripts de inicialização do aplicativo ou ativados.d.
3. Alguns aplicativos dependem de bibliotecas não instaladas por padrão nas frotas gerenciadas pelo serviço Deadline Cloud. Por exemplo, o sistema de janelas X11 geralmente é desnecessário para trabalhos não interativos, mas alguns aplicativos ainda exigem que ele seja executado sem uma interface gráfica. Você deve fornecer essas dependências dentro do pacote criado.
4. Se o aplicativo oferecer suporte a plug-ins, forneça uma convenção clara que os pacotes de plug-ins devem seguir para se integrarem ao aplicativo em um ambiente virtual. Por exemplo, a receita de amostra de Maya 2026 documenta essa convenção para Maya plug-ins.

5. Certifique-se de seguir os contratos de direitos autorais e de licença dos aplicativos que você empacota. Recomendamos usar um bucket Amazon S3 privado para seu canal conda para controlar a distribuição e limitar o acesso de pacotes à sua fazenda.

As receitas de amostra dos pacotes no `deadline-cloud` canal estão disponíveis no repositório de amostras do Deadline Cloud emGitHub.

Package um plugin

Os plug-ins de aplicativos podem ser empacotados como seus próprios pacotes conda. Ao criar um pacote de plug-ins, siga estas diretrizes:

- Inclua o pacote do aplicativo host como dependência de compilação e execução na receita `recipe.yaml` de compilação. Use uma restrição de versão para que a receita de compilação seja instalada somente com pacotes compatíveis.
- Siga as convenções do pacote do aplicativo host para registrar o plug-in.

Pacotes de adaptadores

Algumas integrações de aplicativos do Deadline Cloud usam um adaptador que estende a interface do aplicativo para simplificar a criação de modelos de trabalho. Um adaptador é uma interface de linha de comando com suporte para executar um daemon em segundo plano, relatar status e aplicar mapeamento de caminhos. Para obter mais informações, consulte o Open Job Description Adaptor Runtime emGitHub. Por exemplo, o deadline-cloud-for-mayaon GitHub inclui uma interface gráfica de envio de tarefas integrada e um Maya adaptador que está disponível como `maya-openjd` pacote em frotas gerenciadas por serviços.

Os envios de trabalhos do remetente do Deadline Cloud GUIs incluem um valor de `CondaPackages` parâmetro que especifica os pacotes conda a serem incluídos em um ambiente virtual para execução do trabalho. O valor do `CondaPackages` parâmetro para Maya normalmente se parece `maya=2026.* maya-openjd=0.15.* maya-mtoa` e pode conter entradas alternativas para pacotes de plug-ins. Quando o ambiente de fila configura um ambiente virtual conda para executar o trabalho, ele resolve esses nomes de pacotes e restrições de versão para serem compatíveis e adiciona todos os pacotes de dependência que eles precisam executar. Cada pacote de adaptador e plug-in especifica com o que é compatível, incluindo quais versõesMaya, quais versões do Python e outras dependências.

Para criar seus próprios pacotes de adaptadores usando nossos exemplos, como a receita maya-openjd emGitHub, você pode criar os pacotes para Python e outras dependências fornecidas pelo conda-forge. Talvez seja necessário criar primeiro o prazo e openjd-adaptor-runtimeas receitas para satisfazer as dependências.

Crie uma receita de conda build para Blender

Blenderé gratuito de usar e simples de empacotar com o conda, o que o torna um bom ponto de partida para aprender a criar pacotes conda para o AWS Deadline Cloud (Deadline Cloud). A Blender Fundação fornece arquivos de aplicativos para vários sistemas operacionais. A receita de amostra Blender 4.5 no repositório de amostras do Deadline Cloud GitHub empacota esses arquivos em um pacote conda.

Entendendo a receita

O arquivo recipe.yaml define os metadados, a fonte e as opções de compilação do pacote na sintaxe do URLs modelo rattler-build. A receita especifica o número da versão uma vez e fornece uma fonte diferente URLs com base no sistema operacional.

A `build` seção em `recipe.yaml` desativa a realocação binária e as verificações de vinculação dinâmica de objetos compartilhados (DSO). Essas opções controlam como o pacote funciona quando instalado em um ambiente virtual conda em qualquer prefixo de diretório. Os valores padrão usados na `build` seção foram projetados para empacotar cada biblioteca de dependências separadamente, mas ao reempacotar binariamente um aplicativo, você precisa alterá-los. Blendernão requer nenhum ajuste de RPATH porque os arquivamentos do aplicativo são criados pensando na realocabilidade. Consulte Criar uma receita de conda para o Maya para ver um exemplo de como adicionar realocabilidade.

Durante a compilação do pacote, o script build.sh ou build_win.sh é executado para instalar arquivos no ambiente. Esses scripts copiam os arquivos de instalação`$PREFIX/opt/blender`, criam links simbólicos a partir de `$PREFIX/bin` (ativadoLinux) e configuram scripts de ativação que configuram variáveis de ambiente, como`BLENDER_LOCATION`. AtivadoWindows, o script de ativação adiciona o Blender diretório ao PATH em vez de criar links simbólicos.

O script de Windows construção usa, `bash` em vez de um `cmd.exe` arquivo.bat, para obter consistência em todas as plataformas. Você pode instalar o git for Windows`bash` para fornecer a criação de pacotes.

A receita também inclui um `deadline-cloud.yaml` arquivo que especifica as plataformas e os metadados do conda para enviar trabalhos automatizados de criação de pacotes para o Deadline Cloud. Para obter mais informações, consulte Enviar um trabalho de criação de pacote.

Construindo o Blender pacote

Use `rattler-build publish` para criar a receita Blender 4.5 e publicar o pacote em um canal. Você pode publicar em um canal local do sistema de arquivos para teste ou diretamente em um canal do Amazon S3 para uso em produção. Se você concluiu a configuração em Criar e testar pacotes localmente, execute o comando a seguir no `conda_recipes` diretório.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Para outras opções de publicação:

- Para publicar em um canal do Amazon S3, consulte Publicar pacotes em um canal conda do S3.
- Para automatizar compilações usando uma fila de criação de pacotes do Deadline Cloud, consulte Automatizar compilações de pacotes com o Deadline Cloud.

Teste seu pacote com um trabalho de Blender renderização

Depois de criar o pacote Blender 4.5, você pode testá-lo com um trabalho de renderização. Se você não tiver uma Blender cena, baixe a cena Blender 3.5 - Cozy Kitchen na página de arquivos de Blender demonstração. O repositório de amostras do Deadline Cloud contém um pacote de `blender_render` tarefas e um ambiente de fila conda que você pode usar para testes locais e na nuvem.

Testes no local

Você pode executar o modelo de trabalho em sua estação de trabalho usando a CLI do Open Job Description. Instale a CLI com. `pip`

```
pip install openjd-cli
```

No `job_bundles` diretório no repositório de amostras, execute o comando a seguir. *`/path/to/scene.blend`*Substitua pelo caminho para seu arquivo de Blender cena.

```
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
```

A `--environment` opção aplica o ambiente conda queue, que cria um ambiente virtual conda com os pacotes especificados em. `CondaPackages` O `CondaChannels` parâmetro informa ao ambiente da fila onde encontrar os pacotes. Se você publicou em um canal do Amazon S3 em vez de em um canal local, substitua o `file://` caminho pela URL do seu `s3://` canal.

Testando no Deadline Cloud

Depois de configurar sua fila de produção para usar o canal conda do Amazon S3, você pode enviar o trabalho de renderização para o Deadline Cloud. No `job_bundles` diretório no repositório de amostras, execute o comando a seguir.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Use o monitor Deadline Cloud para acompanhar o progresso do trabalho. No monitor, selecione a tarefa para o trabalho e escolha Exibir registros. Selecione a ação Launch Conda session para verificar se o pacote foi encontrado no canal Amazon S3.

Crie uma receita de conda build para Autodesk Maya

Aplicativos comerciais, como, Autodesk Maya introduzem requisitos de embalagem adicionais em comparação com aplicativos de código aberto, comoBlender. A Blenderreceita empacota um arquivo simples realocável sob uma licença de código aberto. Os aplicativos comerciais geralmente são distribuídos por meio de instaladores e exigem configuração de gerenciamento de licenças.

Considerações para aplicativos comerciais

As considerações a seguir se aplicam ao empacotar aplicativos comerciais. Os detalhes ilustram como cada um se aplica a. Maya

- Licenciamento — Entenda os direitos e restrições de licenciamento do aplicativo. Talvez seja necessário configurar um sistema de gerenciamento de licenças. Leia as perguntas frequentes sobre os benefícios da Autodesk assinatura sobre os direitos da nuvem para entender os direitos da nuvem paraMaya. Autodeskos produtos dependem de um `ProductInformation.pit` arquivo que normalmente requer acesso de administrador para serem configurados. Os recursos do produto para clientes finos oferecem uma alternativa realocável. Consulte Licenciamento Thin Client para Maya e MotionBuilder para obter mais informações.
- Dependências da biblioteca do sistema — alguns aplicativos dependem de bibliotecas não instaladas em hosts de trabalhadores de frota gerenciados por serviços. Mayadepende de bibliotecas, incluindo freetype e fontconfig. Quando essas bibliotecas estão disponíveis no gerenciador de pacotes do sistema, como `dnf` for AL2023, você pode usar o gerenciador de pacotes como fonte. Como os pacotes RPM não foram criados para serem realocáveis, você precisa usar ferramentas como `patchelf` para resolver dependências dentro do prefixo de instalação. Maya
- Acesso de administrador para instalação — Alguns instaladores exigem acesso de administrador. As frotas gerenciadas por serviços não fornecem acesso de administrador, então você precisa instalar o aplicativo em um sistema separado e criar um arquivamento dos arquivos para a compilação do pacote. O Windows instalador do Maya requer essa abordagem. O README.md na receita documenta um procedimento repetível usando uma instância recém-lançada do Amazon Elastic Compute Cloud (Amazon EC2).
- Integração de plug-ins — O Maya pacote de amostra define como `MAYA_NO_HOME=1` isolar o aplicativo da configuração em nível de usuário e adiciona caminhos de pesquisa de módulos para `MAYA_MODULE_PATH` que os pacotes de plug-ins possam colocar `.mod` arquivos no ambiente virtual. Veja a receita de amostra de Maya 2026 para ver a convenção completa de integração de plug-ins.

Entendendo a receita

O arquivo recipe.yaml define os metadados do pacote na sintaxe do modelo rattler-build. Examine as seguintes seções do arquivo:

- fonte — Faz referência aos arquivos do instalador, incluindo o hash sha256. AtivadoLinux, a fonte é o arquivo Autodesk do instalador. WindowsAtivado, a fonte inclui o arquivo do instalador e um `cleanMayaForCloud.py` script Autodesk que Maya prepara a implantação na nuvem. Atualize os hashes ao alterar os arquivos de origem, por exemplo, ao empacotar uma nova versão.

- build — Desativa a realocação binária padrão e as verificações de vinculação de DSO porque os mecanismos automáticos não funcionam corretamente para a biblioteca e os diretórios binários que usa. Maya LinuxAtivado, a receita inclui `patchelf` uma dependência de construção para definir manualmente o relativo RPATHs.
- about — Metadados sobre o aplicativo para navegar ou processar o conteúdo de um canal conda.

Os scripts de construção (build.sh paraLinux, build_win.sh paraWindows) incluem comentários explicando cada etapa. Os scripts executam as seguintes tarefas principais:

- Extraia o instalador — Extrai os arquivos Maya de instalação no prefixo conda. Os Windows scripts Linux e lidam com isso de forma diferente devido aos formatos do instalador. Consulte os scripts de construção para obter detalhes.
- Instalar dependências da biblioteca do sistema — AtivadoLinux, o script baixa e extrai as bibliotecas do sistema que Maya precisam, mas que não estão presentes, nos hosts de frota gerenciados por serviços. O script copia essas bibliotecas no Maya `lib` diretório para que estejam disponíveis no ambiente conda.
- Definir relativo RPATHs com patchelf — AtivadoLinux, o script usa `patchelf --add-rpath` para adicionar caminhos `$ORIGIN` -relativos às bibliotecas compartilhadas. Essa abordagem segue a recomendação do conda de nunca usar `LD_LIBRARY_PATH` em ambientes conda. O script corrige bibliotecas em vários níveis de diretório (`liblib/python*/site-packages`,,`lib/python*/lib-dynload`) para que cada biblioteca possa encontrar suas dependências em relação à sua própria localização. A receita segue a melhor prática de configuração `DT_RUNPATH` em vez de`DT_RPATH`, o que permite `LD_LIBRARY_PATH` substituir o caminho de pesquisa quando necessário para depuração.
- Configurar o licenciamento de thin client — O script configura o licenciamento de thin client conforme documentado por Autodesk, para que o `ProductInformation.pit` arquivo possa ser localizado no ambiente conda, em vez de exigir acesso de administrador em nível de sistema.
- Configurar scripts de ativação — Os scripts criam scripts de ativação e desativação que definem variáveis de ambiente`MAYA_LOCATION`, incluindo`MAYA_VERSION`,`MAYA_NO_HOME`, e. `MAYA_MODULE_PATH` AtivadoWindows, os scripts produzem ambos `.sh` e arquivos de `.bat` ativação porque os ambientes de fila de amostra do Deadline Cloud são usados `bash` para ativar ambientes ativadosWindows.

Construindo o Maya pacote

Antes de criar o Maya pacote, baixe o Maya instalador da sua Autodesk conta. ParaLinux, coloque o arquivo diretamente no `conda_recipes/archive_files` diretório. ParaWindows, siga o procedimento no README.md para criar o arquivo.

Use `rattler-build publish` para criar e publicar o pacote. A Maya receita requer `patchelf` uma dependência de construção doLinux, que está disponível no conda-forge. Adicione `-c conda-forge` para disponibilizar a dependência durante a compilação. No `conda_recipes` diretório, execute o comando a seguir.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Para outras opções de publicação:

- Para publicar em um canal do Amazon S3, consulte Publicar pacotes em um canal conda do S3.
- Para automatizar compilações usando uma fila de criação de pacotes do Deadline Cloud, consulte Automatizar compilações de pacotes com o Deadline Cloud. Para criar Windows pacotes Linux e ambos, use a `--all-platforms` opção com o `submit-package-job` script.

Para renderizar a amostra do toca-discos com Maya eArnold, crie os pacotes do MtoAplug-in e do Mayaadaptador. Depois de publicar os três pacotes, você pode enviar um trabalho de renderização de teste usando o pacote Turntable withMaya/Arnoldjob do repositório de amostras do Deadline Cloud. Consulte Teste seus pacotes com um trabalho de renderização do Maya.

Crie uma receita de construção conda para o adaptador Maya

O `maya-openjd` pacote fornece o adaptador que se Maya integra aos envios de trabalhos do AWS Deadline Cloud (Deadline Cloud). Quando você envia um trabalho de Maya renderização usando uma GUI de remetente do Deadline Cloud, o `CondaPackages` parâmetro é incluído `maya-openjd` ao lado do pacote. `maya` O adaptador executa a inicializaçãoMaya, a comunicação dos parâmetros de renderização e o gerenciamento do ciclo de vida do aplicativo durante uma sessão de trabalho. Para obter mais informações sobre adaptadores, consulte Pacotes de adaptadores.

Entendendo a receita

A receita de amostra maya-openjd cria o adaptador a partir do pacote fonte deadline-cloud-for-mayapublicado no PyPI. O recipe.yaml instala o pacote usando `pip` o ambiente conda.

A receita depende do Python e de dois outros pacotes do repositório de amostras do Deadline Cloud que você precisa criar primeiro:

- deadline — A biblioteca cliente do Deadline Cloud.
- openjd-adaptor-runtime— O tempo de execução do adaptador Open Job Description.

Python e outras dependências estão disponíveis no conda-forge, então adicione `-c conda-forge` ao `rattler-build publish` comando ao criar o pacote do adaptador.

Construindo o pacote de adaptadores

O `maya-openjd` pacote depende de outros dois pacotes do repositório de amostras do Deadline Cloud. Crie todos os três pacotes em ordem a partir do `conda_recipes` diretório. A `-c conda-forge` opção em cada comando é satisfazer as dependências de receitas para Python e bibliotecas Python.

Crie o `deadline` pacote.

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Crie o `openjd-adaptor-runtime` pacote.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Crie o `maya-openjd` pacote.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
```

```
    --build-number=+1 \
    -c conda-forge
```

Para outras opções de publicação:

- Para publicar em um canal do Amazon S3, consulte Publicar pacotes em um canal conda do S3.
- Para automatizar compilações usando uma fila de criação de pacotes do Deadline Cloud, consulte Automatizar compilações de pacotes com o Deadline Cloud.

Crie uma receita de compilação de conda para o plugin Autodesk Maya to Arnold (MtoA)

O Maya to Arnold (MtoA) plug-in adiciona o Arnold renderizador como uma opção internaMaya. A receita de amostra do MtoA demonstra como empacotar um plug-in como um pacote conda separado que se integra ao pacote do aplicativo host.

Entendendo a receita

O recipe.yaml especifica uma dependência do `maya` pacote para os requisitos de compilação e execução. Essa dependência usa uma restrição de versão para que o plug-in seja instalado somente com uma versão compatívelMaya.

A receita usa os mesmos arquivos de origem da Maya receita. O script de construção instala MtoA e cria um `mtoa.mod` arquivo no `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` diretório em que o Maya pacote é configurado. `MAYA_MODULE_PATH` Arnolde Maya use a mesma tecnologia de licenciamento, então o Maya pacote já inclui as informações de licenciamento necessárias. Arnold

Construindo o MtoA pacote

Crie o Maya pacote antes de compilá-lo, pois MtoA depende Maya do momento da compilação. MtoA Use `rattler-build publish` para criar e publicar o pacote. No `conda_recipes` diretório, execute o comando a seguir.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

O `rattler-build publish` comando usa o canal de destino como o canal de maior prioridade ao resolver dependências, portanto, o `maya` pacote que você publicou anteriormente está disponível automaticamente.

Para outras opções de publicação:

- Para publicar em um canal do Amazon S3, consulte Publicar pacotes em um canal conda do S3.
- Para automatizar compilações usando uma fila de criação de pacotes do Deadline Cloud, consulte Automatizar compilações de pacotes com o Deadline Cloud.

Teste seus pacotes com um trabalho de Maya renderização

Depois de criar os `maya-openjd` pacotes MayaMtoA, e, você pode testá-los com um trabalho de renderização. O repositório de amostras do Deadline Cloud contém um toca-discos com o pacote de Arnold tarefas Maya /que renderiza uma animação usando e. Maya Arnold O pacote de tarefas também é usado FFmpeg para codificar um vídeo, que está disponível no `conda-forge` canal.

Testes no local

Você pode executar o modelo de trabalho em sua estação de trabalho usando a CLI do Open Job Description. Instale a CLI com. `pip`

```
pip install openjd-cli
```

No `job_bundles` diretório no repositório de amostras, execute o comando a seguir. O `ErrorOnArnoldLicenseFail=false` parâmetro diz Arnold para renderizar com marcas d'água em vez de falhar quando nenhuma licença está disponível.

```
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
```

A `--environment` opção aplica o ambiente conda queue, que cria um ambiente virtual conda com os pacotes especificados em. `CondaPackages` O `CondaChannels` parâmetro inclui o canal local para seus pacotes personalizados e `conda-forge` para`ffmpeg`. Se você publicou em um canal do

Amazon S3 em vez de em um canal local, substitua o `file://` caminho pela URL do seu `s3://` canal.

Quando o trabalho é concluído, a saída renderizada está no `turntable_with_maya_arnold/output/` diretório.

Testando no Deadline Cloud

Depois de configurar sua fila de produção para usar o canal conda do Amazon S3, envie o trabalho de renderização para o Deadline Cloud. Adicione o `conda-forge` canal ao `CondaChannels` parâmetro em seu ambiente de fila conda para fornecer uma fonte `ffmpeg` e as dependências do Python que o adaptador exige. No `job_bundles` diretório no repositório de amostras, execute o comando a seguir.

```
deadline bundle submit turntable_with_maya_arnold
```

Use o monitor Deadline Cloud para acompanhar o progresso do trabalho. No monitor, selecione a tarefa para o trabalho e escolha Exibir registros. Selecione a ação de sessão Launch Conda para verificar se os `maya-openjd` pacotes `mayamaya-mtoa`,, e foram encontrados no canal Amazon S3.

Automatize a criação de pacotes com o Deadline Cloud

Para CI/CD fluxos de trabalho ou quando você precisa criar pacotes para vários sistemas operacionais, você pode criar uma fila de criação de pacotes do Deadline Cloud. A fila agenda trabalhos de criação em sua frota, que criam os pacotes e os publicam em seu canal conda do Amazon Simple Storage Service (Amazon S3). Isso simplifica a manutenção de compilações contínuas de pacotes para versões de software em todas as configurações necessárias.

Você pode criar uma fila de criação de pacotes usando um modelo AWS CloudFormation (CloudFormation) ou manualmente no console do Deadline Cloud. O CloudFormation modelo implanta uma fazenda completa com uma fila de produção e uma fila de criação de pacotes já configuradas. Criar a fila a partir do console oferece mais controle sobre as configurações individuais.

Crie uma fila de criação de pacotes com CloudFormation

Você pode usar um CloudFormation modelo para criar um farm do Deadline Cloud que inclui uma fila de criação de pacotes. O modelo configura uma fila de produção e uma fila de criação de pacotes com um canal conda privado do Amazon S3.

Antes de implantar o modelo, crie um bucket do Amazon S3 para armazenar os anexos do trabalho e seu canal conda. Você pode criar um bucket a partir do console do Amazon S3. Você precisa do nome do bucket ao implantar o modelo.

Para implantar o CloudFormation modelo

1. Baixe o modelo deadline-cloud-starter-farm-template.yaml do repositório de amostras do Deadline Cloud em. GitHub
2. No CloudFormation console, escolha Criar pilha e, em seguida, Com novos recursos (padrão).
3. Selecione a opção de fazer upload de um arquivo de modelo e, em seguida, faça o upload do `deadline-cloud-starter-farm-template.yaml` arquivo.
4. Insira um nome para a pilha, como**`StarterFarm`**, e forneça o nome de um bucket do Amazon S3 para anexos de tarefas e o canal conda.
5. Siga as etapas do CloudFormation console para concluir a criação da pilha.

Para obter mais informações sobre os parâmetros do modelo e as opções de personalização, consulte o README do starter farm no repositório de amostras do Deadline Cloud em. GitHub

Crie uma fila de criação de pacotes a partir do console

Siga as instruções em Criar uma fila no Guia do usuário do Deadline Cloud. Faça as seguintes alterações em:

- Na etapa 5, escolha um bucket Amazon S3 existente. Especifique um nome de pasta raiz, **`DeadlineCloudPackageBuild`** para que os artefatos de construção permaneçam separados dos anexos normais do Deadline Cloud.
- Na etapa 6, você pode associar a fila de criação de pacotes a uma frota existente ou criar uma frota totalmente nova se sua frota atual não for adequada.
- Na etapa 9, crie uma nova função de serviço para sua fila de criação de pacotes. Você modificará as permissões para dar à fila as permissões necessárias para carregar pacotes e reindexar um canal conda.

Configurar as permissões da fila de criação de pacotes

Para permitir que a fila de criação de pacotes acesse o `/Conda` prefixo no bucket Amazon S3 da fila, você deve modificar a função da fila para conceder acesso a ela. read/write A função precisa das

seguintes permissões para que os trabalhos de criação de pacotes possam carregar novos pacotes e reindexar o canal.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Abra o console do Deadline Cloud e navegue até a página de detalhes da fila de criação de pacotes.
2. Escolha a função de serviço de fila e, em seguida, escolha Editar fila.
3. Role até a seção Função de serviço de fila e escolha Visualizar essa função no console do IAM.
4. Na lista de políticas de permissão, escolha a AmazonDeadlineCloudQueuePolicypara sua fila.
5. Na guia Permissões, escolha Editar.
6. Adicione uma nova seção à função de serviço de fila, como a seguir. *111122223333*Substitua *amzn-s3-demo-bucket* e por seu próprio bucket e conta.

```
{
   "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"
    }
   }
```

```
  },
```

Envie um trabalho de criação de pacotes

Depois de criar uma fila de criação de pacotes e configurar as permissões da fila, você pode enviar trabalhos para criar pacotes conda. O `submit-package-job` script no repositório de amostras do Deadline Cloud em GitHub envia um trabalho de construção para uma receita de conda.

Você precisará do seguinte:

- A CLI do Deadline Cloud instalada em sua estação de trabalho.
- Uma sessão ativa de login do AWS Deadline Cloud Monitor (Deadline Cloud monitor).
- Um clone do repositório de amostras do Deadline Cloud.

Para enviar um trabalho de criação de pacotes

1. Abra a GUI de configuração do Deadline Cloud e defina a fazenda e a fila padrão para sua fila de criação de pacotes.

   ```
   deadline config gui
   ```

2. Mude para o `conda_recipes` diretório no repositório de amostras.

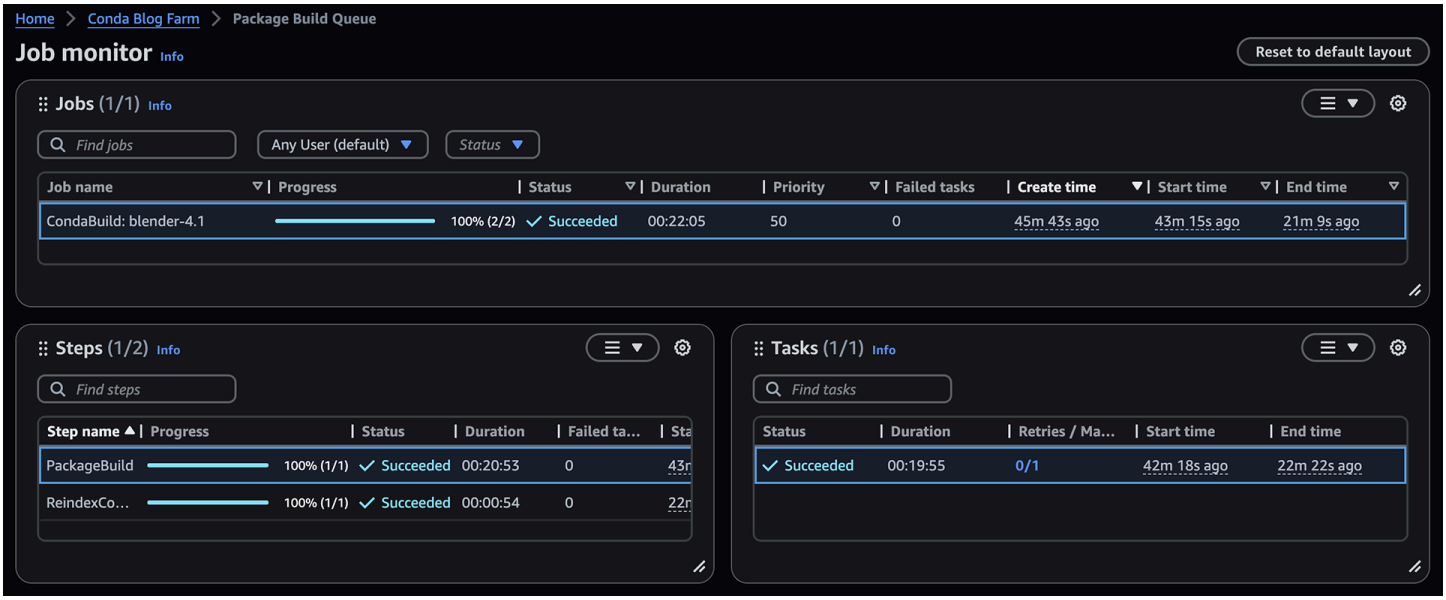
   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Execute o `submit-package-job` script com o diretório de receitas. O exemplo a seguir cria a receita Blender 4.5.

   ```
   ./submit-package-job blender-4.5/
   ```

   Se a receita exigir um arquivo de origem que você ainda não baixou, o script fornecerá instruções de download. Baixe o arquivo e execute o script novamente.

Depois de enviar o trabalho, use o monitor do Deadline Cloud para ver o progresso e o status do trabalho.



O monitor mostra as duas etapas do trabalho: criar o pacote e depois reindexar o canal conda. Quando você clica com o botão direito do mouse na tarefa da etapa de criação do pacote e escolhe Exibir registros, o monitor mostra as ações da sessão:

- Sincronizar anexos — copia os anexos da tarefa de entrada ou monta um sistema de arquivos virtual.
- Inicie o Conda — A ação do ambiente de fila. O trabalho de construção não especifica pacotes conda, então essa ação termina rapidamente.
- Launch CondaBuild Env — Cria um ambiente virtual conda com o software necessário para criar um pacote conda e reindexar um canal.
- Execução da tarefa — Cria o pacote e carrega os resultados para o Amazon S3.

Conforme as ações são executadas, elas enviam registros para a Amazon CloudWatch (CloudWatch). Quando um trabalho for concluído, selecione Exibir registros de todas as tarefas para ver registros adicionais sobre a configuração e a desmontagem do ambiente.

Execute scripts de configuração do host com privilégios de administrador

Os scripts de configuração do host permitem que você execute tarefas administrativas, como instalação de software, em seus funcionários de frota gerenciados por serviços. Esses scripts são

executados com privilégios elevados (`sudo`ativadoLinux, administrador ativadoWindows), oferecendo a flexibilidade de configurar seus trabalhadores para o seu sistema.

O Deadline Cloud executa o script depois que o trabalhador entra no `STARTING` estado e antes de executar qualquer tarefa.

> **Important**
>
> O script é executado com permissões elevadas. É sua responsabilidade garantir que o script não apresente nenhum problema de segurança.
> Ao usar um script de configuração do host, você é responsável por monitorar a integridade da sua frota.

Os usos comuns dos scripts de configuração do host incluem:

- Instalação de software que requer acesso de administrador
- Instalação de Docker contêineres
- Instalação de soluções de armazenamento em nuvem de terceiros, comoLucidLink. Para ver uma explicação passo a passo, consulte Configurar LucidLink com scripts de frota gerenciados por serviços para o Deadline Cloud no blog AWS for M&E.

Você pode criar e atualizar um script de configuração do host usando o console ou usando AWS CLI o.

Console

1. Na página de detalhes da frota, escolha a guia Configurações.
2. No campo Script, insira o script a ser executado com permissões elevadas. Você pode escolher Importar para carregar um script da sua estação de trabalho.
3. Defina um período de tempo limite em segundos para executar o script. O padrão é 300 segundos (5 minutos).
4. Escolha Salvar alterações para salvar o script.

Create with CLI

Use o AWS CLI comando a seguir para criar uma frota com um script de configuração do host. Substitua o *placeholder* texto por suas informações.

```
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

Use o AWS CLI comando a seguir para atualizar o script de configuração do host de uma frota. Substitua o *placeholder* texto por suas informações.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Os scripts a seguir demonstram:

- As variáveis de ambiente disponíveis para o script
- Essas AWS credenciais estão funcionando no shell.
- Que o script está sendo executado em um shell elevado

Linux

Use o script a seguir para mostrar que um script está sendo executado com `root` privilégios:

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Use o PowerShell script a seguir para mostrar que um script está sendo executado com privilégios 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
```

Solucionar problemas de scripts de configuração do host

Quando você executa o script de configuração do host:

- Sobre o sucesso: o trabalhador executa o trabalho
- Em caso de falha (código de saída diferente de zero ou falha):
  - O trabalhador fecha

A frota lança automaticamente um novo trabalhador usando o script de configuração de host mais recente

Para monitorar o script:

1. Abra a página da frota no console do Deadline Cloud.
2. Escolha Exibir trabalhadores para abrir o monitor do Deadline Cloud.
3. Visualize o status do trabalhador na página do monitor.

> **Tip**
>
> Ao testar os scripts de configuração do host, defina a contagem máxima de trabalhadores da frota como 1 para evitar iniciar vários trabalhadores durante a iteração no script.

Observações importantes:

- Os trabalhadores que foram desligados devido a um erro não estão disponíveis na lista de trabalhadores no monitor. Use CloudWatch Registros para visualizar os registros do trabalhador no seguinte grupo de registros:

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  Dentro desse grupo de registros, procure um fluxo chamado`worker-ZZZZZ`.
- CloudWatch O Logs retém os registros do trabalhador de acordo com o período de retenção configurado.

Monitore a execução do script de configuração do host

Com os scripts de configuração do host, você pode assumir o controle total de um funcionário do Deadline Cloud. Você pode instalar qualquer pacote de software, reconfigurar os parâmetros do sistema operacional ou montar sistemas de arquivos compartilhados. Com esse recurso avançado e a capacidade do Deadline Cloud de escalar para milhares de trabalhadores, você pode monitorar quando os scripts de configuração são executados com sucesso ou falham.

Recomendamos as seguintes soluções para monitorar a execução do script de configuração do host.

CloudWatch Monitoramento de registros

Todos os registros de configuração do host da frota são transmitidos para o grupo de CloudWatch registros da frota e, especificamente, para o fluxo de CloudWatch registros de um trabalhador. Por exemplo, `/aws/deadline/farm-123456789012/fleet-777788889999` é o grupo de registros para fazenda`123456789012`, frota`777788889999`.

Cada trabalhador provisiona um fluxo de registros dedicado, por exemplo`worker-123456789012`. Os registros de configuração do host incluem banners de registro, como Running Host Configuration Script e Finished running Host Configuration Script, código de saída: 0. O código de saída do script está incluído no banner finalizado e pode ser consultado usando CloudWatch ferramentas.

CloudWatch Informações sobre registros

CloudWatch O Logs Insights oferece recursos avançados para analisar informações de registro. Por exemplo, a seguinte consulta do Log Insights analisa o código de saída da configuração do host, classificado 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 obter mais informações sobre o CloudWatch Logs Insights, consulte Análise de dados de log com o CloudWatch Logs Insights no Guia do usuário do Amazon CloudWatch Logs.

Registro estruturado do agente de trabalho

O agente de trabalho do Deadline Cloud publica registros JSON estruturados no. CloudWatch O agente do trabalhador oferece uma ampla variedade de registros estruturados para analisar a saúde do trabalhador. Para obter mais informações, consulte o login do agente Deadline Cloud Worker GitHub.

Os atributos dos registros estruturados são descompactados em campos no Log Insights. Você pode usar esse CloudWatch recurso para contar e analisar as falhas de inicialização da configuração do host. Por exemplo, uma consulta de contagem e compartimento pode ser usada para determinar com que frequência as falhas ocorrem:

```
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 e alarmes

Você pode configurar filtros de CloudWatch métricas para gerar CloudWatch métricas a partir de registros. Os filtros métricos permitem criar alarmes e painéis para monitorar a execução do script de configuração do host.

Para criar um filtro de métricas

1. Abra o CloudWatch console.
2. No painel de navegação, escolha Registros e, em seguida, Grupos de registros.
3. Selecione o grupo de registros da sua frota.
4. Escolha Criar filtro de métrica.
5. Defina seu padrão de filtro usando uma das seguintes opções:

   - Para métricas de sucesso:

     ```
     {$.message = "*Worker Agent host configuration succeeded.*"}
     ```

   - Para métricas de falha:

     ```
     {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
      exit code*"}
     ```

6. Escolha Avançar para criar uma métrica com os seguintes valores:

   - Namespace métrico: Seu namespace métrico (por exemplo,) **`MyDeadlineFarm`**
   - Nome da métrica: o nome da métrica solicitada (por exemplo,**`host_config_failure`**)
   - Valor métrico: **`1`** (cada instância é uma contagem de 1)
   - Valor padrão: Deixe em branco
   - Unidade: **`Count`**

Depois de criar filtros de métricas, você pode configurar CloudWatch alarmes padrão para agir em taxas elevadas de falha na configuração do host ou adicionar as métricas a um CloudWatch painel para day-to-day operações e monitoramento.

Para obter mais detalhes, consulte Sintaxe de filtros e padrões no Guia do usuário do Amazon CloudWatch Logs.

Usando licenças de software com o Deadline Cloud

O Deadline Cloud fornece dois métodos para fornecer licenças de software para seus trabalhos:

- Licenciamento baseado no uso (UBL) — rastreia e fatura com base no número de horas que sua frota usa para processar um trabalho. Não há um número definido de licenças para que sua frota possa ser expandida conforme necessário. O UBL é padrão para frotas gerenciadas por serviços. Para frotas gerenciadas pelo cliente, você pode conectar um endpoint de licença do Deadline Cloud para UBL. A UBL fornece licenças para seus funcionários do Deadline Cloud renderizarem, mas não fornece licenças para seus aplicativos de DCC.
- Traga sua própria licença (BYOL) — permite que você use licenças de software existentes com suas frotas gerenciadas por serviços ou clientes. Você pode usar o BYOL para se conectar a servidores de licenças de software não suportado pelas licenças baseadas no uso do Deadline Cloud. Você pode usar o BYOL com frotas gerenciadas por serviços conectando-se a um servidor de licenças personalizado.

Tópicos

Combinando BYOL e UBL

Você pode combinar BYOL e UBL para que seus funcionários usem primeiro suas licenças existentes e retornem automaticamente às licenças baseadas no uso do Deadline Cloud quando suas licenças BYOL estiverem esgotadas. Essa abordagem é útil quando você tem um número limitado de licenças existentes, mas precisa escalar além dessa capacidade durante picos de carga de trabalho.

Como funciona o licenciamento combinado

Quando você configura o licenciamento combinado, o ambiente de fila configura as variáveis do ambiente de licença para que o servidor de licenças BYOL seja listado antes do endpoint da licença UBL. A maioria dos aplicativos de terceiros verifica os servidores de licenças na ordem em que aparecem na variável de ambiente. Quando um trabalhador solicita uma licença, o aplicativo primeiro

entra em contato com seu servidor de licenças BYOL. Se nenhuma licença BYOL estiver disponível, o aplicativo retornará ao endpoint da licença UBL.

O modelo de ambiente de fila BYOL fornecido em Conecte frotas gerenciadas por serviços a um servidor de licenças personalizado configura esse comportamento de fallback automaticamente. O script Python no ambiente de fila acrescenta o endereço do servidor de licenças BYOL às variáveis existentes do ambiente de licença UBL. Para usar o licenciamento combinado, mantenha as seções UBL no script do ambiente de filas para os produtos em que você deseja um comportamento de fallback.

Para usar somente BYOL sem UBL fallback para um produto específico, remova a seção UBL desse produto do script e adicione a variável de ambiente de licença diretamente à `variables` seção do ambiente de fila. Por exemplo, para usar somente BYOL para Cinema 4D, remova a seção Cinema 4D do script e adicione-a `g_licenseServerRLM: 127.0.0.1:7057` à seção. `variables`

Exemplo: usando licenças BYOL Cinema 4D com UBL fallback

Considere um estúdio que tenha licenças existentes do Cinema 4D em um servidor de licenças em sua rede local. O estúdio quer usar essas licenças para renderização do Deadline Cloud, mas também quer escalar além da contagem de licenças recorrendo à UBL quando todas as licenças BYOL estiverem em uso.

Para configurar essa configuração, siga as etapas Conecte frotas gerenciadas por serviços a um servidor de licenças personalizado e faça as seguintes alterações no modelo de ambiente de filas:

Para configurar licenças BYOL Cinema 4D com UBL fallback

1. Defina o `LicenseInstanceId` parâmetro como o ID da instância do Amazon Elastic Compute Cloud (Amazon EC2) do servidor de licenças ou proxy que tem acesso ao servidor de licenças Cinema 4D.
2. Defina o `LicensePorts` parâmetro para incluir a porta `7057` (a porta de licença do Cinema 4D RLM).
3. No script Python, mantenha a seção Cinema 4D que anexa o servidor BYOL à configuração da 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']}")
```

Essa configuração é definida `g_licenseServerRLM` como`127.0.0.1:7057;UBL_endpoint:7057`. O Cinema 4D verifica primeiro o servidor BYOL. `127.0.0.1:7057` Se nenhuma licença estiver disponível, o Cinema 4D retornará ao endpoint da UBL.

4. Remova as seções dos produtos que você não usa (por exemplo, Arnold, Nuke ou SideFX) para manter a configuração limpa.

Se você também tiver outros produtos que usam somente BYOL sem UBL fallback, adicione essas variáveis de ambiente de licença diretamente à `variables` seção do ambiente de fila e remova as seções correspondentes do script Python.

Considerações sobre o licenciamento combinado

Lembre-se das seguintes considerações ao usar o licenciamento combinado:

- Alguns aplicativos não oferecem suporte a vários servidores de licenças em uma única variável de ambiente. Por exemplo, o V-Ray usa um arquivo de configuração XML em vez disso. O modelo de ambiente de fila manipula a configuração do V-Ray separadamente. Para obter mais informações, consulte a seção V-Ray no modelo de ambiente de fila em. Conecte frotas gerenciadas por serviços a um servidor de licenças personalizado
- A ordem dos servidores de licenças na variável de ambiente determina a prioridade. Liste primeiro o servidor BYOL para que suas licenças existentes sejam consumidas antes das licenças UBL.
- Nos Windows trabalhadores, separe as entradas do servidor de licenças nas variáveis de ambiente com ponto e vírgula (`;`) em vez de dois pontos (). `:` Para obter mais informações sobre como configurar variáveis de ambiente de licença, consulteConecte frotas gerenciadas pelo cliente a um endpoint de licença.
- Para usar o UBL fallback com frotas gerenciadas pelo cliente, configure um endpoint de licença. Para obter mais informações, consulte Conecte frotas gerenciadas pelo cliente a um endpoint de licença.

Conecte frotas gerenciadas por serviços a um servidor de licenças personalizado

Você pode trazer seu próprio servidor de licenças para usar com uma frota gerenciada por serviços do Deadline Cloud. Para trazer sua própria licença, você pode configurar um servidor de licenças

usando um ambiente de fila em sua fazenda. Para configurar seu servidor de licenças, você já deve ter uma fazenda e uma fila configuradas.

A forma como você se conecta a um servidor de licenças de software depende da configuração da sua frota e dos requisitos do fornecedor do software. Normalmente, você acessa o servidor de duas maneiras:

- Diretamente para o servidor de licenças. Seus funcionários obtêm uma licença do servidor de licenças do fornecedor de software usando a Internet. Todos os seus funcionários devem ser capazes de se conectar ao servidor.
- Por meio de um proxy de licença. Seus trabalhadores se conectam a um servidor proxy em sua rede local. Somente o servidor proxy tem permissão para se conectar ao servidor de licenças do fornecedor pela Internet.

Com as instruções abaixo, você usa o Amazon EC2 Systems Manager (SSM) para encaminhar portas de uma instância de trabalho para seu servidor de licenças ou instância proxy. No exemplo abaixo, se seu servidor de licenças não puder fornecer uma licença, o licenciamento baseado no uso do Deadline Cloud será usado. Remova as seções que não se aplicam ao seu funil ou aos produtos para os quais você não deseja usar o licenciamento baseado no uso depois de esgotar suas licenças.

Tópicos

Etapa 1: Configurar o ambiente de filas

Você pode configurar um ambiente de fila em sua fila para acessar seu servidor de licenças. Primeiro, verifique se você tem uma AWS instância configurada com acesso ao servidor de licenças usando um dos seguintes métodos:

- Servidor de licenças — A instância hospeda os servidores de licenças diretamente.
- Proxy de licença — A instância tem acesso de rede ao servidor de licenças e encaminha as portas do servidor de licenças para o servidor de licenças. Para obter detalhes sobre como configurar

uma instância de proxy de licença, consulteEtapa 2: configuração da instância proxy de licença (opcional).

Para obter informações sobre como configurar variáveis de ambiente de licença, consulteEtapa 3: Conectar um aplicativo de renderização a um endpoint. Para uma configuração personalizada do servidor de licenças, o endereço do servidor de licenças permanece localhost em vez do endpoint da Amazon VPC.

Para adicionar as permissões necessárias à função de fila

1. No console do Deadline Cloud, escolha Ir para o painel.
2. No painel, selecione a fazenda e, em seguida, a fila que você deseja configurar.
3. Em Detalhes da fila > função de serviço, selecione a função.
4. Escolha Adicionar permissão e, em seguida, escolha Criar política embutida.
5. Selecione o editor de políticas JSON e, em seguida, copie e cole o texto a seguir no 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 salvar a nova política, substitua os seguintes valores no texto da política:

   - `region`Substitua pela AWS região onde sua fazenda está localizada

- `instance_id`Substitua pelo ID da instância do servidor de licenças ou instância proxy que você está usando
- `account_id`Substitua pelo número AWS da conta que contém sua fazenda

7. Escolha Próximo.
8. Para o nome da política, insira**`LicenseForwarding`**.
9. Escolha Criar política para salvar suas alterações e criar a política com as permissões necessárias.

Para adicionar um novo ambiente de fila à fila

1. No console do Deadline Cloud, escolha Ir para o painel, caso ainda não tenha feito isso.
2. No painel, selecione a fazenda e, em seguida, a fila que você deseja configurar.
3. Escolha Ambientes de fila > Ações > Criar novo com YAML.
4. Copie e cole o texto a seguir no editor de scripts 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 salvar o ambiente de fila, faça as seguintes alterações no texto do ambiente conforme necessário:

   - Atualize os valores padrão dos seguintes parâmetros para refletir seu ambiente:
     - LicenseInstanceID — O ID da instância Amazon EC2 do seu servidor de licenças ou instância proxy
     - LicenseInstanceRegion— A AWS região que contém sua fazenda
     - LicensePorts— Uma lista separada por vírgulas de portas a serem encaminhadas para o servidor de licenças ou instância proxy (por exemplo, 2700.2701)
   - Se você quiser usar o licenciamento baseado no uso (UBL) após o esgotamento da opção Traga sua própria licença (BYOL), verifique se a porta está correta para seu servidor de licenças. Se você não quiser usar o UBL depois de ficar sem BYOL, adicione todas as variáveis de ambiente de licenciamento necessárias à seção de variáveis.

     Essas variáveis devem direcionar o DCCs para localhost na porta do servidor de licenças. Por exemplo, se seu servidor de licenças Foundry estiver escutando na porta 6101, você adicionaria a variável as. **`foundry_LICENSE: 6101@localhost`**
6. (Opcional) Você pode deixar a prioridade definida como 0 ou alterá-la para ordenar a prioridade de forma diferente entre vários ambientes de fila.
7. Escolha Criar ambiente de fila para salvar o novo ambiente.

   Com o ambiente de fila definido, os trabalhos enviados a essa fila recuperarão as licenças do servidor de licenças configurado.

Etapa 2: configuração da instância proxy de licença (opcional)

Como alternativa ao uso de um servidor de licenças, você pode usar um proxy de licença. Para criar um proxy de licença, crie uma nova instância do Amazon Linux 2023 que tenha acesso de rede ao

servidor de licenças. Se necessário, você pode configurar esse acesso usando uma conexão VPN. Para obter mais informações, consulte Conexões VPN no Guia do usuário da Amazon VPC.

Para configurar uma instância de proxy de licença para o Deadline Cloud, siga as etapas deste procedimento. Execute as seguintes etapas de configuração nessa nova instância para permitir o encaminhamento do tráfego de licenças para o seu servidor de licenças:

1. Para instalar o HAProxy pacote, digite

```
sudo yum install haproxy
```

2. Atualize a seção listen license-server do arquivo de configuração/etc/haproxy/haproxy.cfg com o seguinte:

   a. Substitua LicensePort1 e LicensePort2 pelos números de porta a serem encaminhados para o servidor de licenças. Adicione ou remova valores separados por vírgula para acomodar o número necessário de portas.

   b. LicenseServerHostSubstitua pelo nome do host ou endereço IP do servidor de licenças.

```
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 ativar e iniciar o HAProxy serviço, execute os seguintes comandos:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Depois de concluir as etapas, as solicitações de licença enviadas ao localhost a partir do ambiente da fila de encaminhamento devem ser encaminhadas para o servidor de licenças especificado.

Etapa 3: configuração CloudFormation do modelo

Você pode usar um CloudFormation modelo para configurar uma fazenda inteira para usar seu próprio licenciamento.

1. Modifique o modelo fornecido na próxima etapa para adicionar as variáveis de ambiente de licenciamento necessárias à seção de variáveis em BYOLQueueAmbiente.
2. Use o CloudFormation modelo a seguir.

```
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. Ao implantar o CloudFormation modelo, forneça os seguintes parâmetros:

   - Atualize o LicenseInstanceID com o ID de instância do Amazon EC2 do seu servidor de licenças ou instância proxy
   - Atualize o LicensePortscom uma lista separada por vírgulas de portas a serem encaminhadas para o servidor de licenças ou instância proxy (por exemplo, 2700.2701)

- Adicione as variáveis de ambiente da licença substituindo-as **`example_LICENSE: 2700@localhost`** no modelo

4. Implante o modelo para configurar sua fazenda com o recurso de trazer sua própria licença.

Conecte frotas gerenciadas pelo cliente a um endpoint de licença

O servidor de licenças baseado no uso do AWS Deadline Cloud fornece licenças sob demanda para produtos selecionados de terceiros. Com licenças baseadas no uso, você pode pagar conforme o uso. Você só é cobrado pelo tempo usado. O licenciamento baseado em uso fornece licenças para seus funcionários do Deadline Cloud renderizarem, mas não fornece licenças para seus aplicativos de DCC.

O servidor de licenças baseado no uso do Deadline Cloud pode ser usado com qualquer tipo de frota, desde que os funcionários do Deadline Cloud possam se comunicar com o servidor de licenças. O servidor de licenças é configurado automaticamente em frotas gerenciadas por serviços. A configuração a seguir só é necessária para frotas gerenciadas pelo cliente.

Para criar o servidor de licenças, você precisa de um grupo de segurança para a VPC da sua fazenda que permita tráfego para licenças de terceiros.

Tópicos

Etapa 1: criar um grupo de segurança

Use o console Amazon VPC para criar um grupo de segurança para a VPC da sua fazenda. Configure o grupo de segurança para permitir as seguintes regras de entrada:

- Autodesk Maya e Arnold — 2701 - 2702, TCP,, IPv4 IPv6
- Cinema 4D — 7057, TCP,, IPv4 IPv6
- Fundição Nuke — 6101, TCP,, IPv4 IPv6
- Gigante vermelho — 7055, TCP, IPV4

- Redshift — 7054, TCP,, IPv4 IPv6
- SideFX Houdini, Mantra e Karma — 1715 - 1717, TCP,, IPv4 IPv6
- VRay — 30304, TCP, IPV4

A origem de cada regra de entrada é o grupo de segurança do trabalhador da frota.

Para obter mais informações sobre a criação de um grupo de segurança, consulte Criar um grupo de segurança no guia do usuário da Amazon Virtual Private Cloud.

Etapa 2: configurar o endpoint da licença

Um endpoint de licença fornece acesso aos servidores de licenças para produtos de terceiros. As solicitações de licença são enviadas para o endpoint da licença. O endpoint os encaminha para o servidor de licenças apropriado. O servidor de licenças rastreia os limites e direitos de uso. A criação de um endpoint de licença no Deadline Cloud provisiona um endpoint de AWS PrivateLink interface em sua VPC. Esses endpoints são cobrados de acordo com o preço padrão AWS PrivateLink . Para obter mais informações, consulte Preços do AWS PrivateLink.

Com as permissões apropriadas, você pode criar seu endpoint de licença. Para saber a política necessária para criar um endpoint de licença, consulte Política para permitir a criação de um endpoint de licença.

Você pode criar seu endpoint de licença em seu painel no console do Deadline Cloud.

1. No painel de navegação esquerdo, escolha Endpoints de licença e, em seguida, escolha Create license endpoint.
2. Na página Criar endpoint de licença, preencha o seguinte:
   - Selecione uma VPC.
   - Selecione as sub-redes que contêm seus trabalhadores do Deadline Cloud. Você pode selecionar até 10 sub-redes.
   - Selecione o grupo de segurança que você criou na etapa 1. Você pode selecionar até 10 grupos de segurança para cenários mais complicados.
   - (Opcional) Escolha Adicionar nova tag e adicione uma ou mais tags. É possível adicionar até 50 tags.
3. Escolha Criar endpoint de licença. Quando o endpoint da licença é criado, ele é exibido na página dos endpoints da licença.

4. Na seção de produtos medidos, escolha Adicionar produtos e, em seguida, selecione os produtos que você deseja adicionar ao seu endpoint de licença. Escolha Adicionar.

Para remover um produto de um endpoint de licença, na seção de produtos medidos, selecione o produto e escolha Remover. Na confirmação, escolha Remover novamente.

Etapa 3: Conectar um aplicativo de renderização a um endpoint

Depois que o endpoint da licença é configurado, os aplicativos o usam da mesma forma que usam um servidor de licenças de terceiros. Normalmente, você configura o servidor de licenças para o aplicativo definindo uma variável de ambiente ou outra configuração do sistema, como uma chave de registro do Microsoft Windows, como uma porta e endereço do servidor de licenças.

Para obter o nome DNS do endpoint da licença, selecione o endpoint da licença no console e escolha o ícone de cópia na seção Nome DNS.

Exemplos de configuração

Example— Autodesk Maya e Arnold

> **Note**
>
> Você pode usar o Autodesk Maya e o Arnold juntos ou separadamente. Use a porta 2702 para o Autodesk Maya e a porta 2701 para Arnold.

Para o Autodesk Maya, defina a variável `ADSKFLEX_LICENSE_FILE` de ambiente como:

```
2702@VPC_Endpoint_DNS_Name
```

Para Arnold, defina a variável `ADSKFLEX_LICENSE_FILE` de ambiente como:

```
2701@VPC_Endpoint_DNS_Name
```

Para o Autodesk Maya e o Arnold, defina a variável de ambiente como: `ADSKFLEX_LICENSE_FILE`

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Para Windows trabalhadores, use ponto e vírgula (;) em vez de dois pontos (:) para separar os pontos finais.

Example— Cinema 4D

Defina a variável de ambiente `g_licenseServerRLM` como:

```
VPC_Endpoint_DNS_Name:7057
```

Depois de criar a variável de ambiente, você poderá renderizar uma imagem usando uma linha de comando semelhante 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— Fundição Nuke

Defina a variável de ambiente `foundry_LICENSE` como:

```
6101@VPC_Endpoint_DNS_Name
```

Para testar se o licenciamento está funcionando corretamente, você pode executar o Nuke em um terminal:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Gigante vermelho

Defina a variável de ambiente `redshift_LICENSE` como:

```
7055@VPC_Endpoint_DNS_Name
```

Observe que Red Giant e Redshift têm a mesma variável de `redshift_LICENSE` ambiente. Se você quiser usar os dois aplicativos, defina a variável de ambiente como:

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Para Windows trabalhadores, use ponto e vírgula (;) em vez de dois pontos (:) para separar os pontos finais.

Para testar se o licenciamento está funcionando corretamente, verifique se você tem o After Effects e o Red Giant instalados. Em seguida, você pode renderizar um projeto usando um comando semelhante 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 a variável de ambiente `redshift_LICENSE` como:

```
7054@VPC_Endpoint_DNS_Name
```

Depois de criar a variável de ambiente, você poderá renderizar uma imagem usando uma linha de comando semelhante a esta:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SideFX Houdini, Mantra e Karma

Execute este 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 testar se o licenciamento está funcionando corretamente, você pode renderizar uma cena do Houdini por meio deste comando:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Defina a variável de ambiente `VRAY_AUTH_CLIENT_SETTINGS` como:

```
licset://VPC_Endpoint_DNS_Name:30304
```

Defina a variável de ambiente `VRAY_AUTH_CLIENT_FILE_PATH` como:

```
/null
```

Para testar se o licenciamento está funcionando corretamente, você pode renderizar uma imagem VRay usando um comando semelhante a este:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Etapa 4: excluir um endpoint de licença

Ao excluir sua frota gerenciada pelo cliente, lembre-se de excluir seu endpoint de licença. Se você não excluir o endpoint da licença, você continuará sendo cobrado pelos custos AWS PrivateLink fixos

Você pode excluir seu endpoint de licença do seu painel no console do Deadline Cloud.

1. No painel de navegação esquerdo, escolha Endpoints de licença.
2. Selecione o endpoint que você deseja excluir e escolha excluir. Em seguida, escolha excluir novamente para confirmar.

Usando agentes de IA com o Deadline Cloud

Use agentes de IA para escrever pacotes de tarefas, desenvolver pacotes conda e solucionar problemas de trabalhos no Deadline Cloud. Este tópico explica o que são os agentes de IA, os principais pontos para trabalhar com eles de forma eficaz e os recursos para ajudar os agentes a entender o Deadline Cloud.

Um agente de IA é uma ferramenta de software que usa um modelo de linguagem grande (LLM) para realizar tarefas de forma autônoma. Os agentes de IA podem ler e gravar arquivos, executar comandos e iterar soluções com base no feedback. Os exemplos incluem ferramentas de linha de comando, como o Kiro e os assistentes integrados ao IDE.

Pontos-chave para trabalhar com agentes de IA

Os pontos principais a seguir ajudam você a obter melhores resultados ao usar agentes de IA com o Deadline Cloud.

- Forneça uma base — os agentes de IA têm melhor desempenho quando têm acesso a documentação, especificações e exemplos relevantes. Você pode fornecer uma base direcionando o agente para páginas de documentação específicas, compartilhando códigos de exemplo existentes como referências, clonando repositórios de código aberto relevantes no espaço de trabalho local e fornecendo documentação para aplicativos de terceiros.
- Especifique os critérios de sucesso — defina o resultado esperado e os requisitos técnicos para o agente. Por exemplo, ao pedir a um agente que desenvolva um pacote de tarefas, especifique as entradas, os parâmetros e as saídas esperadas da tarefa. Se você não tiver certeza sobre as especificações, peça ao agente que proponha as opções primeiro e depois refine os requisitos juntos.
- Habilite um ciclo de feedback — os agentes de IA interagem com mais eficiência quando podem testar suas soluções e receber feedback. Em vez de esperar uma solução funcional na primeira tentativa, dê ao agente a capacidade de executar a solução e analisar os resultados. Essa abordagem funciona bem quando o agente pode acessar atualizações de status, registros e erros de validação. Por exemplo, ao desenvolver um pacote de tarefas, permita que o agente envie a tarefa e revise os registros.
- Espere iterar — mesmo com um bom contexto, os agentes podem se desviar do caminho certo ou fazer suposições que não combinam com seu ambiente. Observe como o agente aborda a tarefa e forneça orientação ao longo do caminho. Adicione o contexto ausente se o agente tiver dificuldades, ajude a encontrar erros apontando para arquivos de log específicos, refine os

requisitos à medida que os descobre e adicione requisitos negativos para declarar explicitamente o que o agente deve evitar.

Recursos para o contexto do agente

Os recursos a seguir ajudam os agentes de IA a entender os conceitos do Deadline Cloud e a produzir resultados precisos.

- Servidor Deadline Cloud Model Context Protocol (MCP) — Para agentes que oferecem suporte ao Model Context Protocol, o repositório deadline-cloud contém o cliente Deadline Cloud, que inclui um servidor MCP para interagir com trabalhos.
- AWSServidor MCP de documentação — Para agentes que oferecem suporte ao MCP, configure o servidor MCP de AWS documentação para dar ao agente acesso direto à AWS documentação, incluindo o Guia do usuário e o Guia do desenvolvedor do Deadline Cloud.
- Especificação Open Job Description — A especificação Open Job Description em GitHub define o esquema para modelos de trabalho. Consulte esse repositório quando os agentes precisarem entender a estrutura e a sintaxe dos modelos de trabalho.
- deadline-cloud-samples— O deadline-cloud-samplesrepositório contém exemplos de pacotes de trabalho, receitas de conda e CloudFormation modelos para aplicativos e casos de uso comuns.
- organização aws-deadline — A GitHub organização aws-deadline GitHub contém plug-ins de referência para muitos aplicativos de terceiros que você pode usar como exemplos para outras integrações.

Exemplo de solicitação: Escrevendo um pacote de tarefas

O exemplo de prompt a seguir demonstra como usar um agente de IA para criar pacotes de tarefas que treinam um adaptador LoRa (Low-Rank Adaptation) para gerar imagens de IA. O prompt ilustra os principais pontos discutidos anteriormente: ele fornece uma base apontando para repositórios relevantes, define critérios de sucesso para os resultados do pacote de tarefas e descreve um ciclo de feedback para o desenvolvimento 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.
```

Nuvem de AWS prazos de monitoramento

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do AWS Deadline Cloud (Deadline Cloud) e de suas AWS soluções. Colete dados de monitoramento de todas as partes da sua AWS solução para que você possa depurar com mais facilidade uma falha multiponto, caso ocorra. Antes de começar a monitorar o Deadline Cloud, você deve criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

AWSe o Deadline Cloud fornecem ferramentas que você pode usar para monitorar seus recursos e responder a possíveis incidentes. Algumas dessas ferramentas fazem o monitoramento para você, outras requerem intervenção manual. Você deve automatizar as tarefas de monitoramento o máximo possível.

- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. Você pode coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas EC2 instâncias da Amazon e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

  O Deadline Cloud tem três CloudWatch métricas.
- O Amazon CloudWatch Logs permite que você monitore, armazene e acesse seus arquivos de log de EC2 instâncias da Amazon e de outras fontes. CloudTrail CloudWatch Os registros podem monitorar as informações nos arquivos de log e notificá-lo quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- A Amazon EventBridge pode ser usada para automatizar seus AWS serviços e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos

ou alterações de recursos. Os eventos dos AWS serviços são entregues quase EventBridge em tempo real. Você pode escrever regras simples para determinar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Para obter mais informações, consulte o Guia EventBridge do usuário da Amazon.

- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por ou em nome de sua AWS conta e entrega os arquivos de log para um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaramAWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o Guia do usuário do AWS CloudTrail.

Tópicos

Registrando chamadas de Deadline Cloud API usando AWS CloudTrail

Deadline Cloud é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou um AWS service (Serviço da AWS). CloudTrail captura todas as chamadas de API Deadline Cloud como eventos. As chamadas capturadas incluem chamadas do Deadline Cloud console e chamadas de código para as operações Deadline Cloud da API. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita Deadline Cloud, o endereço IP do qual a solicitação foi feita, quando foi feita e detalhes adicionais.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

CloudTrail está ativo Conta da AWS quando você cria a conta e você tem acesso automático ao histórico de CloudTrail eventos. O histórico de CloudTrail eventos fornece um registro visível, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento registrados em um. Região da AWS Para obter mais informações, consulte Trabalhando com o histórico de CloudTrail eventos no Guia AWS CloudTrail do usuário. Não há CloudTrail cobrança pela visualização do histórico de eventos.

Para um registro contínuo dos eventos dos Conta da AWS últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do CloudTrailLake.

CloudTrail trilhas

Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Todas as trilhas criadas usando o Console de gerenciamento da AWS são multirregionais. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. É recomendável criar uma trilha multirregional porque você captura todas as atividades Regiões da AWS em sua conta. Ao criar uma trilha de região única, é possível visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte Criar uma trilha para a Conta da AWS e Criar uma trilha para uma organização no Guia do usuário do AWS CloudTrail .

Você pode entregar uma cópia dos seus eventos de gerenciamento contínuos para o bucket do Amazon S3 sem nenhum custo CloudTrail criando uma trilha. No entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços. Para receber informações sobre a definição de preços do Amazon S3, consulte Definição de preços do Amazon S3.

CloudTrail Armazenamentos de dados de eventos em Lake

CloudTrail O Lake permite que você execute consultas baseadas em SQL em seus eventos. CloudTrail O Lake converte eventos existentes no formato JSON baseado em linhas para o formato Apache ORC. O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de seletores de eventos avançados. Os seletores que aplicados a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para consulta. Para obter mais informações sobre o CloudTrail Lake, consulte Trabalhando com o AWS CloudTrail Lake no Guia AWS CloudTrail do usuário.

CloudTrail Os armazenamentos e consultas de dados de eventos em Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a opção de preço que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Deadline Cloud eventos de dados em CloudTrail

Os Eventos de dados fornecem informações sobre as operações de recursos realizadas em um recurso (por exemplo, leitura ou gravação em um objeto do Amazon S3). Também são conhecidas como operações de plano de dados. Os eventos de dados costumam ser atividades de alto volume. Por padrão, CloudTrail não registra eventos de dados. O histórico de CloudTrail eventos não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Você pode registrar eventos de dados para os tipos de Deadline Cloud recursos usando o CloudTrail console ou AWS CLI as operações CloudTrail da API. Para saber mais sobre como registrar eventos de dados em log, consulte Registrar eventos de dados com o Console de gerenciamento da AWS e Registrar eventos de dados com a AWS Command Line Interface no Guia do usuário do AWS CloudTrail .

A tabela a seguir lista os tipos de Deadline Cloud recursos para os quais você pode registrar eventos de dados. A coluna Tipo de evento de dados (console) mostra o valor a ser escolhido na lista Tipo de evento de dados no CloudTrail console. A coluna de valor resources.type mostra o `resources.type` valor, que você especificaria ao configurar seletores de eventos avançados usando o ou. AWS CLI CloudTrail APIs A CloudTrail coluna Dados APIs registrados em mostra as chamadas de API registradas CloudTrail para o tipo de recurso.

Tipo de evento de dados (console)	valor resources.type	Dados APIs registrados em CloudTrail
Frota com prazo	`AWS::Deadline::Fleet`	• SearchWorkers
Fila de prazos	`AWS::Deadline::Fleet`	• SearchJobs

Tipo de evento de dados (console)	valor resources.type	Dados APIs registrados em CloudTrail
Trabalhador com prazo	`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

É possível configurar seletores de eventos avançados para filtrar os campos `eventName`, `readOnly` e `resources.ARN` para registrar em log somente os eventos que são importantes para você. Para saber mais sobre esses campos, consulte AdvancedFieldSelector na Referência de API do AWS CloudTrail .

Deadline Cloud eventos de gerenciamento em CloudTrail

Os eventos de gerenciamento fornecem informações sobre as operações de gerenciamento que são realizadas nos recursos do seu Conta da AWS. Também são conhecidas como operações de ambiente de gerenciamento. Por padrão, CloudTrail registra eventos de gerenciamento.

AWS Deadline Cloud registra as seguintes operações do plano de Deadline Cloud controle CloudTrail como eventos de gerenciamento.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-leia
- assume-fleet-role-for-trabalhador
- assume-queue-role-for-leia
- assume-queue-role-for-usuário
- assume-queue-role-for-trabalhador
- criar orçamento
- criar fazenda
- create-fleet
- create-license-endpoint
- limite de criação
- criar-monitorar
- criar fila
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile

- criar trabalhador
- excluir orçamento
- delete-farm
- delete-fleet
- delete-license-endpoint
- limite de exclusão
- delete-metered-product
- excluir monitor
- fila de exclusão
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- excluir trabalhador
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- obter orçamento
- get-farm
- get-feature-map
- obtenha a frota
- get-license-endpoint
- get-limit
- get-monitor
- obter fila
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association

- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-fila
- list-available-metered-products
- orçamentos de lista
- list-farm-members
- listas de fazendas
- list-fleet-members
- listas de frotas
- list-job-members
- list-license-endpoints
- limite 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-fila
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- atualizar o orçamento
- atualize-farm
- atualizar frota
- limite de atualização

- monitor de atualização
- fila de atualização
- update-queue-environment
- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- operador de atualização

Deadline Cloud exemplos de eventos

Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a operação de API solicitada, a data e a hora da operação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, os eventos não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra um CloudTrail evento que demonstra a `CreateFarm` operação.

```
{
    "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",
}
```

O exemplo de JSON mostra o Região da AWS endereço IP e outros "`requestParameters`", como "" e "`displayName`", que podem ajudar você a identificar o evento. `kmsKeyArn`

Para obter informações sobre o conteúdo do CloudTrail registro, consulte o conteúdo do CloudTrail registro no Guia AWS CloudTrail do usuário.

Monitoramento com CloudWatch

A Amazon CloudWatch (CloudWatch) coleta dados brutos e os processa em métricas legíveis, quase em tempo real. Você pode abrir o CloudWatch console em https://console.aws.amazon.com/cloudwatch/para ver e filtrar as métricas do Deadline Cloud.

Essas estatísticas são mantidas por 15 meses para que você possa acessar informações históricas para ter uma melhor perspectiva sobre o desempenho de seu aplicativo ou serviço web. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

O Deadline Cloud tem dois tipos de registros: registros de tarefas e registros de trabalhadores. Um registro de tarefas é quando você executa registros de execução como um script ou como um DCC é executado. Um registro de tarefas pode mostrar eventos como carregamento de ativos, renderização de blocos ou texturas não encontradas.

Um registro de trabalho mostra os processos do agente de trabalho. Isso pode incluir coisas como quando os agentes de trabalho são inicializados, se registram, relatam o progresso, carregam configurações ou concluem tarefas.

O namespace para esses registros é. `/aws/deadline/*`

Para o Deadline Cloud, os trabalhadores fazem o upload desses registros para o CloudWatch Logs. Por padrão, os registros nunca expiram. Se um trabalho gerar um grande volume de dados, você poderá incorrer em custos extras. Para obter mais informações, consulte os CloudWatch preços da Amazon.

Você pode ajustar a política de retenção para cada grupo de registros. Uma retenção mais curta remove registros antigos e pode ajudar a reduzir os custos de armazenamento. Para manter os registros, você pode arquivá-los no Amazon Simple Storage Service antes de remover o registro. Para obter mais informações, consulte Exportar dados de log para o Amazon S3 usando o console no guia CloudWatch do usuário da Amazon.

> **Note**
>
> CloudWatch as leituras de registro são limitadas porAWS. Se você planeja contratar muitos artistas, sugerimos que entre em contato com o suporte AWS ao cliente e solicite um aumento na `GetLogEvents` cota. CloudWatch Além disso, recomendamos que você feche o portal de rastreamento de registros quando não estiver depurando.

Para obter mais informações, consulte CloudWatch Registrar cotas no guia do CloudWatch usuário da Amazon.

CloudWatch métricas

O Deadline Cloud envia métricas para a Amazon CloudWatch. Você pode usar o Console de gerenciamento da AWSAWS CLI, o ou uma API para listar as métricas para as quais o Deadline Cloud envia CloudWatch. Por padrão, cada ponto de dados cobre o 1 minuto após o horário de início da atividade. Para obter informações sobre como visualizar as métricas disponíveis usando o Console de gerenciamento da AWS ou oAWS CLI, consulte Visualizar métricas disponíveis no Guia do CloudWatch usuário da Amazon.

Métricas de frota gerenciadas pelo cliente

O `AWS/DeadlineCloud` namespace contém as seguintes métricas para suas frotas gerenciadas pelo cliente:

Métrica	Description	Unidade
`RecommendedFleetSize`	O número de trabalhadores que o Deadline Cloud recomenda que você use para processar trabalhos. Você pode usar essa métrica para expandir ou contrair o número de trabalhadores da sua frota.	Contagem
`UnhealthyWorkerCount`	O número de trabalhadores designados para processar trabalhos que não estão íntegros.	Contagem

Você pode usar as seguintes dimensões para refinar as métricas da frota gerenciada pelo cliente:

Dimensão	Description
FarmId	Essa dimensão filtra os dados que você solicita para a fazenda especificada.

Dimensão	Description
FleetId	Essa dimensão filtra os dados que você solicita para a frota de trabalhadores especificada.

Métricas de licenciamento

O `AWS/DeadlineCloud` namespace contém as seguintes métricas para licenciamento:

Métrica	Description	Unidade
`LicensesInUse`	O número de sessões de licença em uso.	Contagem

Você pode usar as seguintes dimensões para refinar as métricas de licenciamento:

Dimensão	Description
FleetId	Use essa dimensão para filtrar os dados para a frota gerenciada por serviços especificada. Para frotas gerenciadas pelo cliente, use a dimensão em vez disso. LicenseEndpointId
LicenseEndpointId	Use essa dimensão para filtrar os dados até o endpoint de licença especificado.
Produto	Use essa dimensão para filtrar os dados para o produto medido especificado.

Métricas de limite de recursos

O `AWS/DeadlineCloud` namespace contém as seguintes métricas para limites de recursos:

Métrica	Description	Unidade
`CurrentCount`	O número de recursos modelados por esse limite em uso.	Contagem
`MaxCount`	O número máximo de recursos modelados por esse limite. Se você definir o `maxCount` valor como -1 usando a API, o Deadline Cloud não emitirá a `MaxCount` métrica.	Contagem

Você pode usar as seguintes dimensões para refinar as métricas de limite simultâneas:

Dimensão	Description
FarmId	Essa dimensão filtra os dados que você solicita para a fazenda especificada.
LimitId	Essa dimensão filtra os dados que você solicita até o limite especificado.

Alarmes recomendados

Com CloudWatch, você pode criar alarmes que monitoram métricas e enviam uma notificação ou executam outra ação quando um limite é ultrapassado. Para obter mais informações sobre a configuração de CloudWatch alarmes, consulte o Guia CloudWatch do usuário da Amazon.

Recomendamos que você defina alarmes para as seguintes métricas do Deadline Cloud:

LicensesInUse

Dimensões: FleetId, LicenseEndpointId

Descrição do alarme: esse alarme detecta quando as sessões de licença ativas de uma frota gerenciada por serviços ou endpoint de licença estão se aproximando da cota da sua conta. Se isso ocorrer, você poderá aumentar a cota da conta para sessões de licença. Veja suas cotas atuais e solicite aumentos usando Service Quotas. Para saber mais, consulte o Guia do Usuário de Quotas de Serviço.

Intenção: evitar falhas na finalização da compra da licença monitorando o uso antes que ele atinja o limite da cota.

Estatística: máxima

Limite recomendado: 90% da sua cota de sessão de licença

Justificativa do limite: defina o limite como uma porcentagem da sua cota, para que você possa agir antes que ela atinja o limite.

Período: 1 minuto

Pontos de dados para o alarme: 1

Períodos de avaliação: 1

Operador de comparação: `GREATER_THAN_THRESHOLD`

Recursos adicionais do

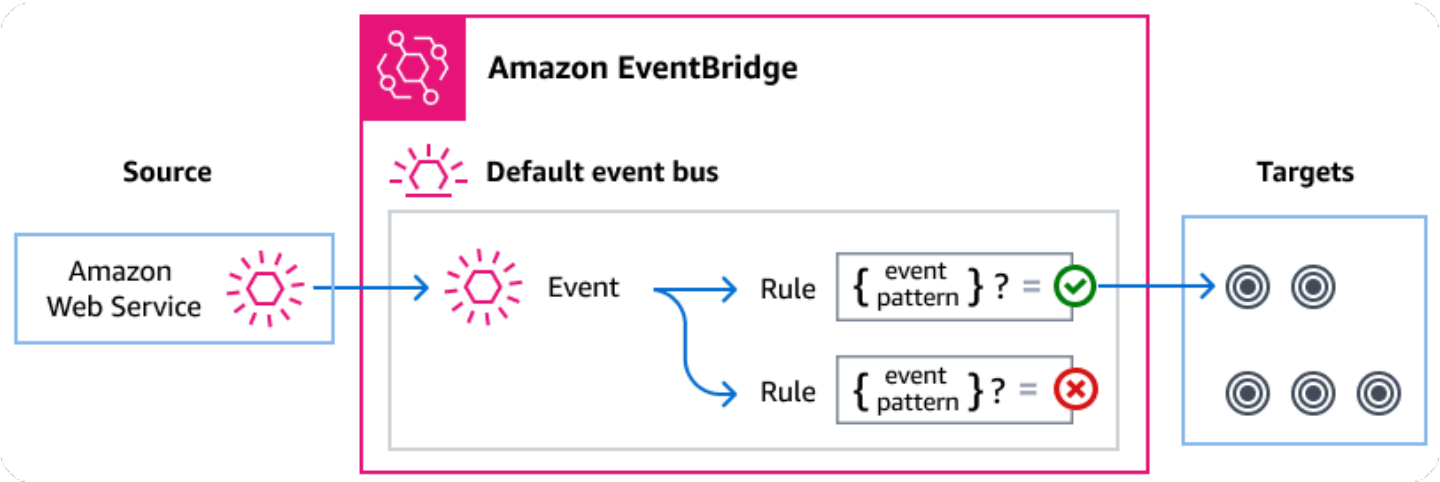
- Guia CloudWatch do usuário da Amazon
- Guia do usuário do Service Quotas

Gerenciando eventos do Deadline Cloud usando Amazon EventBridge

Amazon EventBridge é um serviço sem servidor que usa eventos para conectar componentes do aplicativo, facilitando a criação de aplicativos escaláveis orientados por eventos. A arquitetura orientada a eventos é um estilo de criação de sistemas de software com acoplamento fraco que funcionam juntos emitindo e respondendo a eventos. Os eventos representam uma mudança em um recurso ou ambiente.

Como isso funciona:

Como acontece com muitos AWS serviços, o Deadline Cloud gera e envia eventos para o barramento de eventos EventBridge padrão. (O barramento de eventos padrão é provisionado automaticamente em todas as AWS contas.) Um barramento de eventos é um roteador que recebe eventos e os entrega a zero ou mais destinos, ou alvos. As regras especificadas para o barramento de eventos avaliam os eventos à medida que eles chegam. Cada regra verifica se um evento corresponde ao padrão do evento. Se o evento corresponder, o barramento de eventos enviará o evento para os destinos especificados.



Tópicos

- Eventos do Deadline Cloud
- Entregando eventos do Deadline Cloud usando EventBridge regras
- Referência detalhada dos eventos do Deadline Cloud

Eventos do Deadline Cloud

O Deadline Cloud envia automaticamente os seguintes eventos para o barramento de EventBridge eventos padrão. Os eventos que correspondem ao padrão de eventos de uma regra são entregues aos alvos especificados com base no melhor esforço. Pode ser que os eventos sejam entregues fora da ordem.

Para obter mais informações, consulte Eventos do EventBridge, no Guia do usuário do Amazon EventBridge .

Tipo de detalhe de evento	Description
Limite orçamentário atingido	Enviado quando uma fila atinge uma porcentagem do orçamento atribuído.
Alteração do status do ciclo de vida do Job	Enviado quando há uma alteração no status do ciclo de vida de um trabalho.
Alteração do status de execução do Job	Enviado quando o status geral das tarefas em um trabalho é alterado.
Alteração do status do ciclo de vida da etapa	Enviado quando há uma alteração no status do ciclo de vida de uma etapa em um trabalho.
Alteração do status de execução da etapa	Enviado quando o status geral das tarefas em uma etapa é alterado.
Alteração do status de execução da tarefa	Enviado quando o status de uma tarefa é alterado.

Entregando eventos do Deadline Cloud usando EventBridge regras

Para que o barramento de eventos EventBridge padrão envie eventos do Deadline Cloud para um destino, você deve criar uma regra. Cada regra contém um padrão de evento, que EventBridge corresponde a cada evento recebido no barramento de eventos. Se os dados do evento corresponderem ao padrão de evento especificado, EventBridge entregará esse evento ao (s) alvo (s) da regra.

Para obter instruções abrangentes de como criar regras de barramento de eventos, consulte Criar regras que reagem a eventos no Guia do usuário do EventBridge .

Criação de padrões de eventos que correspondam aos eventos do Deadline Cloud

Cada padrão de evento é um objeto JSON que contém:

- Um atributo `source` que identifica o serviço que envia o evento. Para eventos do Deadline Cloud, a fonte é `aws.deadline`.

- (Opcional): um atributo `detail-type` que contém uma matriz dos tipos de eventos a serem correlacionados.
- (Opcional): um atributo `detail` que contém quaisquer outros dados relacionados aos eventos a serem correlacionados.

Por exemplo, o padrão de eventos a seguir corresponde a todos os eventos de alteração da recomendação de tamanho da frota para o especificado `farmId` para o Deadline Cloud:

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

Para obter mais informações sobre como escrever padrões de eventos, consulte Padrões de eventos no Guia do usuário do EventBridge .

Referência detalhada dos eventos do Deadline Cloud

Todos os eventos dos AWS serviços têm um conjunto comum de campos contendo metadados sobre o evento, como o AWS serviço que é a origem do evento, a hora em que o evento foi gerado, a conta e a região em que o evento ocorreu e outros. Para obter as definições desses campos gerais, consulte Referência da estrutura de eventos no Guia do usuário do Amazon EventBridge .

Além disso, cada evento tem um campo de `detail` que contém dados específicos desse determinado evento. A referência abaixo define os campos de detalhes dos vários eventos do Deadline Cloud.

Ao usar EventBridge para selecionar e gerenciar eventos do Deadline Cloud, é útil ter em mente o seguinte:

- O `source` campo para todos os eventos do Deadline Cloud está definido como`aws.deadline`.
- O campo do `detail-type` especifica o tipo de evento.

  Por exemplo, .`Fleet Size Recommendation Change`
- O campo de `detail` contém os dados específicos desse determinado evento.

Para obter informações sobre como criar padrões de eventos que permitem que as regras correspondam aos eventos do Deadline Cloud, consulte Padrões de eventos no Guia do Amazon EventBridge usuário.

Para obter mais informações sobre eventos e como EventBridge os processa, consulte Amazon EventBridge eventos no Guia Amazon EventBridge do usuário.

Tópicos

Evento com limite de orçamento atingido

Você pode usar o evento Limite de orçamento atingido para monitorar a porcentagem de um orçamento que foi usado. O Deadline Cloud envia eventos quando a porcentagem usada ultrapassa os seguintes limites:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

A frequência com que o Deadline Cloud envia eventos com o limite de orçamento atingido aumenta à medida que o orçamento se aproxima do limite. Essa frequência permite monitorar de perto um orçamento à medida que ele se aproxima do limite e tomar medidas para evitar gastos excessivos. Você também pode definir seus próprios limites orçamentários. O Deadline Cloud envia um evento quando o uso ultrapassa seus limites personalizados.

Se você alterar o valor de um orçamento, na próxima vez que o Deadline Cloud enviar um evento de limite de orçamento atingido, ele se baseará na porcentagem atual do orçamento que foi usado. Por exemplo, se você adicionar $50 a um orçamento de $100 que atingiu seu limite, o próximo evento Limite de orçamento atingido indicará que o orçamento está em 75 por cento.

Abaixo estão os campos de detalhes do evento `Budget Threshold Reached`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.

```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Budget Threshold Reached`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é`aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`budgetId`

O identificador do orçamento que atingiu um limite.

`thresholdInPercent`

A porcentagem do orçamento que foi usada.

Evento de alteração da recomendação de tamanho da frota

Quando você configura sua frota para usar o escalonamento automático baseado em eventos, o Deadline Cloud envia eventos que você pode usar para gerenciar suas frotas. Cada um desses eventos contém informações sobre o tamanho atual e o tamanho solicitado de uma frota. Para ver um exemplo de uso de um EventBridge evento e um exemplo de função Lambda para lidar com o evento, consulte Dimensione automaticamente sua frota do Amazon EC2 com o recurso de recomendação de escala do Deadline Cloud.

O evento de alteração da recomendação de tamanho da frota é enviado quando ocorre o seguinte:

- Quando o tamanho recomendado da frota muda e oldFleetSize é diferente de newFleetSize.
- Quando o serviço detecta que o tamanho real da frota não corresponde ao tamanho recomendado. Você pode obter o tamanho real da frota a partir do WorkerCount na resposta da GetFleet operação. Isso pode acontecer quando uma instância ativa do Amazon EC2 não consegue se registrar como funcionária do Deadline Cloud.

Abaixo estão os campos de detalhes do evento `Fleet Size Recommendation Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.

```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Fleet Size Recommendation Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é`aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`fleetId`

O identificador da frota que precisa de uma mudança de tamanho.

`oldFleetSize`

O tamanho atual da frota.

`newFleetSize`

O novo tamanho recomendado para a frota.

Evento de alteração do status do ciclo de vida do trabalho

Quando você cria ou atualiza um trabalho, o Deadline Cloud define o status do ciclo de vida para mostrar o status da ação mais recente iniciada pelo usuário.

Um evento de mudança de status do ciclo de vida do trabalho é enviado para qualquer alteração do status do ciclo de vida, inclusive quando o trabalho é criado.

Abaixo estão os campos de detalhes do evento `Job Lifecycle Status Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.

```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Job Lifecycle Status Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é`aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`queueId`

O identificador da fila que contém o trabalho.

`jobId`

O identificador do trabalho.

`previousLifecycleStatus`

O ciclo de vida indica que o trabalho está saindo. Esse campo não é incluído quando você envia um trabalho pela primeira vez.

`lifecycleStatus`

O estado do ciclo de vida em que o trabalho está entrando.

Evento de alteração do status do Job Run

Um trabalho é composto por muitas tarefas. Cada tarefa tem um status. O status de todas as tarefas é combinado para fornecer um status geral para um trabalho. Para obter mais informações, consulte Job states in Deadline Cloud no Guia do usuário do AWS Deadline Cloud.

Um evento de alteração do status de execução do trabalho é enviado quando:

- O `taskRunStatus` campo combinado muda.
- O trabalho é colocado novamente na fila, a menos que esteja no estado PRONTO.

Um evento de alteração do status de execução do trabalho NÃO é enviado quando:

- O trabalho é criado primeiro. Para monitorar a criação de empregos, monitore os eventos de alteração do status do ciclo de vida do trabalho em busca de alterações.
- O `taskRunStatusCounts` campo do trabalho muda, mas o status de execução da tarefa combinada não é alterado.

Abaixo estão os campos de detalhes do evento `Job Run Status Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.


```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Job Run Status Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é `aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`queueId`

O identificador da fila que contém o trabalho.

`jobId`

O identificador do trabalho.

`previousTaskRunStatus`

A tarefa executada indica que o trabalho está saindo.

`taskRunStatus`

O estado de execução da tarefa em que o trabalho está sendo inserido.

`taskRunStatusCounts`

O número de tarefas do trabalho em cada estado.

Evento de alteração do status do ciclo de vida da etapa

Quando você cria ou atualiza um evento, o Deadline Cloud define o status do ciclo de vida do trabalho para descrever o status da ação mais recente iniciada pelo usuário.

Um evento de mudança de status do ciclo de vida da etapa é enviado quando:

- Uma atualização de etapa é iniciada (UPDATE_IN_PROGRESS).
- Uma atualização de etapa foi concluída com êxito (UPDATE_SUCCEEDED).
- Falha na atualização de uma etapa (UPDATE_FAILED).

Um evento não é enviado quando a etapa é criada pela primeira vez. Para monitorar a criação de etapas, monitore os eventos de alteração do status do ciclo de vida do Job em busca de alterações.

Abaixo estão os campos de detalhes do evento `Step Lifecycle Status Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.

```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Step Lifecycle Status Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é`aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`queueId`

O identificador da fila que contém o trabalho.

`jobId`

O identificador do trabalho.

`stepId`

O identificador da etapa de trabalho atual.

`previousLifecycleStatus`

O ciclo de vida indica que a etapa está terminando.

`lifecycleStatus`

O estado do ciclo de vida em que a etapa está entrando.

Evento de alteração de status do Step Run

Cada etapa de um trabalho é composta por várias tarefas. Cada tarefa tem um status. Os status das tarefas são combinados para fornecer um status geral para etapas e trabalhos.

Um evento de mudança de status de execução da etapa é enviado quando:

- As `taskRunStatus` mudanças combinadas.
- A etapa é reenfileirada, a menos que esteja no estado PRONTO.

Um evento não é enviado quando:

- A etapa é criada primeiro. Para monitorar a criação de etapas, monitore os eventos de alteração do status do ciclo de vida do Job em busca de alterações.
- A etapa `taskRunStatusCounts` muda, mas o status de execução da tarefa da etapa combinada não muda.

Abaixo estão os campos de detalhes do evento `Step Run Status Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.


```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Step Run Status Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é `aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`queueId`

O identificador da fila que contém o trabalho.

`jobId`

O identificador do trabalho.

`stepId`

O identificador da etapa de trabalho atual.

`previousTaskRunStatus`

O estado de execução em que a etapa está saindo.

`taskRunStatus`

O estado de execução em que a etapa está entrando.

`taskRunStatusCounts`

O número de tarefas da etapa em cada estado.

Evento de alteração do status de execução da tarefa

O `runStatus` campo é atualizado à medida que a tarefa é executada. Um evento é enviado quando:

- O status de execução da tarefa muda.
- A tarefa é reenfileirada, a menos que esteja no estado PRONTO.

Um evento não é enviado quando:

- A tarefa é criada primeiro. Para monitorar a criação de tarefas, monitore os eventos de alteração do status do ciclo de vida do Job em busca de alterações.

Abaixo estão os campos de detalhes do evento `Task Run Status Change`.

Os `detail-type` campos `source` e estão incluídos abaixo porque contêm valores específicos para eventos do Deadline Cloud. Para obter as definições dos outros campos de metadados incluídos em todos os eventos, consulte Referência de estrutura de eventos no Guia do Amazon EventBridge usuário.

```
{
    "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 o tipo de evento.

Para esse evento, esse valor é `Fleet Size Recommendation Change`.

`source`

Identifica o serviço que gerou o evento. Para eventos do Deadline Cloud, esse valor é`aws.deadline`.

`detail`

Um objeto JSON contém informações sobre o evento. O serviço que gera o evento determina o conteúdo desse campo.

Para esse evento, esses dados incluem:

`farmId`

O identificador da fazenda que contém o trabalho.

`queueId`

O identificador da fila que contém o trabalho.

`jobId`

O identificador do trabalho.

stepId

O identificador da etapa de trabalho atual.

`taskId`

O identificador da tarefa em execução.

`previousRunStatus`

O estado de execução em que a tarefa está saindo.

`runStatus`

O status de execução que a tarefa está inserindo.

Consultando dados agregados de estatísticas de sessão usando o AWS CLI

Para monitorar custos, analisar o uso de recursos ou identificar quais usuários estão consumindo mais recursos, você pode usar o AWS Command Line Interface (AWS CLI) para consultar estatísticas de sessão agregadas para suas fazendas do AWS Deadline Cloud (Deadline Cloud). A API de estatísticas de sessão fornece dados sobre custos, tempo de execução e uso que você pode agrupar por várias dimensões, como fila, frota, tipo de instância ou usuário.

A consulta das estatísticas da sessão é um processo assíncrono. Primeiro, você inicia uma solicitação de agregação e, em seguida, recupera os resultados usando o ID de agregação.

Iniciando uma solicitação de agregação

Para iniciar uma solicitação de agregação, execute o `start-sessions-statistics-aggregation` comando. O exemplo a seguir agrupa estatísticas por ID de usuário para uma fila específica. Substitua o *`placeholder`* texto por suas informações.

```
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
```

Você pode agrupar estatísticas por outras dimensões`QUEUE_ID`, como`FLEET_ID`,`JOB_ID`,`INSTANCE_TYPE`, ou`LICENSE_PRODUCT`. Para obter mais informações sobre todos os parâmetros disponíveis, consulte start-sessions-statistics-aggregationna Referência de AWS CLI Comandos.

A resposta contém uma ID de agregação:

```
{
```

```
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Recuperando resultados

Execute o `get-sessions-statistics-aggregation` comando com o ID de agregação para recuperar os resultados. Substitua o *`placeholder`* texto por suas informações.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

O exemplo a seguir mostra uma resposta quando você agrupa estatísticas por ID de usuário. O `userId` campo contém um UUID que você deve mapear para um nome de usuário para identificar o usuário:

```
{
    "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 o nome de usuário associado a um`userId`, consultethe section called “Recuperando metadados do usuário usando UserID”.

Para mais informações sobre a API, consulte a Referência da API Deadline Cloud.

Tópicos

- the section called “Recuperando metadados do usuário usando UserID”

Recuperação de metadados e atributos do usuário usando UserID em um repositório de identidades

> **Note**
>
> Esse procedimento também se aplica ao `createdBy` campo retornado pela SearchJobsAPI, que usa o mesmo formato de ID de usuário.

O `userId` campo nas estatísticas da sessão contém um dos seguintes valores:

- Um ARN de função AWS Identity and Access Management (IAM), por exemplo:. `arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`
- Um ID de usuário do IAM Identity Center (UUID), por exemplo:. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Para a função IAM ARNs, o nome de usuário é visível no próprio ARN. Para o usuário do IAM Identity Center IDs, você pode pesquisar o nome de usuário usando a API IAM Identity Center Identity Store.

Para identificar o nome de usuário associado a uma ID de usuário do IAM Identity Center, use o procedimento a seguir. Antes de começar, obtenha o ID do Identity Store nas configurações do IAM Identity Center. Para obter mais informações, consulte the section called “Encontrando seu ID do Identity Store”.

Para mapear uma ID de usuário

1. Execute o comando a seguir, *`IdentityStoreId`* substituindo-o pelo ID `userId` do Identity Store e *`userUUID`* pela resposta das estatísticas da sessão:

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Analise a resposta, que inclui o nome de usuário:

```
{
    "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"
}
```

Encontrando seu ID do Identity Store

Para mapear usuários IDs para nomes de usuário, você precisa do ID do Identity Store. Você pode encontrar o ID do Identity Store usando o console do IAM Identity Center ou AWS CLI o.

Console

Para encontrar seu ID do Identity Store usando o console, use o procedimento a seguir.

1. Faça login no AWS Management Console e abra o console do IAM Identity Center.
2. No painel de navegação, selecione Configurações.
3. Copie o valor do ID do IAM Identity Center Identity Store. O formato é `d-xxxxxxxxxx`.

AWS CLI

Execute o comando a seguir, *`region-name`* substituindo-o pela região em que sua instância do IAM Identity Center está configurada:

```
aws sso-admin list-instances --region region-name
```

A resposta inclui`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"
        }
    ]
}
```

Verificando o mapeamento do usuário

Depois de mapear uma ID de usuário para um nome de usuário, você pode verificar no console do IAM Identity Center se a ID de usuário corresponde ao usuário esperado. Para verificar o mapeamento do usuário, use o procedimento a seguir.

1. Faça login no AWS Management Console e abra o console do IAM Identity Center.
2. No painel de navegação, escolha Users.
3. Escolha o nome de usuário na AWS CLI resposta.
4. Na seção Informações gerais, verifique se o ID do usuário corresponde às estatísticas `userId` da sua sessão.

Recursos adicionais do

- Usuários do IAM Identity Center
- Referência da API do IAM Identity Center Identity Store

Segurança em Deadline Cloud

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que é executada Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Third-party auditores testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade aplicáveis AWS Deadline Cloud, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS em Escopo por Programa .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar Deadline Cloud. Os tópicos a seguir mostram como configurar para atender Deadline Cloud aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros Serviços da AWS que o ajudem a monitorar e proteger seus Deadline Cloud recursos.

Tópicos

- Melhores práticas de segurança para o Deadline Cloud

Proteção de dados em Deadline Cloud

O AWS modelo de responsabilidade compartilhada se aplica à proteção de dados no AWS Deadline Cloud. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados . Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamento Geral sobre a Proteção de Dados (RGPD).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome.

Isso inclui quando você trabalha com Deadline Cloud ou Serviços da AWS usa o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Os dados inseridos nos campos de nome nos modelos de Deadline Cloud trabalho também podem ser incluídos nos registros de faturamento ou diagnóstico e não devem conter informações confidenciais ou sigilosas.

Tópicos

Criptografia em repouso

AWS Deadline Cloud protege dados confidenciais criptografando-os em repouso usando chaves de criptografia armazenadas em AWS Key Management Service (AWS KMS). A criptografia em repouso está disponível em todos os Regiões da AWS lugares Deadline Cloud disponíveis.

Criptografar dados significa que dados confidenciais salvos em discos não podem ser lidos por um usuário ou aplicativo sem uma chave válida. Somente uma parte com uma chave gerenciada válida pode descriptografar os dados.

Deadline Cloud exclui os volumes do Amazon Elastic Block Store quando as instâncias de trabalhadores da frota gerenciadas pelo serviço são encerradas.

Para obter informações sobre como Deadline Cloud usar AWS KMS a criptografia de dados em repouso, consulteGerenciamento de chaves.

Criptografia em trânsito

Para dados em trânsito, AWS Deadline Cloud usa o Transport Layer Security (TLS) 1.2 ou 1.3 para criptografar dados enviados entre o serviço e os trabalhadores. Exigimos TLS 1.2 e recomendamos

TLS 1.3. Além disso, se você usa uma nuvem privada virtual (VPC), você pode usá-la AWS PrivateLink para estabelecer uma conexão privada entre sua VPC e. Deadline Cloud

Gerenciamento de chaves

Ao criar uma nova fazenda, você pode escolher uma das seguintes chaves para criptografar os dados da sua fazenda:

- AWS chave KMS de propriedade — Tipo de criptografia padrão se você não especificar uma chave ao criar o farm. A chave KMS é de propriedade de AWS Deadline Cloud. Você não pode visualizar, gerenciar ou usar chaves AWS próprias. No entanto, você não precisa realizar nenhuma ação para proteger as chaves que criptografam seus dados. Para obter mais informações, consulte chaves AWS próprias no guia do AWS Key Management Service desenvolvedor.
- Chave KMS gerenciada pelo cliente — Você especifica uma chave gerenciada pelo cliente ao criar uma fazenda. Todo o conteúdo da fazenda é criptografado com a chave KMS. A chave é armazenada em sua conta e é criada, de propriedade e gerenciada por você, e AWS KMS cobranças são aplicadas. Você tem controle total sobre a chave KMS. Você pode realizar tarefas como:
  - Estabelecendo e mantendo as principais políticas
  - Estabelecer e manter subsídios e IAM policies
  - Habilitar e desabilitar políticas de chaves
  - Adicionar etiquetas
  - Criar réplicas de chaves

  Você não pode alternar manualmente uma chave de propriedade do cliente usada em uma Deadline Cloud fazenda. A rotação automática da chave é suportada.

  Para obter mais informações, consulte Chaves de propriedade do cliente no Guia do AWS Key Management Service desenvolvedor.

  Para criar uma chave gerenciada pelo cliente, siga as etapas para Criar chaves simétricas gerenciadas pelo cliente no Guia do AWS Key Management Service desenvolvedor.

Como Deadline Cloud uses AWS KMS subsídios

Deadline Cloud exige uma concessão para usar sua chave gerenciada pelo cliente. Quando você cria uma fazenda criptografada com uma chave gerenciada pelo cliente, Deadline Cloud cria uma

concessão em seu nome enviando uma `CreateGrant` solicitação AWS KMS para obter acesso à chave KMS que você especificou.

Deadline Cloud usa várias concessões. Cada concessão é usada por uma parte diferente Deadline Cloud que precisa criptografar ou descriptografar seus dados. Deadline Cloud também usa concessões para permitir o acesso a outros AWS serviços usados para armazenar dados em seu nome, como Amazon Simple Storage Service, Amazon Elastic Block Store ou OpenSearch.

Os subsídios que Deadline Cloud permitem gerenciar máquinas em uma frota gerenciada por serviços incluem um número de Deadline Cloud conta e uma função no, em `GranteePrincipal` vez de um diretor de serviço. Embora não seja típico, isso é necessário para criptografar volumes do Amazon EBS para trabalhadores em frotas gerenciadas por serviços usando a chave KMS gerenciada pelo cliente especificada para a fazenda.

Política de chave gerenciada pelo cliente

As políticas de chaves controlam o acesso à chave gerenciada pelo cliente. Cada chave deve ter exatamente uma política de chaves que contenha declarações que determinem quem pode usar a chave e como usá-la. Ao criar sua chave gerenciada pelo cliente, você pode especificar uma política de chaves. Para obter mais informações, consulte Gerenciar o acesso às chaves gerenciadas pelo cliente no Guia do desenvolvedor do AWS Key Management Service .

Política mínima de IAM para CreateFarm

Para usar sua chave gerenciada pelo cliente para criar fazendas usando o console ou a operação de `CreateFarm` API, as seguintes operações de AWS KMS API devem ser permitidas:

- `kms:CreateGrant`: adiciona uma concessão a uma chave gerenciada pelo cliente. Concede acesso ao console a uma AWS KMS chave especificada. Para obter mais informações, consulte Como usar subsídios no guia do AWS Key Management Service desenvolvedor.
- `kms:Decrypt`— Deadline Cloud Permite descriptografar dados na fazenda.
- `kms:DescribeKey`— Fornece os detalhes da chave gerenciada pelo cliente Deadline Cloud para permitir a validação da chave.
- `kms:GenerateDataKey`— Permite Deadline Cloud criptografar dados usando uma chave de dados exclusiva.

A declaração de política a seguir concede as permissões necessárias para a `CreateFarm` operação.

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 mínima de IAM para operações somente de leitura

Usar sua chave gerenciada pelo cliente para Deadline Cloud operações somente de leitura, como obter informações sobre fazendas, filas e frotas. As seguintes operações de AWS KMS API devem ser permitidas:

- `kms:Decrypt`— Deadline Cloud Permite descriptografar dados na fazenda.
- `kms:DescribeKey`— Fornece os detalhes da chave gerenciada pelo cliente Deadline Cloud para permitir a validação da chave.

A declaração de política a seguir concede as permissões necessárias para operações somente para leitura.

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 mínima de IAM para operações de leitura e gravação

Para usar sua chave gerenciada pelo cliente para Deadline Cloud operações de leitura e gravação, como criar e atualizar fazendas, filas e frotas. As seguintes operações de AWS KMS API devem ser permitidas:

- `kms:Decrypt`— Deadline Cloud Permite descriptografar dados na fazenda.
- `kms:DescribeKey`— Fornece os detalhes da chave gerenciada pelo cliente Deadline Cloud para permitir a validação da chave.
- `kms:GenerateDataKey`— Permite Deadline Cloud criptografar dados usando uma chave de dados exclusiva.

A declaração de política a seguir concede as permissões necessárias para a `CreateFarm` operação.

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"
                }
            }
        }
    ]
}
```

Monitorar suas chaves de criptografia

Ao usar uma chave gerenciada pelo AWS KMS cliente em suas Deadline Cloud fazendas, você pode usar AWS CloudTrailo Amazon CloudWatch Logs para rastrear solicitações Deadline Cloud enviadas para AWS KMS.

CloudTrail evento para bolsas

O CloudTrail evento de exemplo a seguir ocorre quando as concessões são criadas, normalmente quando você chama a `CreateFleet` operação `CreateFarmCreateMonitor`, ou.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


CloudTrail evento para decodificação

O CloudTrail evento de exemplo a seguir ocorre ao descriptografar valores usando a chave KMS gerenciada pelo 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 para criptografia

O CloudTrail evento de exemplo a seguir ocorre ao criptografar valores usando a chave KMS gerenciada pelo 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-abcdef1234567890098765432 1fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
```

```
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Excluindo uma chave KMS gerenciada pelo cliente

A exclusão de uma chave KMS gerenciada pelo cliente em AWS Key Management Service (AWS KMS) é destrutiva e potencialmente perigosa. Exclui irreversivelmente o material da chave e todos os metadados associados à chave. Depois que uma chave do KMS gerenciada pelo cliente é excluída, não é mais possível descriptografar os dados que foram criptografados com ela. Excluir a chave significa que os dados se tornam irrecuperáveis.

É por isso que AWS KMS oferece aos clientes um período de espera de até 30 dias antes de excluir a chave KMS. O período de espera padrão é de 30 dias.

Sobre o período de espera

Como é destrutivo e potencialmente perigoso excluir uma chave KMS gerenciada pelo cliente, exigimos que você defina um período de espera de 7 a 30 dias. O período de espera padrão é de 30 dias.

No entanto, o período de espera real pode ser até 24 horas a mais do que o período programado. Para obter a data e a hora reais em que a chave será excluída, use a DescribeKeyoperação. Você também pode ver a data de exclusão agendada de uma chave no console AWS KMS, na página de detalhes da chave, na seção Configuração geral. Observe o fuso horário.

Durante o período de espera, o status e o estado da chave gerenciada pelo cliente são Exclusão pendente.

- Uma chave KMS gerenciada pelo cliente que está com exclusão pendente não pode ser usada em nenhuma operação criptográfica.
- AWS KMS não gira as chaves de backup das chaves KMS gerenciadas pelo cliente que estão pendentes de exclusão.

Para obter mais informações sobre como excluir uma chave KMS gerenciada pelo cliente, consulte Excluir chaves mestras do cliente no Guia do AWS Key Management Service desenvolvedor.

Inter-network privacidade no trânsito

AWS Deadline Cloud oferece suporte à Amazon Virtual Private Cloud (Amazon VPC) para proteger conexões. A Amazon VPC fornece atributos que você pode usar para aumentar e monitorar a segurança da sua nuvem privada virtual (VPC).

Você pode configurar uma frota gerenciada pelo cliente (CMF) com instâncias do Amazon Elastic Compute Cloud (Amazon EC2) que são executadas dentro de uma VPC. Ao implantar endpoints Amazon VPC para AWS PrivateLink uso, o tráfego entre os trabalhadores em sua CMF e o endpoint permanece dentro Deadline Cloud da sua VPC. Além disso, você pode configurar sua VPC para restringir o acesso à Internet às suas instâncias.

Em frotas gerenciadas por serviços, os trabalhadores não podem ser acessados pela Internet, mas eles têm acesso à Internet e se conectam ao serviço pela Deadline Cloud Internet. Cada frota gerenciada por serviços funciona em sua própria rede isolada, e as instâncias de trabalho permanecem dedicadas a clientes individuais.

Rejeitar

AWS Deadline Cloud coleta determinadas informações operacionais para nos ajudar a desenvolver e melhorar Deadline Cloud. Os dados coletados incluem itens como seu ID de AWS conta e ID de usuário, para que possamos identificá-lo corretamente se você tiver um problema com Deadline Cloud o. Também coletamos informações Deadline Cloud específicas, como IDs de recursos (um FarmID ou QueueID quando aplicável), o nome do produto (por exemplo, JobAttachments WorkerAgent, e mais) e a versão do produto.

Você pode optar por não participar dessa coleta de dados usando a configuração do aplicativo. Cada computador que interage com Deadline Cloud, tanto as estações de trabalho do cliente quanto com os trabalhadores da frota, precisa optar por não participar separadamente.

Deadline Cloud monitor - área de trabalho

Deadline Cloud monitor - o desktop coleta informações operacionais, como quando ocorrem falhas e quando o aplicativo é aberto, para nos ajudar a saber quando você está tendo problemas com o aplicativo. Para cancelar a coleta dessas informações operacionais, acesse a página de configurações e desmarque Ativar a coleta de dados para medir o desempenho do Deadline Cloud Monitor.

Depois que você optar por não participar, o monitor do desktop não enviará mais os dados operacionais. Todos os dados coletados anteriormente são retidos e ainda podem ser usados para melhorar o serviço. Para obter mais informações, consulte Perguntas frequentes sobre a privacidade de dados da .

AWS Deadline Cloud CLI e ferramentas

A AWS Deadline Cloud CLI, os remetentes e o agente de trabalho coletam informações operacionais, como quando ocorrem falhas e quando os trabalhos são enviados, para nos ajudar a saber quando você está tendo problemas com esses aplicativos. Para optar por não coletar essas informações operacionais, use qualquer um dos seguintes métodos:

- No terminal, entre**`deadline config set telemetry.opt_out true`**.

  Isso excluirá a CLI, os remetentes e o agente de trabalho quando executados como o usuário atual.
- Ao instalar o Deadline Cloud agente de trabalho, adicione o argumento da linha de **`--telemetry-opt-out`** comando. Por exemplo, . **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**
- Antes de executar o agente de trabalho, a CLI ou o remetente, defina uma variável de ambiente: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Depois que você optar por não participar, as Deadline Cloud ferramentas não enviarão mais os dados operacionais. Todos os dados coletados anteriormente são retidos e ainda podem ser usados para melhorar o serviço. Para obter mais informações, consulte Perguntas frequentes sobre a privacidade de dados da .

Identity and Access Management na Deadline Cloud

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do Deadline Cloud. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS Identidade e acesso ao Deadline Cloud).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como o Deadline Cloud funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Identity-based exemplos de políticas para Deadline Cloud)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário-raiz

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos

usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Identity-based políticas

Identity-based políticas são documentos de políticas de permissões JSON que você anexa a uma identidade (usuário, grupo ou função). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Identity-based as políticas podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Resource-based políticas

Resource-based políticas são documentos de política JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Resource-based políticas são políticas embutidas que estão localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de Controle de Serviços (SCPs): as SCPs especificam o número máximo de permissões para uma organização ou uma unidade organizacional no AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs): definem o número máximo de permissões disponíveis para recursos em suas contas. Consulte mais informações em Resource control policies (RCPs) no Guia do usuário do AWS Organizations .
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida

quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Deadline Cloud funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Deadline Cloud, saiba quais recursos do IAM estão disponíveis para uso com o Deadline Cloud.

Recursos do IAM que você pode usar com AWS Nuvem de prazos

Recurso do IAM	Suporte Deadline Cloud
Identity-based políticas	Sim
Resource-based políticas	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Sim
Sessões de acesso direto (FAS)	Sim
Perfis de serviço	Sim
Service-linked funções	Não

Para ter uma visão de alto nível de como o Deadline Cloud e outros Serviços da AWS funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Identity-based políticas para Deadline Cloud

Compatível com políticas baseadas em identidade: sim

Identity-based políticas são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como um usuário do IAM, um grupo de usuários ou uma função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais atributos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Identity-based exemplos de políticas para Deadline Cloud

Para ver exemplos de políticas baseadas em identidade do Deadline Cloud, consulte. Identity-based exemplos de políticas para Deadline Cloud

Resource-based políticas dentro do Deadline Cloud

Compatibilidade com políticas baseadas em recursos: não

Resource-based políticas são documentos de política JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações políticas para o Deadline Cloud

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista das ações do Deadline Cloud, consulte Ações definidas pelo AWS Deadline Cloud na Referência de Autorização do Serviço.

As ações políticas no Deadline Cloud usam o seguinte prefixo antes da ação:

```
deadline
```

Para especificar várias ações em uma única declaração, separe-as com vírgulas.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Para ver exemplos de políticas baseadas em identidade do Deadline Cloud, consulte. Identity-based exemplos de políticas para Deadline Cloud

Recursos de políticas para o Deadline Cloud

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Deadline Cloud e seus ARNs, consulte Recursos definidos pelo AWS Deadline Cloud na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo AWS Deadline Cloud.

Para ver exemplos de políticas baseadas em identidade do Deadline Cloud, consulte. Identity-based exemplos de políticas para Deadline Cloud

Chaves de condição de política para o Deadline Cloud

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista das chaves de condição do Deadline Cloud, consulte Chaves de condição do AWS Deadline Cloud na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas pelo AWS Deadline Cloud.

Para ver exemplos de políticas baseadas em identidade do Deadline Cloud, consulte. Identity-based exemplos de políticas para Deadline Cloud

ACLs na Deadline Cloud

Compatível com ACLs: não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com Deadline Cloud

Compatível com ABAC (tags em políticas): sim

Attribute-based controle de acesso (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Usando credenciais temporárias com o Deadline Cloud

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Sessões de acesso direto para o Deadline Cloud

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Funções de serviço do Deadline Cloud

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber

mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper a funcionalidade do Deadline Cloud. Edite as funções de serviço somente quando o Deadline Cloud fornecer orientação para fazer isso.

Service-linked funções para o Deadline Cloud

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir a função de realizar uma ação em seu nome. Service-linked as funções aparecem no seu Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna de Service-linked função. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Identity-based exemplos de políticas para Deadline Cloud

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do Deadline Cloud. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Deadline Cloud, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição do AWS Deadline Cloud na Referência de Autorização de Serviço.

Tópicos

- Práticas recomendadas de política

- Usando o console do Deadline Cloud
- Política para acessar o console
- Política para enviar trabalhos para uma fila
- Política para permitir a criação de um endpoint de licença
- Política para permitir o monitoramento de uma fila específica da fazenda

Práticas recomendadas de política

Identity-based as políticas determinam se alguém pode criar, acessar ou excluir recursos do Deadline Cloud em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com políticas AWS gerenciadas e avance para permissões de privilégios mínimos — Para começar a conceder permissões para seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar

a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usando o console do Deadline Cloud

Para acessar o console do AWS Deadline Cloud, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Deadline Cloud em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o console do Deadline Cloud, anexe também o Deadline Cloud *ConsoleAccess* ou a política *ReadOnly* AWS gerenciada às entidades. Para saber mais, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Política para acessar o console

Para conceder acesso a todas as funcionalidades no console do Deadline Cloud, anexe essa política de identidade a um usuário ou função que você deseja ter acesso total.

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 trabalhos para uma fila

Neste exemplo, você cria uma política de escopo reduzido que concede permissão para enviar trabalhos para uma fila específica em uma fazenda 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 para permitir a criação de um endpoint de licença

Neste exemplo, você cria uma política de escopo reduzido que concede as permissões necessárias para criar e gerenciar endpoints de licença. Use essa política para criar o endpoint de licença para a VPC associada à sua fazenda.

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 para permitir o monitoramento de uma fila específica da fazenda

Neste exemplo, você cria uma política de escopo reduzido que concede permissão para monitorar trabalhos em uma fila específica para uma fazenda 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 gerenciadas para Deadline Cloud

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) for lançada ou novas operações de API forem disponibilizadas para serviços existentes.

Para saber mais, consulte AWS Políticas gerenciadas pela no Guia do usuário do IAM.

AWS política gerenciada: AWSDeadlineCloud-FleetWorker

Você pode anexar a `AWSDeadlineCloud-FleetWorker` política às suas identidades AWS Identity and Access Management (IAM).

Essa política concede aos trabalhadores dessa frota as permissões necessárias para se conectar e receber tarefas do serviço.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que os diretores gerenciem os trabalhadores em uma frota.

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-FleetWorkerde referência da AWS Managed Policy.

AWS política gerenciada: AWSDeadlineCloud-WorkerHost

É possível anexar a política `AWSDeadlineCloud-WorkerHost` às suas identidades do IAM.

Essa política concede as permissões necessárias para se conectar inicialmente ao serviço. Ele pode ser usado como um perfil de instância do Amazon Elastic Compute Cloud (Amazon EC2).

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que o usuário crie trabalhadores, assuma a função de frota para trabalhadores e aplique etiquetas aos trabalhadores

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-WorkerHostde referência da AWS Managed Policy.

AWS política gerenciada: AWSDeadlineCloud-UserAccessFarms

É possível anexar a política `AWSDeadlineCloud-UserAccessFarms` às suas identidades do IAM.

Essa política permite que os usuários acessem os dados da fazenda com base nas fazendas das quais são membros e em seu nível de associação.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que o usuário acesse os dados da fazenda.
- `ec2`— Permite que os usuários vejam detalhes sobre os tipos de instância do Amazon EC2.
- `identitystore`— Permite que os usuários vejam nomes de usuários e grupos.
- `kms`— Permite que os usuários configurem AWS Key Management Service (AWS KMS) chaves gerenciadas pelo cliente para sua instância Centro de Identidade do AWS IAM (IAM Identity Center).

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-UserAccessFarmsde referência da AWS Managed Policy.

AWS política gerenciada: AWSDeadlineCloud-UserAccessFleets

É possível anexar a política `AWSDeadlineCloud-UserAccessFleets` às suas identidades do IAM.

Essa política permite que os usuários acessem os dados da frota com base nas fazendas das quais são membros e em seu nível de associação.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que o usuário acesse os dados da fazenda.
- `ec2`— Permite que os usuários vejam detalhes sobre os tipos de instância do Amazon EC2.
- `identitystore`— Permite que os usuários vejam nomes de usuários e grupos.

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-UserAccessFleetsde referência da AWS Managed Policy.

AWS política gerenciada: AWSDeadlineCloud-UserAccessJobs

É possível anexar a política `AWSDeadlineCloud-UserAccessJobs` às suas identidades do IAM.

Essa política permite que os usuários acessem dados de trabalho com base nas fazendas das quais são membros e em seu nível de associação.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que o usuário acesse os dados da fazenda.
- `ec2`— Permite que os usuários vejam detalhes sobre os tipos de instância do Amazon EC2.
- `identitystore`— Permite que os usuários vejam nomes de usuários e grupos.

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-UserAccessJobsde referência da AWS Managed Policy.

AWS política gerenciada: AWSDeadlineCloud-UserAccessQueues

É possível anexar a política `AWSDeadlineCloud-UserAccessQueues` às suas identidades do IAM.

Essa política permite que os usuários acessem os dados da fila com base nas fazendas das quais são membros e em seu nível de associação.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `deadline`— Permite que o usuário acesse os dados da fazenda.
- `ec2`— Permite que os usuários vejam detalhes sobre os tipos de instância do Amazon EC2.
- `identitystore`— Permite que os usuários vejam nomes de usuários e grupos.

Para obter uma lista JSON dos detalhes da política, consulte o guia AWSDeadlineCloud-UserAccessQueuesde referência da AWS Managed Policy.

Atualizações do Deadline Cloud para AWS políticas gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Deadline Cloud desde que esse serviço começou a monitorar essas mudanças. Para receber alertas automáticos sobre alterações nessa página, assine o feed RSS na página de histórico de documentos do Deadline Cloud.

Alteração	Descrição	Data
AWSDeadlineCloud-UserAccessFarms: alteração	O Deadline Cloud adicionou uma nova ação `kms:Decrypt` para que você possa usar uma chave AWS KMS gerenciada pelo cliente com sua instância do IAM Identity Center.	22 de dezembro de 2025
AWSDeadlineCloud-WorkerHost: alteração	O Deadline Cloud adicionou novas ações `deadline:TagResource` `deadline:ListTagsForResource` para permitir que você adicione e visualize etiquetas associadas aos trabalhadores em sua frota.	30 de maio de 2025
AWSDeadlineCloud-UserAccessFarms: alteração AWSDeadlineCloud-UserAccessJobs: alteração AWSDeadlineCloud-UserAccessQueues: alteração	O Deadline Cloud adicionou novas ações `deadline:GetJobTemplate` `deadline:ListJobParameterDefinitions` para permitir que você reenvie trabalhos.	7 de outubro de 2024
O Deadline Cloud começou a monitorar as mudanças	O Deadline Cloud começou a monitorar as mudanças em	2 de abril de 2024

Alteração	Descrição	Data
	suas políticas AWS gerenciadas.	

Perfis de serviço

Como o Deadline Cloud usa as funções de serviço do IAM

O Deadline Cloud assume automaticamente as funções do IAM e fornece credenciais temporárias para trabalhadores, empregos e para o monitor do Deadline Cloud. Essa abordagem elimina o gerenciamento manual de credenciais e, ao mesmo tempo, mantém a segurança por meio do controle de acesso baseado em funções.

Ao criar monitores, frotas e filas, você especifica as funções do IAM que o Deadline Cloud assume em seu nome. Os trabalhadores e o monitor do Deadline Cloud então recebem credenciais temporárias dessas funções para acessar Serviços da AWS.

Função da frota

Configure uma função de frota para dar aos funcionários do Deadline Cloud as permissões de que precisam para receber trabalho e relatar o progresso desse trabalho.

Normalmente, você não precisa configurar essa função sozinho. Essa função pode ser criada para você no console do Deadline Cloud para incluir as permissões necessárias. Use o guia a seguir para entender as especificidades dessa função na solução de problemas.

Ao criar ou atualizar frotas de forma programática, especifique o ARN da função da frota usando as operações da API ou. `CreateFleet` `UpdateFleet`

O que a função da frota faz

A função de frota fornece aos trabalhadores permissões para:

- Receba novos trabalhos e relate o progresso do trabalho em andamento ao serviço Deadline Cloud
- Gerencie o ciclo de vida e o status do trabalhador
- Registre eventos de log no Amazon CloudWatch Logs para os registros do trabalhador

Configurar a política de confiança da função da frota

Sua função de frota deve confiar no serviço Deadline Cloud e ser direcionada para sua fazenda específica.

Como melhor prática, a política de confiança deve incluir condições de segurança para a proteção do Confused Deputy. Para saber mais sobre a proteção do Confused Deputy, consulte Confused Deputy no Guia do usuário do Deadline Cloud.

- `aws:SourceAccount`garante que somente recursos da mesma Conta da AWS possam assumir essa função.
- `aws:SourceArn`restringe a assunção de funções a um farm específico do 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"
        }
      }
    }
  ]
}
```

Anexe as permissões da função Fleet

Anexe a seguinte política AWS gerenciada à sua função de frota:

AWSDeadlineCloud-FleetWorker

Essa política gerenciada fornece permissões para:

- `deadline:AssumeFleetRoleForWorker`- Permite que os trabalhadores atualizem suas credenciais.
- `deadline:UpdateWorker`- Permite que os trabalhadores atualizem seu status (por exemplo, para PARADO ao sair).
- `deadline:UpdateWorkerSchedule`- Para obter trabalho e relatar o progresso.
- `deadline:BatchGetJobEntity`- Para obter informações sobre o trabalho.
- `deadline:AssumeQueueRoleForWorker`- Para acessar as credenciais da função de fila durante a execução do trabalho.

Adicione permissões KMS para fazendas criptografadas

Se sua fazenda foi criada usando uma chave KMS, adicione essas permissões à sua função de frota para garantir que o trabalhador possa acessar dados criptografados na fazenda.

As permissões do KMS só são necessárias se sua fazenda tiver uma chave KMS associada. A `kms:ViaService` condição deve usar o formato`deadline.{region}.amazonaws.com`.

Ao criar uma frota, um grupo de CloudWatch registros de registros é criado para essa frota. As permissões do trabalhador são usadas pelo serviço Deadline Cloud para criar um fluxo de registros específico para esse trabalhador em particular. Depois que o trabalhador estiver configurado e executado, ele usará essas permissões para enviar eventos de registro diretamente para o 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"
        }
      }
    }
  ]
}
```

Modificando a função da frota

As permissões para a função de frota não são personalizáveis. As permissões descritas são sempre necessárias e a adição de permissões adicionais não tem efeito.

Customer-managed função de anfitrião da frota

Configure uma WorkerHost função se você usar frotas gerenciadas pelo cliente em instâncias do Amazon EC2 ou em hosts locais.

O que a WorkerHost função faz

A WorkerHost função impulsiona os trabalhadores em hospedeiros de frotas gerenciados pelo cliente. Ele fornece as permissões mínimas necessárias para que um host:

- Crie um trabalhador no Deadline Cloud
- Assuma a função da frota para obter credenciais operacionais
- Marque trabalhadores com etiquetas de frota (se a propagação de tags estiver ativada)

Configurar permissões de WorkerHost função

Anexe a seguinte política AWS gerenciada à sua WorkerHost função:

AWSDeadlineCloud-WorkerHost

Essa política gerenciada fornece permissões para:

- `deadline:CreateWorker`- Permite que o anfitrião registre um novo trabalhador.
- `deadline:AssumeFleetRoleForWorker`- Permite que o anfitrião assuma o papel da frota.
- `deadline:TagResource`- Permite marcar trabalhadores durante a criação (se habilitado).
- `deadline:ListTagsForResource`- Permite ler as etiquetas da frota para propagação.

Entenda o processo de bootstrap

A WorkerHost função é usada somente durante a inicialização inicial do trabalhador:

1. O agente de trabalho começa no host usando WorkerHost credenciais.
2. Ele invoca `deadline:CreateWorker` o registro no Deadline Cloud.
3. Em seguida, ele invoca `deadline:AssumeFleetRoleForWorker` para obter as credenciais da função da frota.
4. Deste ponto em diante, o trabalhador usa somente as credenciais da função de frota para todas as operações.

A WorkerHost função não é usada depois que o trabalhador começa a funcionar. Essa política não é obrigatória para Service-managed frotas. Nas Service-managed frotas, o bootstrapping é realizado automaticamente.

Função da fila

A função da fila é assumida pelo trabalhador ao processar uma tarefa. Essa função fornece as permissões necessárias para concluir a tarefa.

Ao criar ou atualizar filas programaticamente, especifique o ARN da função da fila usando as operações de API ou. `CreateQueue` `UpdateQueue`

Configurar a política de confiança da função de fila

Sua função de fila deve confiar no serviço Deadline Cloud.

Como melhor prática, a política de confiança deve incluir condições de segurança para a proteção do Confused Deputy. Para saber mais sobre a proteção do Confused Deputy, consulte Confused Deputy no Guia do usuário do Deadline Cloud.

- `aws:SourceAccount`garante que somente recursos da mesma Conta da AWS possam assumir essa função.
- `aws:SourceArn`restringe a assunção de funções a um farm específico do 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}"
        }
      }
    }
```

```
  ]
}
```

Entenda as permissões de função de fila

A função de fila não usa uma única política gerenciada. Em vez disso, quando você configura sua fila no console, o Deadline Cloud cria uma política personalizada para sua fila com base na sua configuração.

Essa política criada automaticamente fornece acesso a:

Anexos de trabalho

Acesso de leitura e gravação ao seu bucket do Amazon S3 especificado para arquivos de entrada e saída do trabalho:

```
{
  "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 trabalhos

Leia o acesso aos CloudWatch registros para trabalhos nessa fila. Cada fila tem seu próprio grupo de registros e cada sessão tem seu próprio fluxo de registros:

```
{
  "Effect": "Allow",
```

```
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Third-party software

Acesso para baixar software de terceiros compatível com o Deadline Cloud (como Maya, Blender e outros):

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Adicione permissões para seus trabalhos

Adicione permissões à sua função de fila para Serviços da AWS as quais seus trabalhos precisam acessar. Ao escrever scripts de OpenJobDescription etapas, o SDK AWS CLI e usará automaticamente as credenciais da sua função de fila. Use isso para acessar os serviços adicionais necessários para concluir seu trabalho.

Exemplo de casos de uso incluem:

- para buscar dados personalizados
- Permissões de SSM para criar um túnel para um servidor de licenças personalizado
- CloudWatch para emitir métricas personalizadas

- Permissão do Deadline Cloud para criar novos trabalhos para fluxos de trabalho dinâmicos

Como as credenciais da função de fila são usadas

O Deadline Cloud fornece credenciais de função de fila para:

- Trabalhadores durante a execução do trabalho
- Usuários por meio da CLI do Deadline Cloud e do monitor ao interagir com anexos e registros de tarefas

O Deadline Cloud cria grupos de CloudWatch registros de registros separados para cada fila. Os trabalhos usam credenciais de função de fila para gravar registros no grupo de registros da fila. A CLI e o monitor do Deadline Cloud usam a função de fila (por meio de`deadline:AssumeQueueRoleForRead`) para ler registros de tarefas do grupo de registros da fila. A CLI e o monitor do Deadline Cloud usam a função de fila (por meio de`deadline:AssumeQueueRoleForUser`) para carregar ou baixar dados de anexos de tarefas.

Função do monitor

Configure uma função de monitor para dar aos aplicativos web e de desktop do Deadline Cloud acesso aos seus recursos do Deadline Cloud.

Ao criar ou atualizar monitores programaticamente, especifique o ARN da função do monitor usando as `CreateMonitor` operações de API ou. `UpdateMonitor`

O que a função de monitor faz

A função de monitor permite que o monitor do Deadline Cloud forneça aos usuários finais acesso a:

- Funcionalidade básica necessária para o Deadline Cloud Integrated Submitters, CLI e monitor
- Funcionalidade personalizada para usuários finais

Configurar a política de confiança da função de monitor

Sua função de monitor deve confiar no serviço Deadline Cloud.

Como melhor prática, a política de confiança deve incluir condições de segurança para a proteção do Confused Deputy. Para saber mais sobre a proteção do Confused Deputy, consulte Confused Deputy no Guia do usuário do Deadline Cloud.

`aws:SourceAccount`garante que somente recursos da mesma Conta da AWS possam assumir essa função.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

Anexe permissões de função de monitor

Anexe todas as políticas AWS gerenciadas a seguir à sua função de monitor para operação básica:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

Como funciona a função de monitor

Ao usar o monitor do Deadline Cloud, um usuário do serviço faz login usando Centro de Identidade do AWS IAM (IAM Identity Center) e a função de monitor é assumida. As credenciais da função assumida são usadas pelo aplicativo de monitoramento para exibir a interface do usuário do monitor, incluindo a lista de fazendas, frotas, filas e outras informações.

Ao usar o aplicativo de desktop Deadline Cloud Monitor, essas credenciais também são disponibilizadas na estação de trabalho usando um perfil de AWS credencial nomeado

correspondente ao nome do perfil fornecido pelo usuário final. Saiba mais sobre perfis nomeados no guia de referência do AWS SDK e das ferramentas.

Esse perfil nomeado é como a CLI do Deadline e os remetentes acessam os recursos do Deadline Cloud.

Personalizando a função de monitor para casos de uso avançados

Você pode personalizar a função de monitor para modificar o que os usuários podem fazer em cada nível de acesso (Visualizador, Colaborador, Gerente, Proprietário) ou para adicionar permissões para fluxos de trabalho avançados.

Personalizando as permissões de nível de acesso

As quatro políticas AWS gerenciadas anexadas à função de monitor controlam o que cada nível de acesso pode fazer. Você pode adicionar políticas personalizadas à função de monitor para conceder ou restringir permissões para níveis de acesso específicos usando a chave de `deadline:MembershipLevel` condição.

Por exemplo, para permitir que os colaboradores atualizem e cancelem trabalhos (o que normalmente é restrito a gerentes e proprietários), adicione uma política como a seguinte:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Com essa política, os colaboradores podem atualizar e cancelar trabalhos, além de enviá-los.

Adicionar permissões para fluxos de trabalho avançados

Você pode adicionar políticas personalizadas do IAM à função de monitor para conceder permissões adicionais a todos os usuários do monitor. Isso é útil para fluxos de trabalho de script avançados, nos quais os usuários precisam acessar outras funcionalidades Serviços da AWS além do padrão do Deadline Cloud.

Siga estas diretrizes ao modificar sua função de monitor:

- Não remova nenhuma das políticas gerenciadas. A remoção dessas políticas interrompe a funcionalidade do monitor.

Como o Deadline Cloud Monitor usa as credenciais da função de monitor

O Deadline Cloud Monitor obtém automaticamente as credenciais da função de monitor quando você se autentica. Esse recurso permite que o aplicativo de desktop forneça recursos aprimorados de monitoramento além do que está disponível em um navegador da Web padrão.

Quando você faz login com o monitor Deadline Cloud, ele cria automaticamente um perfil que você pode usar com a AWS CLI ou qualquer outra AWS ferramenta. Esse perfil usa as credenciais da função de monitor, oferecendo acesso programático Serviços da AWS com base nas permissões da sua função de monitor.

Os remetentes do Deadline Cloud trabalham da mesma forma: eles usam o perfil criado pelo monitor do Deadline Cloud para acessar Serviços da AWS com as permissões de função apropriadas.

Personalização avançada das funções do Deadline Cloud

Você pode estender as funções do Deadline Cloud com permissões adicionais para permitir casos de uso avançados além dos fluxos de trabalho de renderização básicos. Essa abordagem aproveita o sistema de gerenciamento de acesso do Deadline Cloud para controlar o acesso a outros Serviços da AWS com base na adesão à fila.

Colaboração em equipe com AWS CodeCommit

Adicione AWS CodeCommit permissões à sua função de fila para permitir a colaboração em equipe nos repositórios do projeto. Essa abordagem usa o sistema de gerenciamento de acesso do Deadline Cloud para casos de uso adicionais. Somente usuários com acesso à fila específica receberão essas AWS CodeCommit permissões, permitindo que você gerencie o acesso ao repositório por projeto por meio da associação à fila do Deadline Cloud.

Isso é útil para cenários em que artistas precisam acessar ativos, scripts ou arquivos de configuração específicos do projeto armazenados em AWS CodeCommit repositórios como parte de seu fluxo de trabalho de renderização.

Adicionar AWS CodeCommit permissões para função de fila

Adicione as seguintes permissões à sua função de fila para habilitar o AWS CodeCommit acesso:

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Configure o provedor de credenciais nas estações de trabalho de artistas

Configure cada estação de trabalho do artista para usar as credenciais de fila do Deadline Cloud para acesso. AWS CodeCommit Essa configuração é feita uma vez por estação de trabalho.

Para configurar o provedor de credenciais

1. Adicione um perfil de provedor de credenciais ao seu arquivo de AWS configuração ()`~/.aws/config`:

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Configure o Git para usar esse perfil para AWS CodeCommit repositórios:

   ```
   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
   ```

Substitua *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* e *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* por seus IDs reais de fazenda e fila. *REGION*Substitua pela sua AWS região (por exemplo,`us-west-2`).

Utilizar AWS CodeCommit com credenciais de fila

Depois de configuradas, as operações do Git usarão automaticamente as credenciais da função de fila ao acessar os repositórios. AWS CodeCommit O `deadline queue export-credentials` comando retorna credenciais temporárias com a seguinte aparência:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Essas credenciais são atualizadas automaticamente conforme necessário, e as operações do Git funcionarão perfeitamente:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Agora, os artistas podem acessar os repositórios do projeto usando suas permissões de fila sem precisar de credenciais separadas. AWS CodeCommit Somente usuários com acesso à fila específica poderão acessar o repositório associado, permitindo um controle de acesso refinado por meio do sistema de associação de filas do Deadline Cloud.

Solução de problemas AWS Identidade e acesso ao Deadline Cloud

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Deadline Cloud e o IAM.

Tópicos

- Não estou autorizado a realizar uma ação no Deadline Cloud
- Não estou autorizado a realizar iam: PassRole

- Quero permitir que pessoas fora da minha Conta da AWS para acessar meus recursos do Deadline Cloud

Não estou autorizado a realizar uma ação no Deadline Cloud

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `deadline:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `deadline:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você passe uma função para o Deadline Cloud.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um usuário do IAM chamado `marymajor` tenta usar o console para realizar uma ação no Deadline Cloud. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS para acessar meus recursos do Deadline Cloud

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Deadline Cloud é compatível com esses recursos, consulteComo o Deadline Cloud funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Validação de conformidade Deadline Cloud

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

Resiliência em Deadline Cloud

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura AWS global.

AWS Deadline Cloud não faz backup dos dados armazenados em seu bucket S3 de anexos de trabalho. Você pode habilitar backups dos dados dos anexos do trabalho usando qualquer mecanismo de backup padrão do Amazon S3, como controle de versão do S3 ou. AWS Backup

Segurança da infraestrutura no Deadline Cloud

Como um serviço gerenciado, AWS o Deadline Cloud é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o Deadline Cloud pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Suítes de criptografia com sigilo direto perfeito (PFS), como DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

O Deadline Cloud não oferece suporte ao uso de políticas de endpoint de nuvem privada AWS PrivateLink virtual (VPC). Ele usa a política AWS PrivateLink padrão, que concede acesso total ao endpoint. Para obter mais informações, consulte Política de endpoint padrão no guia do AWS PrivateLink usuário.

Análise de configuração e vulnerabilidade no Deadline Cloud

AWS lida com tarefas básicas de segurança, como sistema operacional (SO) convidado e aplicação de patches em bancos de dados, configuração de firewall e recuperação de desastres. Esses procedimentos foram revisados e certificados por terceiros certificados. Para obter mais detalhes, consulte os seguintes recursos da :

- Modelo de responsabilidade compartilhada
- Amazon Web Services: visão geral do processo de segurança (whitepaper)

AWS O Deadline Cloud gerencia tarefas em frotas gerenciadas por serviços ou pelo cliente:

- Para frotas gerenciadas por serviços, o Deadline Cloud gerencia o sistema operacional convidado.
- Para frotas gerenciadas pelo cliente, você é responsável por gerenciar o sistema operacional.

Para obter informações adicionais sobre configuração e análise de vulnerabilidades do AWS Deadline Cloud, consulte

- Melhores práticas de segurança para o Deadline Cloud

Cross-service prevenção delegada confusa

O problema do “confused deputy” é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executar a ação. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado. Cross-service a representação pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Recomendamos usar aws:SourceArnas chaves de contexto de condição aws:SourceAccountglobal nas políticas de recursos para limitar as permissões que AWS Deadline Cloud fornecem outro serviço ao recurso. Use `aws:SourceArn` se quiser que apenas um recurso seja associado ao acesso entre serviços. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o nome do recurso da Amazon (ARN) completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (*) para as partes desconhecidas do ARN. Por exemplo, `.arn:aws:deadline:*:123456789012:*`

Se o valor de `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar ambas as chaves de contexto de condição global para limitar as permissões.

O exemplo a seguir mostra como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto Deadline Cloud para evitar o confuso problema substituto.

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"
      }
    }
```

```
  }
}
```

Acesso AWS Deadline Cloud usando um endpoint de interface (AWS PrivateLink)

Você pode usar AWS PrivateLink para criar uma conexão privada entre sua VPC e. AWS Deadline Cloud Você pode acessar Deadline Cloud como se estivesse em sua VPC, sem o uso de um gateway de internet, dispositivo NAT, conexão VPN ou conexão. Direct Connect As instâncias na sua VPC não precisam de endereços IP públicos para acessar o Deadline Cloud.

Estabeleça essa conectividade privada criando um endpoint de interface, habilitado pelo AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Estas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Deadline Cloud.

Deadline Cloud também tem endpoints de pilha dupla disponíveis. Dual-stack os endpoints oferecem suporte a solicitações via IPv6 e IPv4.

Para saber mais, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink .

Considerações para Deadline Cloud

Antes de configurar um endpoint de interface para Deadline Cloud, consulte Acessar um serviço da AWS usando um endpoint VPC de interface no Guia.AWS PrivateLink

Deadline Cloud suporta fazer chamadas para todas as suas ações de API por meio do endpoint da interface.

Por padrão, o acesso total ao Deadline Cloud é permitido por meio do endpoint da interface. Como alternativa, você pode associar um grupo de segurança às interfaces de rede do endpoint para controlar o tráfego Deadline Cloud por meio do endpoint da interface.

Deadline Cloud também oferece suporte a políticas de endpoint de VPC. Para obter mais informações, consulte Controlar o acesso aos endpoints da VPC usando políticas de endpoint no Guia AWS PrivateLink .

Deadline Cloud endpoints

Deadline Cloud usa quatro endpoints para acessar o serviço usando AWS PrivateLink - dois para IPv4 e dois para IPv6.

Os trabalhadores usam o `scheduling.deadline.region.amazonaws.com` endpoint para obter tarefas da fila, relatar o progresso e enviar a Deadline Cloud saída da tarefa de volta. Se você estiver usando uma frota gerenciada pelo cliente, o endpoint de agendamento é o único endpoint que você precisa criar, a menos que esteja usando operações de gerenciamento. Por exemplo, se um trabalho criar mais trabalhos, você precisará habilitar o endpoint de gerenciamento para chamar a `CreateJob` operação.

O Deadline Cloud monitor usa o `management.deadline.region.amazonaws.com` para gerenciar os recursos em sua fazenda, como criar e modificar filas e frotas ou obter listas de trabalhos, etapas e tarefas.

Os AWS SDKs e a CLI adicionam automaticamente os `management` prefixos `scheduling` e ao endpoint. Se você quiser desativar esse comportamento, consulte a seção de injeção de prefixo do host no Guia de referência de AWS SDKs e ferramentas.

Deadline Cloud também requer endpoints para os seguintes endpoints AWS de serviço:

- Se você configurar sua frota gerenciada pelo cliente em uma sub-rede sem conexão com a Internet, deverá criar um VPC endpoint para o CloudWatch Amazon Logs para que os trabalhadores possam gravar registros. Para obter mais informações, consulte Monitoramento com CloudWatch.
- Se você usar anexos de trabalho, deverá criar um VPC endpoint para o Amazon Simple Storage Service (Amazon S3) para que os trabalhadores possam acessar os anexos. Para obter mais informações, consulte Job attachments in. Deadline Cloud

Crie endpoints para Deadline Cloud

Você pode criar endpoints de interface para Deadline Cloud usar o console Amazon VPC ou AWS Command Line Interface o AWS CLI(). Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário do AWS PrivateLink .

Crie endpoints de gerenciamento e agendamento para Deadline Cloud usar os seguintes nomes de serviço. *region*Substitua pelo Região da AWS local em que você implantou. Deadline Cloud

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud suporta endpoints de pilha dupla.

Se você habilitar o DNS privado para os endpoints da interface, poderá fazer solicitações de API Deadline Cloud usando o nome DNS regional padrão. Por exemplo, `scheduling.deadline.us-east-1.amazonaws.com` para operações de trabalhadores ou `management.deadline.us-east-1.amazonaws.com` para todas as outras operações.

Se sua frota gerenciada pelo cliente estiver em uma sub-rede sem conexão com a Internet, você deverá criar um endpoint de CloudWatch registros usando o seguinte nome de serviço:

```
com.amazonaws.region.logs
```

Se você usar anexos de trabalho para transferir arquivos, deverá criar um endpoint do Amazon S3 usando o seguinte nome de serviço:

```
com.amazonaws.region.s3
```

Ambientes de rede restritos

O Deadline Cloud fornece ferramentas que são usadas por artistas ou outros usuários em suas estações de trabalho locais. Essas ferramentas exigem acesso à AWS API e aos endpoints da web para realizar suas funções. Se você filtrar o acesso a AWS domínios ou endpoints de URL específicos usando uma solução de filtragem de conteúdo da Web, como firewalls de próxima geração (NGFW) ou Secure Web Gateways (SWG), deverá adicionar os seguintes domínios ou endpoints de URL às suas listas de permissões da solução de filtragem de conteúdo da web.

AWS Pontos finais da API para a lista de permissões

As ferramentas de cliente do Deadline Cloud Console de gerenciamento da AWS, como monitor, CLI e remetentes integrados, exigem acesso às AWS APIs além do Deadline Cloud. Esses endpoints oferecem suporte apenas ao 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

Domínios da Web para lista de permissões

O monitor Deadline Cloud requer acesso aos seguintes domínios para operar.

Para obter informações adicionais sobre a lista de domínios permitidos para AWS Sign-In, consulte Domínios a serem adicionados à sua lista de permissões no Guia do AWS Sign-In usuário.

- 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 endpoints para a lista de permissões

Esses domínios variam de acordo com a configuração específica do Deadline Cloud. Se forem criados monitores ou filas adicionais do Deadline Cloud, domínios adicionais precisarão ser incluídos na lista de permissões.

- *[Directory ID or alias]*.awsapps.com

Esse domínio está vinculado à configuração do IAM Identity Center e deve ser o mesmo para todas as configurações usando a mesma instância do IAM Identity Center. O valor exato pode ser encontrado pelo administrador corporativo no console do IAM Identity Center em Configurações → Portal de acesso da AWS URL.

- *[Monitor alias]*.*[Region]*.deadlinecloud.amazonaws.com

  Esse domínio é para a configuração do Monitor no Deadline Cloud. Os artistas inserem esse link no navegador ou no aplicativo de monitoramento do Deadline Cloud. Se o Deadline Cloud for configurado em contas ou regiões adicionais no futuro, esse domínio mudará. Você pode encontrar esse valor no console do Deadline Cloud em Painel → Visão geral do monitor → Detalhes do monitor → URL.

- *[Bucket name]*.*[Region]*.s3.amazonaws.com

  Esse é o domínio do bucket de anexos de trabalho usado pelas filas do Deadline Cloud. Cada fila pode ter seu próprio bucket de anexos de trabalho configurado. O nome exato do bucket pode ser encontrado no console do Deadline Cloud em Queues → Queue details → Job attachments. Para obter mais informações sobre anexos de tarefas, consulte a documentação das filas.

Melhores práticas de segurança para o Deadline Cloud

AWS O Deadline Cloud (Deadline Cloud) fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

> **Note**
>
> Para obter mais informações sobre a importância de muitos tópicos de segurança, consulte o Modelo de Responsabilidade Compartilhada.

Proteção de dados

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas individuais com AWS Identity and Access Management (IAM). Dessa maneira,

cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon Simple Storage Service (Amazon S3).
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou uma API, use um endpoint FIPS. Para obter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de formato livre, como um campo Nome. Essa recomendação inclui quando você trabalha com o AWS Deadline Cloud ou outro Serviços da AWS usando o console, a API ou AWS os SDKs. AWS CLI Todos os dados que você inserir no Deadline Cloud ou em outros serviços podem ser coletados para inclusão nos registros de diagnóstico. Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar a solicitação a esse servidor.

AWS Identity and Access Management permissões

Gerencie o acesso aos AWS recursos usando usuários, funções AWS Identity and Access Management (IAM) e concedendo o mínimo de privilégios aos usuários. Estabeleça políticas e procedimentos de gerenciamento de credenciais para criar, distribuir, alternar e revogar AWS credenciais de acesso. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Execute trabalhos como usuários e grupos

Ao usar a funcionalidade de fila no Deadline Cloud, é uma prática recomendada especificar um usuário do sistema operacional (OS) e seu grupo principal para que o usuário do sistema operacional tenha permissões de privilégio mínimo para os trabalhos da fila.

Quando você especifica “Executar como usuário” (e grupo), todos os processos para trabalhos enviados à fila serão executados usando esse usuário do sistema operacional e herdarão as permissões de sistema operacional associadas a esse usuário.

As configurações de frota e fila se combinam para estabelecer uma postura de segurança. No lado da fila, o “Job run as user” e o papel do IAM podem ser especificados para usar o sistema operacional e AWS as permissões para os trabalhos da fila. A frota define a infraestrutura (hosts de trabalho, redes, armazenamento compartilhado montado) que, quando associada a uma fila específica, executa trabalhos dentro da fila. Os dados disponíveis nos hosts de trabalho precisam ser acessados por trabalhos de uma ou mais filas associadas. Especificar um usuário ou grupo ajuda a proteger os dados nos trabalhos de outras filas, outros softwares instalados ou outros usuários com acesso aos hosts de trabalho. Quando uma fila está sem um usuário, ela é executada como o usuário agente que pode representar (`sudo`) qualquer usuário da fila. Dessa forma, uma fila sem um usuário pode escalar privilégios para outra fila.

Redes

Para evitar que o tráfego seja interceptado ou redirecionado, é essencial proteger como e para onde seu tráfego de rede é roteado.

Recomendamos que você proteja seu ambiente de rede das seguintes maneiras:

- Proteja as tabelas de rotas de sub-rede da Amazon Virtual Private Cloud (Amazon VPC) para controlar como o tráfego da camada IP é roteado.
- Se você estiver usando o Amazon Route 53 (Route 53) como provedor de DNS na configuração de sua fazenda ou estação de trabalho, proteja o acesso à API do Route 53.
- Se você se conectar ao Deadline Cloud fora dela, por AWS exemplo, usando estações de trabalho locais ou outros data centers, proteja qualquer infraestrutura de rede local. Isso inclui servidores DNS e tabelas de rotas em roteadores, switches e outros dispositivos de rede.

Vagas e dados de vagas

Os trabalhos do Deadline Cloud são executados em sessões em hosts de trabalhadores. Cada sessão executa um ou mais processos no host do trabalhador, que geralmente exigem a entrada de dados para produzir a saída.

Para proteger esses dados, você pode configurar os usuários do sistema operacional com filas. O agente de trabalho usa o usuário do sistema operacional de fila para executar subprocessos de sessão. Esses subprocessos herdam as permissões do usuário do sistema operacional de fila.

Recomendamos que você siga as melhores práticas para proteger o acesso aos dados que esses subprocessos acessam. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

Estrutura da fazenda

Você pode organizar frotas e filas do Deadline Cloud de várias maneiras. No entanto, existem implicações de segurança em certos arranjos.

Uma fazenda tem um dos limites mais seguros porque não pode compartilhar recursos do Deadline Cloud com outras fazendas, incluindo frotas, filas e perfis de armazenamento. No entanto, você pode compartilhar AWS recursos externos dentro de uma fazenda, o que compromete o limite de segurança.

Você também pode estabelecer limites de segurança entre filas dentro da mesma fazenda usando a configuração apropriada.

Siga estas práticas recomendadas para criar filas seguras na mesma fazenda:

- Associe uma frota somente a filas dentro do mesmo limite de segurança. Observe o seguinte:
  - Depois que o trabalho é executado no host do trabalhador, os dados podem permanecer atrasados, como em um diretório temporário ou no diretório inicial do usuário da fila.
  - O mesmo usuário do sistema operacional executa todos os trabalhos em um host de trabalhadores de frota de propriedade do serviço, independentemente da fila para a qual você envia o trabalho.
  - Um trabalho pode deixar processos em execução em um host de trabalho, possibilitando que trabalhos de outras filas observem outros processos em execução.
- Certifique-se de que somente filas dentro do mesmo limite de segurança compartilhem um bucket do Amazon S3 para anexos de trabalhos.
- Certifique-se de que somente filas dentro do mesmo limite de segurança compartilhem um usuário do sistema operacional.
- Proteja todos AWS os outros recursos integrados à fazenda até o limite.

Filas de anexação de trabalhos

Os anexos de trabalho estão associados a uma fila, que usa seu bucket do Amazon S3.

- Os anexos do trabalho são gravados e lidos a partir de um prefixo raiz no bucket do Amazon S3. Você especifica esse prefixo raiz na chamada da `CreateQueue` API.
- O bucket tem um correspondente`Queue Role`, que especifica a função que concede aos usuários da fila acesso ao bucket e ao prefixo raiz. Ao criar uma fila, você especifica o `Queue Role` Amazon Resource Name (ARN) junto com o bucket de anexos do trabalho e o prefixo raiz.
- As chamadas autorizadas para o`AssumeQueueRoleForRead`,`AssumeQueueRoleForUser`, e as operações `AssumeQueueRoleForWorker` da API retornam um conjunto de credenciais de segurança temporárias para o. `Queue Role`

Se você criar uma fila e reutilizar um bucket e um prefixo raiz do Amazon S3, há o risco de as informações serem divulgadas a terceiros não autorizados. Por exemplo, o QueuEA e o QueueB compartilham o mesmo bucket e o mesmo prefixo raiz. Em um fluxo de trabalho seguro, o Artista tem acesso ao QueuEA, mas não ao QueueB. No entanto, quando várias filas compartilham um intervalo, o Artista pode acessar os dados nos dados do QueueB porque usa o mesmo intervalo e prefixo raiz do QueuEA.

O console configura filas que são seguras por padrão. Certifique-se de que as filas tenham uma combinação distinta de bucket e prefixo raiz do Amazon S3, a menos que façam parte de um limite de segurança comum.

Para isolar suas filas, você deve configurar o `Queue Role` para permitir apenas o acesso da fila ao bucket e ao prefixo raiz. No exemplo a seguir, substitua cada um *`placeholder`* por suas informações específicas do 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/*"
        }
    ]
}
```

Você também deve definir uma política de confiança para a função. No exemplo a seguir, substitua o *placeholder* texto pelas informações específicas do seu 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"
                }
            }
        }
    ]
}
```

Software personalizado: buckets Amazon S3

Você pode adicionar a seguinte declaração à sua `Queue Role` para acessar o software personalizado em seu bucket do Amazon S3. No exemplo a seguir, *SOFTWARE_BUCKET_NAME* substitua pelo nome do seu bucket do S3 e *BUCKET_ACCOUNT_OWNER* pelo Conta da AWS ID que possui o 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 obter mais informações sobre as melhores práticas de segurança do Amazon S3, consulte Melhores práticas de segurança para o Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Trabalhadores anfitriões

Hospedagem segura de trabalhadores para ajudar a garantir que cada usuário só possa realizar operações para a função atribuída.

Recomendamos as seguintes melhores práticas para proteger os anfitriões dos trabalhadores:

- Usar um script de configuração do host pode alterar a segurança e as operações de um trabalhador. Uma configuração incorreta pode fazer com que o trabalhador fique instável ou pare de trabalhar. É sua responsabilidade depurar essas falhas.
- Não use o mesmo `jobRunAsUser` valor com várias filas, a menos que os trabalhos enviados a essas filas estejam dentro do mesmo limite de segurança.
- Não defina a fila `jobRunAsUser` com o nome do usuário do sistema operacional com o qual o agente de trabalho é executado.
- Conceda aos usuários da fila as permissões de sistema operacional menos privilegiadas necessárias para as cargas de trabalho de fila pretendidas. Certifique-se de que eles não tenham permissões de gravação do sistema de arquivos para arquivos de programas do agente de trabalho ou outro software compartilhado.
- Certifique-se de que somente o usuário root Linux e a conta `Administrator` proprietária estejam ativos Windows e possam modificar os arquivos do programa do agente de trabalho.

- Em hosts de Linux trabalho, considere configurar uma `umask` substituição `/etc/sudoers` que permita ao usuário do agente de trabalho iniciar processos como usuários da fila. Essa configuração ajuda a garantir que outros usuários não possam acessar arquivos gravados na fila.
- Conceda a indivíduos confiáveis acesso com menos privilégios aos anfitriões dos trabalhadores.
- Restrinja as permissões aos arquivos de configuração de substituição do DNS local (`/etc/hosts`ativados Linux e `C:\Windows\system32\etc\hosts` ativadosWindows) e às tabelas de roteamento em estações de trabalho e sistemas operacionais de host de trabalho.
- Restrinja as permissões à configuração de DNS em estações de trabalho e sistemas operacionais de host de trabalho.
- Corrija regularmente o sistema operacional e todo o software instalado. Essa abordagem inclui software usado especificamente com o Deadline Cloud, como remetentes, adaptadores, agentes de trabalho, OpenJD pacotes e outros.
- Use senhas fortes para a Windows fila. `jobRunAsUser`
- Alterne regularmente as senhas da sua fila`jobRunAsUser`.
- Garanta o menor privilégio de acesso aos segredos da Windows senha e exclua os segredos não utilizados.
- Não dê `jobRunAsUser` permissão à fila para que os comandos schedule sejam executados no futuro:
  - AtivadoLinux, negue a essas contas o acesso a `cron` `at` e.
  - AtivadoWindows, negue o acesso dessas contas ao agendador de Windows tarefas.

> **Note**
>
> Para obter mais informações sobre a importância de corrigir regularmente o sistema operacional e o software instalado, consulte o Modelo de Responsabilidade Compartilhada.

Script de configuração do host

- Usar um script de configuração do host pode alterar a segurança e as operações de um trabalhador. Uma configuração incorreta pode fazer com que o trabalhador fique instável ou pare de trabalhar. É sua responsabilidade depurar essas falhas.

Estações de trabalho

É importante proteger as estações de trabalho com acesso ao Deadline Cloud. Essa abordagem ajuda a garantir que qualquer trabalho que você enviar para o Deadline Cloud não possa executar cargas de trabalho arbitrárias cobradas de você. Conta da AWS

Recomendamos as seguintes melhores práticas para proteger as estações de trabalho de artistas. Para mais informações, consulte o .Modelo de responsabilidade compartilhada da .

- Proteja todas as credenciais persistentes que fornecem acesso ao Deadline Cloud AWS, incluindo o Deadline Cloud. Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.
- Instale somente software confiável e seguro.
- Exija que os usuários se federem com um provedor de identidade para acessar AWS com credenciais temporárias.
- Use permissões seguras nos arquivos do programa de envio do Deadline Cloud para evitar adulterações.
- Conceda a indivíduos de confiança acesso menos privilegiado às estações de trabalho de artistas.
- Use somente remetentes e adaptadores obtidos por meio do Deadline Cloud Monitor.
- Restrinja as permissões aos arquivos de configuração de substituição do DNS local (`/etc/hosts`ativados Linux e macOS `C:\Windows\system32\etc\hosts` ativadosWindows) e para rotear tabelas em estações de trabalho e sistemas operacionais de host de trabalho.
- Restrinja as permissões `/etc/resolve.conf` em estações de trabalho e sistemas operacionais hospedeiros de trabalhadores.
- Corrija regularmente o sistema operacional e todo o software instalado. Essa abordagem inclui software usado especificamente com o Deadline Cloud, como remetentes, adaptadores, agentes de trabalho, OpenJD pacotes e outros.

Verifique a autenticidade do software baixado

Verifique a autenticidade do seu software depois de baixar o instalador para se proteger contra adulteração de arquivos. Esse procedimento funciona para ambos Windows os Linux sistemas.

Windows

Para verificar a autenticidade dos arquivos baixados, conclua as etapas a seguir.

1. No comando a seguir, *`file`* substitua pelo arquivo que você deseja verificar. Por exemplo, . ***`C:\PATH\TO\MY\`*`DeadlineCloudSubmitter-windows-x64-installer.exe`** Além disso, *`signtool-sdk-version`* substitua pela versão do SignTool SDK instalada. Por exemplo, .**`10.0.22000.0`**

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Por exemplo, você pode verificar o arquivo de instalação do remetente do Deadline Cloud executando o seguinte 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 verificar a autenticidade dos arquivos baixados, use a ferramenta de linha de gpg comando.

1. Importe a OpenPGP chave executando o seguinte 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 se você deve confiar na OpenPGP chave. Alguns fatores a serem considerados ao decidir se deve confiar na chave acima incluem o seguinte:

   - A conexão com a internet que você usou para obter a chave GPG deste site é segura.
   - O dispositivo em que você está acessando este site é seguro.
   - AWS tomou medidas para garantir a hospedagem da chave OpenPGP pública neste site.

3. Se você decidir confiar na OpenPGP chave, edite a chave para confiar de gpg forma semelhante ao exemplo a seguir:

```
$ 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. Verifique o instalador de envio do Deadline Cloud

   Para verificar o instalador de envio do Deadline Cloud, conclua as seguintes etapas:

   a. Baixe o arquivo de assinatura do instalador do Deadline Cloud Submitter.

      Baixar arquivo de assinatura (.sig)

   b. Verifique a assinatura do instalador remetente do Deadline Cloud executando:

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. Verifique o monitor Deadline Cloud

   > **Note**
   >
   > Você pode verificar o download do monitor Deadline Cloud usando arquivos de assinatura ou métodos específicos da plataforma. Para métodos específicos da plataforma, consulte a Linux (Debian) guia, a guia Linux (RPM) ou a Linux (AppImage) guia com base no tipo de arquivo baixado.

   Para verificar o aplicativo de desktop Deadline Cloud Monitor com arquivos de assinatura, conclua as seguintes etapas:

a. Baixe o arquivo de assinatura correspondente para o instalador do monitor Deadline Cloud:

   - Baixe o arquivo de assinatura.deb
   - Baixe o arquivo de assinatura.rpm
   - Baixar. AppImage arquivo de assinatura

b. Verifique a assinatura:

   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 se a saída é semelhante à seguinte:

   ```
   gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

   gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
   ```

   Se a saída contiver a fraseGood `signature from "AWS Deadline Cloud"`, significa que a assinatura foi verificada com sucesso e você pode executar o script de instalação do monitor Deadline Cloud.

Chaves 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 pacotes que usam umLinux. AppImage binário, primeiro conclua as etapas 1 a 3 na Linux guia e, em seguida, conclua as etapas a seguir.

1. Na AppImageUpdate página em GitHub, baixe o validate-x86_64. AppImagearquivo.
2. Depois de baixar o arquivo, para adicionar permissões de execução, execute o comando a seguir.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```

3. Para adicionar permissões de execução, execute o comando a seguir.

```
chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

4. Para verificar a assinatura do monitor do Deadline Cloud, execute o comando a seguir.

```
./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

Se a saída contiver a fraseValidation successful, significa que a assinatura foi verificada com sucesso e você pode executar com segurança o script de instalação do monitor Deadline Cloud.

Linux (Debian)

Para verificar os pacotes que usam um Linux binário.deb, primeiro conclua as etapas 1 a 3 na guia. Linux

O dpkg é a principal ferramenta de gerenciamento de pacotes na maioria das distribuições debian baseadasLinux. Você pode verificar o arquivo.deb com a ferramenta.

1. Baixe o arquivo.deb do monitor Deadline Cloud:

   Baixe o monitor Deadline Cloud (.deb)

2. Verifique o arquivo.deb:

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. A saída será semelhante a:

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. Para verificar o arquivo.deb, confirme se ele GOODSIG está presente na saída.

Linux (RPM)

Para verificar os pacotes que usam um Linux binário.rpm, primeiro conclua as etapas 1 a 3 na Linux guia.

1. Baixe o arquivo .rpm do monitor Deadline Cloud:

Baixe o monitor Deadline Cloud (.rpm)

2. Verifique o arquivo.rpm:

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. A saída será semelhante a:

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. Para verificar o arquivo.rpm, confirme se ele `digests signatures OK` está na saída.

Histórico do documento

Para obter informações sobre as atualizações do AWS Deadline Cloud, consulte as notas de lançamento do Deadline Cloud.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.