aws

Panduan Developer

Batas Waktu Cloud

Batas Waktu Cloud: Panduan Developer

Table of Contents

Apa itu AWS Deadline Cloud?

AWS Deadline Cloud adalah AWS layanan yang dikelola sepenuhnya yang memungkinkan Anda memiliki pertanian pemrosesan yang dapat diskalakan dan berjalan dalam hitungan menit. Ini menyediakan konsol administrasi untuk mengelola pengguna, peternakan, antrian untuk pekerjaan penjadwalan, dan armada pekerja yang melakukan pemrosesan.

Panduan pengembang ini untuk pengembang pipeline, alat, dan aplikasi dalam berbagai kasus penggunaan, termasuk yang berikut ini:

- Pengembang pipa dan direktur teknis dapat mengintegrasikan Deadline Cloud APIs dan fitur ke dalam pipeline produksi khusus mereka.
- Vendor perangkat lunak independen dapat mengintegrasikan Deadline Cloud ke dalam aplikasi mereka yang memungkinkan seniman dan pengguna pembuatan konten digital untuk mengirimkan pekerjaan render Deadline Cloud dengan mulus dari workstation mereka.
- Pengembang layanan berbasis web dan cloud dapat mengintegrasikan rendering Deadline Cloud ke dalam platform mereka, memungkinkan pelanggan menyediakan aset untuk melihat produk secara virtual.

Kami menyediakan alat yang memungkinkan Anda untuk bekerja secara langsung dengan setiap langkah pipa Anda:

- Antarmuka baris perintah yang dapat Anda gunakan secara langsung atau dari skrip.
- AWS SDK untuk 11 bahasa pemrograman populer.
- Antarmuka web berbasis REST yang dapat Anda panggil dari aplikasi Anda.

Anda juga dapat menggunakan yang lain Layanan AWS di aplikasi khusus Anda. Misalnya, Anda dapat menggunakan:

- AWS CloudFormationuntuk mengotomatiskan pembuatan dan penghapusan peternakan, antrian, dan armada.
- Amazon CloudWatch mengumpulkan metrik untuk pekerjaan.
- Amazon Simple Storage Service untuk menyimpan dan mengelola aset digital dan output pekerjaan.
- AWS IAM Identity Centeruntuk mengelola pengguna dan grup untuk peternakan Anda.

Open Job Description

Deadline Cloud menggunakan spesifikasi Open Job Description (OpenJD) untuk menentukan detail pekerjaan. OpenJD dikembangkan untuk mendefinisikan pekerjaan yang portabel antara solusi. Anda menggunakannya untuk menentukan pekerjaan yang merupakan sekumpulan perintah yang berjalan pada host pekerja.

Anda dapat membuat template pekerjaan OpenJD menggunakan pengirim yang disediakan Deadline Cloud, atau Anda dapat menggunakan alat apa pun yang ingin Anda buat templat. Setelah membuat template, Anda mengirimkannya ke Deadline Cloud. Jika Anda menggunakan submitter, itu akan mengurus pengiriman template. Jika Anda membuat template dengan cara lain, Anda memanggil tindakan baris perintah Deadline Cloud, atau Anda dapat menggunakan salah satunya AWS SDKs untuk mengirim pekerjaan. Either way, Deadline Cloud menambahkan pekerjaan ke antrian yang ditentukan dan menjadwalkan pekerjaan.

Konsep dan terminologi untuk Deadline Cloud

Untuk membantu Anda memulai dengan AWS Deadline Cloud, topik ini menjelaskan beberapa konsep dan terminologi utamanya.

Sumber daya pertanian

Diagram ini menunjukkan bagaimana sumber daya pertanian Deadline Cloud bekerja sama.

Peternakan

> Sebuah peternakan berisi semua sumber daya lain yang terkait dengan mengirimkan dan menjalankan pekerjaan. Peternakan independen satu sama lain sehingga berguna untuk memisahkan lingkungan produksi.

Antrean

> Antrian memegang pekerjaan untuk penjadwalan pada armada terkait. Pengguna dapat mengirimkan pekerjaan ke antrian dan mengelola prioritas dan status mereka di dalam antrian. Antrian harus dikaitkan dengan armada dengan asosiasi antrian armada agar pekerjaannya dapat dijalankan, dan antrian dapat dikaitkan dengan beberapa armada.

Armada

Armada berisi kapasitas komputasi untuk menjalankan pekerjaan. Armada dapat dikelola layanan atau dikelola pelanggan. Armada yang dikelola layanan berjalan di Deadline Cloud dan menyertakan fungsionalitas bawaan seperti penskalaan otomatis, lisensi, dan akses perangkat lunak. Armada yang dikelola pelanggan berjalan di sumber daya komputasi Anda sendiri seperti EC2 instans Amazon atau server lokal.

Anggaran

Anggaran menetapkan ambang batas pengeluaran untuk aktivitas pekerjaan Anda dan memungkinkan Anda untuk mengambil tindakan ketika ambang batas tercapai, seperti menghentikan penjadwalan pekerjaan.

Lingkungan antrian

Lingkungan antrian mendefinisikan skrip yang berjalan pada setiap pekerja untuk mengatur atau meruntuhkan lingkungan beban kerja. Mereka berguna untuk mengatur variabel lingkungan, menginstal perangkat lunak, dan mengkonfigurasi penyimpanan aset.

Profil penyimpanan

Profil penyimpanan adalah konfigurasi untuk sekelompok host dan workstation, yang memberi tahu di mana data berada pada sistem file. Deadline Cloud menggunakan profil penyimpanan untuk memetakan jalur saat menjalankan pekerjaan pada host yang dikonfigurasi berbeda, seperti pekerjaan yang dikirimkan dari Windows dan dijalankan. Linux

Kuota

Batas memungkinkan Anda melacak penggunaan sumber daya bersama seperti lisensi mengambang dan mengontrol bagaimana mereka dialokasikan di antara pekerjaan. Batas dikaitkan dengan antrian dengan asosiasi batas antrian.

Memantau

Monitor mengkonfigurasi URL untuk aplikasi web monitor Deadline Cloud, memungkinkan pengguna akhir untuk memantau dan mengelola pekerjaan. Itu dapat diakses di browser atau melalui aplikasi desktop monitor Deadline Cloud.

Sumber daya eksekusi Job

Diagram ini menunjukkan bagaimana sumber daya pekerjaan Deadline Cloud bekerja sama.

Pekerjaan

Pekerjaan adalah serangkaian pekerjaan yang pengguna kirimkan ke Deadline Cloud untuk dijadwalkan dan dijalankan pada pekerja yang tersedia. Pekerjaan dapat membuat adegan 3D atau menjalankan simulasi. Pekerjaan dibuat dari templat pekerjaan yang dapat digunakan kembali, yang menentukan lingkungan dan proses runtime, dan parameter khusus pekerjaan. Pekerjaan berisi langkah-langkah dan tugas yang menentukan pekerjaan yang akan dilakukan, dan mereka dapat dikonfigurasi dengan prioritas, jumlah pekerja maksimum, dan pengaturan coba lagi.

Prioritas Job

Prioritas Job adalah perkiraan urutan Deadline Cloud memproses pekerjaan dalam antrian. Anda dapat menetapkan prioritas pekerjaan antara 1 dan 100, pekerjaan dengan prioritas angka yang lebih tinggi umumnya diproses terlebih dahulu. Pekerjaan dengan prioritas yang sama diproses dalam urutan yang diterima.

Properti Job

Properti Job adalah pengaturan yang Anda tentukan saat mengirimkan pekerjaan render. Beberapa contoh termasuk rentang bingkai, jalur keluaran, lampiran pekerjaan, kamera yang dapat dirender, dan banyak lagi. Properti bervariasi berdasarkan DCC tempat render dikirimkan.

Langkah

Langkah adalah bagian dari pekerjaan yang menyediakan template untuk menjalankan banyak tugas yang identik kecuali untuk nilai parameter tugas. Langkah-langkah dapat memiliki dependensi pada langkah lain, memungkinkan Anda membuat alur kerja yang kompleks dengan jalur eksekusi sekuensial atau paralel. Dalam pekerjaan rendering, langkah sering mendefinisikan perintah untuk merender bingkai dan menggunakan nomor bingkai sebagai parameter tugas.

Tugas

Tugas adalah unit kerja terkecil di Deadline Cloud. Tugas adalah bagian dari langkah-langkah dan dilaksanakan oleh pekerja, mewakili operasi individu yang perlu dilakukan sebagai bagian dari pekerjaan. Tugas dapat dikonfigurasi dengan parameter tertentu dan ditugaskan ke pekerja berdasarkan kemampuan dan ketersediaannya. Dalam rendering pekerjaan, tugas sering membuat satu frame.

Pekerja

Pekerja adalah bagian dari armada dan melaksanakan tugas dari pekerjaan. Pekerja dapat dikonfigurasi dengan kemampuan khusus seperti akselerator GPU, arsitektur CPU, dan sistem

operasi. Dalam armada yang dikelola layanan, pekerja dibuat secara otomatis saat armada keluar dan masuk.

Instans

Armada menggunakan instance untuk sumber daya CPU. Instance adalah instance EC2 kinerja Amazon. Deadline Cloud menggunakan instans On-Demand dan Spot.

Contoh Sesuai Permintaan

Instans On-Demand dihargai oleh yang kedua, tidak memiliki komitmen jangka panjang, dan tidak akan terganggu.

Contoh spot

Instans spot adalah kapasitas tanpa reservasi yang dapat Anda gunakan dengan harga diskon, tetapi dapat terganggu oleh permintaan Sesuai Permintaan.

Tunggu dan Simpan

Fitur Tunggu dan Simpan menyediakan penjadwalan pekerjaan tertunda untuk biaya lebih rendah dan dapat terganggu oleh permintaan On-Demand dan Spot. Tunggu dan Simpan hanya tersedia dalam armada yang dikelola layanan Deadline Cloud.

Tunggu dan Simpan adalah untuk mengelola eksekusi beban kerja komputasi visual di AWS Deadline Cloud. Lihat ketentuan AWS layanan untuk detailnya.

Sesi

Sesi mewakili urutan pekerjaan pekerja pada suatu pekerjaan. Selama satu sesi, seorang pekerja dapat diberikan beberapa tugas yang dijalankan satu demi satu. Sesi sering memiliki tindakan penyiapan yang mengonfigurasi lingkungan dan memuat aset sebelum menjalankan tindakan tugas.

Aksi sesi

Tindakan sesi mewakili operasi spesifik yang dilakukan selama sesi seperti menyiapkan lingkungan, menjalankan tugas, dan menyinkronkan aset.

Konsep dan terminologi penting lainnya

Penjelajah penggunaan

Penjelajah penggunaan adalah fitur monitor Deadline Cloud. Ini memberikan perkiraan perkiraan biaya dan penggunaan Anda.

Manajer anggaran

Manajer anggaran adalah bagian dari monitor Deadline Cloud. Gunakan manajer anggaran untuk membuat dan mengelola anggaran. Anda juga dapat menggunakannya untuk membatasi aktivitas agar tetap sesuai anggaran.

Pustaka klien Cloud batas waktu

Pustaka klien open-source mencakup antarmuka baris perintah dan pustaka untuk mengelola Deadline Cloud. Fungsionalitas termasuk mengirimkan bundel pekerjaan berdasarkan spesifikasi Open Job Description ke Deadline Cloud, mengunduh output lampiran pekerjaan, dan memantau pertanian Anda menggunakan antarmuka baris perintah (CLI).

Aplikasi pembuatan konten digital (DCC)

Aplikasi pembuatan konten digital (DCCs) adalah produk pihak ketiga tempat Anda membuat konten digital. Deadline Cloud memiliki integrasi bawaan dengan banyak DCCs seperti Autodesk Maya, Blender, dan Maxon Cinema 4D yang memungkinkan Anda mengirimkan pekerjaan dari dalam DCC dan merender armada yang dikelola layanan dengan perangkat lunak dan lisensi yang telah dikonfigurasi sebelumnya.

Lampiran Job

Lampiran Job adalah fitur Deadline Cloud yang Anda unggah dan unduh aset sebagai bagian dari pekerjaan seperti tekstur, model 3D, dan rig pencahayaan. Lampiran Job disimpan di Amazon S3 dan menghindari kebutuhan akan penyimpanan jaringan bersama.

Templat Job

Template pekerjaan mendefinisikan lingkungan runtime dan semua proses yang berjalan sebagai bagian dari pekerjaan Deadline Cloud.

Batas waktu pengirim Cloud

Submitter Deadline Cloud adalah plugin untuk DCC yang memungkinkan pengguna untuk dengan mudah mengirimkan pekerjaan dari dalam DCC.

Titik akhir lisensi

Titik akhir lisensi membuat lisensi berbasis penggunaan Deadline Cloud untuk produk pihak ketiga tersedia di dalam VPC Anda. Model ini dibayar saat Anda pergi, dan Anda dikenakan biaya untuk jumlah jam dan menit yang Anda gunakan. Titik akhir lisensi tidak terhubung ke peternakan dan dapat digunakan secara independen.

Tanda

Tag adalah label yang dapat Anda tetapkan ke AWS sumber daya. Setiap tag terdiri dari kunci dan nilai opsional yang Anda tentukan. Dengan tag, Anda dapat mengkategorikan AWS sumber daya Anda dengan berbagai cara, seperti berdasarkan tujuan, pemilik, atau lingkungan.

Lisensi berbasis penggunaan (UBL)

Lisensi berbasis penggunaan (UBL) adalah model lisensi berdasarkan permintaan yang tersedia untuk produk pihak ketiga tertentu. Model ini dibayar sesuai keinginan Anda, dan Anda dikenakan biaya untuk jumlah jam dan menit yang Anda gunakan.

Batas Waktu Panduan Arsitektur Cloud

Topik ini memberikan panduan dan praktik terbaik untuk merancang dan membangun render farm yang andal, aman, efisien, dan hemat biaya untuk beban kerja Anda menggunakan Deadline Cloud. Menggunakan panduan ini dapat membantu Anda membangun beban kerja yang stabil dan efisien, memungkinkan Anda untuk fokus pada inovasi, mengurangi biaya, dan meningkatkan pengalaman pelanggan Anda.

Konten ini ditujukan untuk kepala petugas teknologi (CTOs), arsitek, pengembang, dan anggota tim operasi.

Alur kerja end-to-end rendering memerlukan solusi di beberapa lapisan proses seperti pembuatan pekerjaan, akses aset, dan pemantauan pekerjaan. Deadline Cloud menawarkan beberapa solusi untuk setiap lapisan proses rendering. Dengan memilih dari opsi Deadline Cloud di setiap lapisan, Anda dapat mendesain alur kerja yang sesuai dengan kasus penggunaan Anda.

Untuk setiap lapisan, Anda harus memutuskan pendekatan mana yang terbaik untuk kasus penggunaan Anda. Ini bukan definisi skenario yang ketat dan bukan satu-satunya cara untuk menggunakan Deadline Cloud. Sebaliknya, ini adalah serangkaian konsep tingkat tinggi untuk membantu Anda memahami bagaimana Deadline Cloud mungkin dapat masuk ke dalam bisnis atau alur kerja Anda. Anda dapat memisahkan beban kerja Deadline Cloud ke dalam lapisan berikut: Job Source, Job Submission, Application Management, Application Licensing, Asset Access, Output Management, dan Worker Infrastructure Management.

Secara umum, Anda dapat skenario mix-and-match apa pun dalam satu lapisan dengan skenario lain di lapisan lain, kecuali kombinasi tertentu yang ditentukan di bawah ini.


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

Sumber Job

Sumber pekerjaan adalah titik akses di mana pekerjaan baru akan memasuki sistem yang akan diberikan oleh Deadline Cloud. Pada tingkat tinggi, ada dua sumber utama pekerjaan: interaktivitas manusia dan sistem komputer otomatis.

Alur kerja interaktif

Dalam skenario ini, seorang seniman atau peran kreatif lainnya adalah generator utama pekerjaan yang akan diproses di pertanian Deadline Cloud. Biasanya output dari pekerjaan ini adalah artefak utama untuk proyek atau tim yang lebih besar. Mereka melakukan pekerjaan mereka menggunakan perangkat lunak seperti alat pembuatan konten digital standar industri (DCC). Mereka secara manual mengirimkan pekerjaan ke pertanian Deadline Cloud dan melihat output setelahnya untuk ditinjau. Workstation itu sendiri tidak dikelola oleh AWS.

Dalam kebanyakan kasus, artis ini menggunakan submitter terintegrasi Deadline Cloud dan monitor Deadline Cloud di lapisan Aplikasi dan Pemantauan Beban Kerja.

Alur kerja otomatis

Dalam skenario ini, sistem program yang dimiliki oleh pelanggan adalah generator utama pekerjaan di pertanian Deadline Cloud. Ini bisa berupa pembuatan aset dalam pipa ritel, seperti video meja putar yang dihasilkan dari model 3D atau pemindaian. Ini bisa berupa pengomposisian otomatis grafik siaran dan kartu pemain untuk olahraga. Tema skenario ini adalah individu tidak secara manual mengirimkan setiap pekerjaan ke Deadline Cloud, melainkan pekerjaan dihasilkan sebagai bagian dari sistem yang lebih besar.

Dengan pekerjaan otomatis, pengirim terintegrasi Deadline Cloud kurang umum dan monitor Deadline Cloud untuk digunakan. Seringkali definisi pekerjaan akan berupa pengembangan aplikasi khusus yang ditulis oleh Anda dan output pekerjaan akan secara otomatis mengalir ke sistem Digital Asset Management (DAM) atau sistem Media Asset Management (MAM) untuk persetujuan dan distribusi.

Pengajuan Job

Pekerjaan dikirimkan ke Deadline Cloud menggunakan OpenJobDescriptiontemplate. OpenJobDescription adalah spesifikasi terbuka yang fleksibel untuk mendefinisikan pekerjaan

pemrosesan batch yang portabel antara penerapan sistem penjadwalan yang berbeda. File definisi Job menjelaskan parameter pekerjaan, langkah-langkah pekerjaan, bagaimana langkah diparameterisasi berdasarkan input pekerjaan, serta skrip aktual yang akan berjalan pada Pekerja untuk melakukan pemrosesan. Gagasan Pengajuan Beban Kerja adalah bagaimana definisi pekerjaan ini dibuat, siapa yang menciptakannya dan bagaimana mereka dikirimkan.

Submitter terintegrasi dengan DCC

Submitter terintegrasi Deadline Cloud adalah perangkat lunak yang mengikat Deadline Cloud dengan DCC standar industri atau paket perangkat lunak. Pengirim terintegrasi menentukan cara mengubah data dan konfigurasi untuk render, komposit, atau beban kerja lainnya menjadi templat pekerjaan, sesuatu yang dapat dipahami oleh Deadline Cloud. Banyak submitter terintegrasi dibuat dan dikelola oleh tim Deadline Cloud atau pencipta paket perangkat lunak, tetapi jika belum ada untuk aplikasi yang diinginkan maka Anda dapat membuat dan memelihara submitter Anda sendiri. Ada satu set terbatas DCCs yang didukung oleh tim Deadline Cloud.

Alur kerja interaktif biasanya melibatkan pengirim terintegrasi, tetapi tidak selalu. Untuk alur kerja templat dan otomatis, alur kerja umum adalah bagi seorang seniman untuk menyiapkan pekerjaan templat di DCC mereka dan melakukan ekspor satu kali bundel pekerjaan. Paket pekerjaan ini mendefinisikan cara menjalankan jenis pekerjaan tertentu di Deadline Cloud dengan cara parameter. Paket pekerjaan ini dapat diintegrasikan ke dalam skenario Alur Kerja Otomatis untuk tujuan otomatisasi.

Definisi pekerjaan khusus

Untuk aplikasi dan alur kerja khusus, dimungkinkan untuk sepenuhnya mengontrol bagaimana definisi pekerjaan ini dibuat dan dikirimkan ke Deadline Cloud. Misalnya, situs e-commerce mungkin meminta penjual untuk mengunggah model 3D dari objek yang mereka jual. Setelah unggahan ini, platform e-commerce dapat secara dinamis menghasilkan definisi pekerjaan untuk dikirimkan ke Deadline Cloud untuk secara otomatis menghasilkan animasi turntable pada latar belakang umum menggunakan pencahayaan umum untuk mencocokkan objek 3D lainnya yang tersedia di situs. Selama pengembangan platform e-commerce, pengembang perangkat lunak akan membuat definisi pekerjaan, menanamkannya ke platform e-commerce dengan parameter yang akhirnya disediakan oleh penjual, dan kode platform untuk mengirimkan pekerjaan ini selama alur kerja unggahan produk platform.

Deadline Cloud menyediakan sejumlah contoh definisi pekerjaan dalam repositori sampel di github.

Manajemen aplikasi

Setelah pekerjaan dikirimkan ke Deadline Cloud dan ditugaskan ke pekerja, skrip dari definisi pekerjaan dijalankan pada pekerja. Dalam kebanyakan kasus, skrip ini akan memanggil aplikasi untuk melakukan pemrosesan yang sebenarnya, seperti perender, komposit, encode, penyaringan, atau lainnya dari sejumlah tugas intensif komputasi. Manajemen aplikasi adalah konsep untuk memastikan versi yang diperlukan dari perangkat lunak yang diperlukan tersedia untuk pekerja.

Anda dapat mengelola aplikasi menggunakan sistem manajemen paket apa pun yang Anda suka, tetapi Deadline Cloud menyediakan sejumlah alat untuk dengan mudah mengaktifkan penggunaan paket conda. Conda adalah open-source, cross-platform, manajer paket agnostik bahasa dan sistem manajemen lingkungan.

Batas waktu Saluran conda yang dikelola Cloud untuk armada yang dikelola layanan (SMF)

Saat menggunakan armada yang dikelola layanan, saluran conda yang dikelola Cloud Deadline secara otomatis disiapkan dan dikonfigurasi untuk digunakan oleh pekerjaan Anda. Layanan Deadline Cloud menyediakan sejumlah aplikasi dan render DCC mitra di saluran conda ini. Untuk informasi selengkapnya, lihat Membuat lingkungan antrian di panduan pengguna Deadline Cloud. Paket-paket ini secara otomatis tetap up to date oleh layanan Deadline Cloud dan tidak memerlukan pemeliharaan dari Anda. Saluran conda ini hanya tersedia saat menggunakan armada yang dikelola layanan dan tidak tersedia saat menggunakan armada yang dikelola pelanggan.

Saluran conda yang dikelola sendiri

Jika Anda tidak dapat menggunakan saluran conda yang dikelola Cloud Deadline, Anda harus menentukan cara menginstal, menambal, dan mengelola aplikasi di armada Deadline Cloud Anda. Salah satu opsi adalah membuat saluran conda yang Anda atur dan pertahankan. Ini akan paling erat beroperasi dengan saluran conda yang dikelola Cloud Deadline. Misalnya, Anda dapat menggunakan DCC dari saluran conda yang dikelola Cloud Deadline tetapi membawa paket Anda sendiri yang berisi plugin DCC tertentu. Untuk informasi selengkapnya tentang proses ini, lihat Membuat saluran conda menggunakan S3.

Manajemen aplikasi khusus

Untuk manajemen aplikasi, persyaratan dari Deadline Cloud adalah bahwa aplikasi tersedia di PATH ketika skrip pekerjaan dijalankan pada pekerja.

Jika Anda sudah membangun dan memelihara paket Rez, Anda dapat menggunakan lingkungan antrian untuk menginstal aplikasi dari repositori Rez. Contoh lingkungan antrian dapat ditemukan di AWS Deadline Cloud GitHub org.

Jika Anda sudah mengelola aplikasi pada armada yang dikelola pelanggan dengan pekerja berumur panjang atau dalam gambar sistem, maka tidak diperlukan lingkungan antrian untuk manajemen aplikasi. Pastikan aplikasi muncul di jalur pengguna pekerjaan dan kirimkan pekerjaan.

Lisensi aplikasi

Banyak beban kerja yang biasanya dijalankan di Deadline Cloud memerlukan lisensi perangkat lunak dari vendor perangkat lunak. Aplikasi ini sering dilisensikan per kursi, per-CPU atau per-host. Anda bertanggung jawab untuk memastikan bahwa penggunaan perangkat lunak pihak ketiga di Deadline Cloud mematuhi perjanjian lisensi pihak ke-3. Jika Anda menggunakan perangkat lunak sumber terbuka, perangkat lunak khusus, atau perangkat lunak bebas lisensi, maka konfigurasi lapisan ini tidak diperlukan. Perlu diingat bahwa Deadline Cloud hanya mendukung lisensi render dan tidak mendukung lisensi workstation.

Armada yang dikelola layanan dan lisensi berbasis penggunaan

Saat menggunakan armada yang dikelola layanan Deadline Cloud, lisensi berbasis penggunaan (UBL) secara otomatis dikonfigurasi untuk perangkat lunak yang didukung. Pekerjaan yang dijalankan pada armada yang dikelola layanan secara otomatis memiliki variabel lingkungan yang disetel untuk aplikasi yang didukung untuk mengarahkan mereka menggunakan server lisensi Deadline Cloud. Saat menggunakan Deadline Cloud UBL, Anda hanya dikenakan biaya untuk jumlah jam Anda menggunakan aplikasi berlisensi.

Armada yang dikelola pelanggan dan lisensi berbasis penggunaan

Deadline Cloud usage-based licensing (UBL) juga tersedia saat tidak menggunakan armada yang dikelola layanan. Dalam skenario ini, Anda akan mengatur titik akhir lisensi Deadline Cloud yang menyediakan alamat IP di subnet VPC pilihan Anda yang menyediakan akses ke server lisensi Deadline Cloud. Setelah Anda mengonfigurasi variabel lingkungan khusus perangkat lunak yang sesuai pada pekerja Anda dan mengonfigurasi konektivitas jaringan dari pekerja ke alamat IP titik akhir lisensi tersebut, pekerja dapat memeriksa dan memeriksa lisensi untuk perangkat lunak yang didukung. Anda dikenakan biaya per jam untuk lisensi yang sama seperti saat menggunakan UBL dalam armada yang dikelola layanan.

Lisensi khusus

Anda dapat menggunakan aplikasi yang tidak didukung oleh Deadline Cloud UBL atau Anda mungkin memiliki lisensi yang sudah ada sebelumnya yang masih valid. Dalam skenario ini, Anda bertanggung jawab untuk mengonfigurasi jalur jaringan dari pekerja Anda (dikelola pelanggan atau layanan) ke server lisensi. Untuk informasi selengkapnya tentang lisensi khusus, lihatConnect armada yang dikelola layanan ke server lisensi kustom.

Akses aset

Setelah pekerjaan diserahkan ke pekerja dan aplikasi dikonfigurasi, pekerja harus dikonfigurasi untuk mengakses data aset yang diperlukan untuk pekerjaan itu. Ini bisa berupa data 3D, data tekstur, data animasi, bingkai video atau jenis data lain yang digunakan dalam pekerjaan Anda.

Mulailah dengan memikirkan di mana data Anda saat ini disimpan. Ini mungkin ada di hard drive workstation, alat kolaborasi pengguna, kontrol sumber, sistem file bersama lokal atau di cloud, Amazon S3, atau sejumlah lokasi lainnya.

Selanjutnya, pertimbangkan apa yang diperlukan bagi pekerja untuk mengakses data ini. Apakah data ini hanya tersedia di jaringan perusahaan Anda? Identitas atau kredensional apa yang diperlukan untuk mengakses data? Apakah sumber data diskalakan untuk mendukung pekerjaan dengan jumlah pekerja yang Anda harapkan untuk memproses pekerjaan?

Lampiran Job

Cara termudah untuk memulai dengan mekanisme akses aset adalah lampiran pekerjaan Deadline Cloud. Saat pekerjaan dikirimkan menggunakan lampiran pekerjaan, data yang diperlukan oleh pekerjaan akan diunggah ke bucket Amazon S3 bersama dengan file manifes yang menentukan file mana yang dibutuhkan pekerjaan tersebut. Dengan lampiran pekerjaan, tidak diperlukan pengaturan jaringan atau penyimpanan bersama yang rumit. File hanya diunggah sekali, jadi unggahan berikutnya selesai lebih cepat. Setelah pekerja selesai memproses pekerjaan, data output diunggah ke Amazon S3 sehingga dapat diunduh oleh artis atau klien lain. Skala lampiran pekerjaan untuk armada dengan berbagai ukuran dan sederhana dan cepat untuk digunakan dan digunakan.

Lampiran Job bukanlah alat terbaik untuk semua situasi. Jika data Anda sudah aktif AWS, maka lampiran pekerjaan menambahkan salinan tambahan data Anda, termasuk waktu transfer terkait dan biaya penyimpanan. Lampiran Job mengharuskan pekerjaan dapat sepenuhnya menentukan data yang dibutuhkan pada saat pengiriman, sehingga data dapat diunggah.

Untuk menggunakan lampiran pekerjaan, antrean Deadline Cloud Anda harus memiliki bucket lampiran pekerjaan terkait dan peran antrian harus digunakan untuk menyediakan akses ke bucket tersebut. Secara default, Deadline Cloud terintegrasi pengirim semua mendukung lampiran pekerjaan. Jika Anda tidak menggunakan submitter terintegrasi Deadline Cloud, lampiran pekerjaan dapat digunakan dengan perangkat lunak khusus Anda dengan mengintegrasikan pustaka Python Deadline Cloud.

Akses penyimpanan khusus

Jika Anda tidak menggunakan lampiran pekerjaan, Anda bertanggung jawab untuk memastikan pekerja memiliki akses ke data yang diperlukan untuk pekerjaan. Deadline Cloud menyediakan sejumlah alat untuk mendukung hal ini dan untuk menjaga pekerjaan tetap portabel. Anda mungkin ingin menggunakan solusi penyimpanan khusus ketika Anda sudah memiliki penyimpanan jaringan bersama untuk artis dan pekerja, Anda lebih suka menggunakan layanan eksternal seperti LucidLink, atau alasan lain.

Gunakan profil penyimpanan untuk memodelkan sistem file di workstation dan host pekerja Anda. Setiap profil penyimpanan menjelaskan sistem operasi dan tata letak sistem file dari salah satu konfigurasi sistem Anda. Menggunakan profil penyimpanan, ketika artis yang menggunakan workstation windows mengirimkan pekerjaan yang diproses oleh Linux pekerja, Deadline Cloud memastikan pemetaan jalur terjadi sehingga pekerja dapat mengakses penyimpanan data yang telah Anda konfigurasi.

Saat menggunakan armada yang dikelola layanan Deadline Cloud, skrip konfigurasi host, dan titik akhir sumber daya VPC memungkinkan pekerja untuk secara langsung memasang dan mengakses penyimpanan bersama atau layanan lain yang tersedia di VPC Anda.

Pemantauan pekerjaan dan manajemen output

Setelah pekerjaan yang dikirim ke Deadline Cloud berhasil diselesaikan, seseorang atau proses akan mengunduh output pekerjaan untuk digunakan dalam alur kerja bisnis di luar Deadline Cloud. Setelah kegagalan pekerjaan, log pekerjaan dan informasi pemantauan membantu mendiagnosis masalah.

Batas waktu Monitor Cloud

Aplikasi monitor Deadline Cloud tersedia di web dan untuk desktop. Solusi ini paling cocok untuk studio yang menggunakan alur kerja interaktif untuk berbagai DCCs penggunaan lampiran pekerjaan untuk penyimpanan. Monitor hanya mendukung Anda saat menggunakan IAM Identity Center.

IAM Identity Center adalah produk Identitas Tenaga Kerja, bukan solusi identitas konsumen (B2C) sehingga tidak sesuai untuk banyak skenario B2C.

Aplikasi monitor kustom

Jika Anda ingin menyesuaikan pengalaman pemantauan pengguna Anda, Anda sedang membangun produk B2C, atau membangun sistem yang sangat khusus menggunakan Deadline Cloud, Anda memilih untuk membuat aplikasi pemantauan khusus. Anda dapat menggunakan AWS Deadline Cloud API untuk membuat aplikasi kustom ini, menggabungkan konteks alur kerja Anda secara keseluruhan dengan konsep Deadline Cloud. Misalnya, produk B2C Anda mungkin memiliki konsep proyeknya sendiri yang disiapkan pengguna dan aplikasi Anda dapat membuat pekerjaan Deadline Cloud di antarmuka yang sama.

Solusi pemantauan otomatis

Dalam beberapa skenario, tidak diperlukan aplikasi pemantauan khusus untuk Deadline Cloud. Skenario ini umum terjadi pada alur kerja otomatis di mana Deadline Cloud digunakan untuk secara otomatis merender aset dalam pipeline, seperti grafik siaran untuk olahraga atau berita. Dalam skenario ini, Deadline Cloud API dan EventBridge peristiwa digunakan untuk mengintegrasikan dengan sistem Manajemen Aset Media eksternal untuk persetujuan dan memindahkan data ke langkah berikutnya dalam proses.

Manajemen infrastruktur pekerja

Armada Deadline Cloud adalah pengelompokan server (pekerja) yang dapat memproses pekerjaan yang dikirimkan ke antrian Deadline Cloud dan merupakan infrastruktur inti dari setiap Deadline Cloud farm.

Armada yang dikelola layanan

Dalam armada yang dikelola layanan, Deadline Cloud bertanggung jawab atas host pekerja, sistem operasi, jaringan, patching, autoscaling, dan faktor lain dalam menjalankan render farm. Anda menentukan jumlah minimum dan maksimum pekerja yang Anda inginkan, bersama dengan spesifikasi sistem yang diperlukan untuk aplikasi Anda dan Deadline Cloud melakukan sisanya. Armada yang dikelola layanan adalah satu-satunya opsi armada yang dapat menggunakan saluran conda yang dikelola Cloud Deadline untuk mengelola aplikasi DCC industri dengan mudah. Selain itu, Deadline Cloud UBL secara otomatis dikonfigurasi dengan armada yang dikelola layanan.

Tunggu dan Simpan armada untuk biaya lebih rendah, beban kerja yang tahan penundaan hanya tersedia menggunakan armada yang dikelola layanan.

Armada yang dikelola pelanggan

Anda menggunakan armada yang dikelola pelanggan saat Anda membutuhkan kontrol lebih besar atas host pekerja dan lingkungannya. Armada yang dikelola pelanggan paling cocok saat menggunakan Deadline Cloud lokal. Untuk mempelajari selengkapnya, lihat Membuat dan menggunakan armada yang dikelola pelanggan Deadline Cloud.

Contoh arsitektur

Studio produksi tradisional

Studio produksi tradisional membutuhkan komputasi, penyimpanan, dan infrastruktur jaringan yang signifikan yang dapat menjangkau beberapa lokasi fisik untuk melayani beban kerja rendering. Setiap paket perangkat lunak individu dan vendor memiliki perangkat keras, perangkat lunak, jaringan, dan persyaratan lisensi yang unik yang harus dipenuhi saat menyelesaikan konflik versi, kompatibilitas, dan sumber daya.

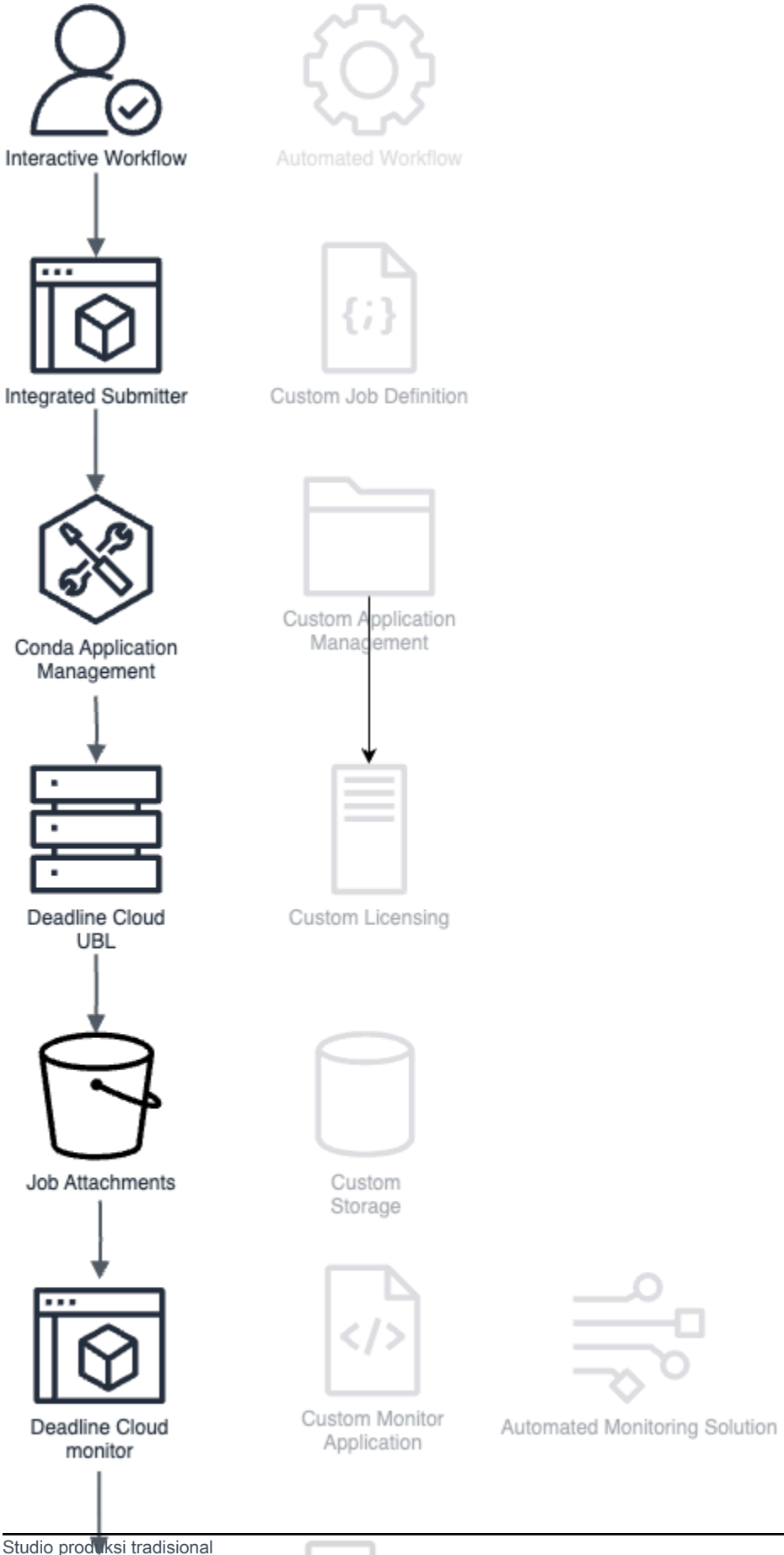
Adalah umum untuk memiliki persyaratan infrastruktur terpisah untuk workstation artis, node render, penyimpanan jaringan, server lisensi, sistem antrian pekerjaan, alat pemantauan, dan manajemen aset. Studio biasanya perlu memelihara beberapa versi alat DCC, perender, plugin, dan alat khusus sambil mengelola pengaturan lisensi yang kompleks di seluruh pertanian render mereka. Infrastruktur studio Anda menjadi lebih rumit ketika Anda mempertimbangkan pengembangan, jaminan kualitas, dan lingkungan produksi.

Penyebaran Deadline Cloud yang khas menggunakan opsi yang dikelola layanan memecahkan atau mengurangi banyak tantangan ini melalui:

- Pengajuan pekerjaan alur kerja interaktif melalui pengirim DCC terintegrasi
- Manajemen aplikasi melalui Deadline Saluran conda yang dikelola Cloud
- Lisensi berbasis penggunaan secara otomatis dikonfigurasi untuk perangkat lunak yang didukung
- Manajemen aset melalui lampiran pekerjaan
- Pemantauan melalui aplikasi monitor Deadline Cloud
- Manajemen infrastruktur melalui armada yang dikelola layanan

Dengan pendekatan ini, seniman dapat mengirimkan pekerjaan langsung dari alat DCC mereka yang sudah dikenal ke cloud render farm yang dapat diskalakan tanpa mengelola infrastruktur yang kompleks. Layanan ini secara otomatis menangani penyebaran perangkat lunak, lisensi, transfer data, dan penskalaan infrastruktur. Artis dapat memantau pekerjaan mereka melalui antarmuka web atau aplikasi desktop, dan output secara otomatis disimpan di Amazon S3 untuk akses mudah.

Dengan konfigurasi ini, studio dapat menciptakan lingkungan pengembangan dan produksi dalam hitungan menit, hanya membayar komputasi dan lisensi yang mereka gunakan, dan fokus pada pekerjaan kreatif daripada manajemen infrastruktur. Pendekatan yang dikelola layanan menyediakan jalur tercepat untuk mengadopsi rendering cloud sambil mempertahankan alur kerja yang akrab bagi seniman.


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

Studio di Awan

Efek visual modern dan studio animasi semakin memindahkan seluruh saluran mereka ke cloud, termasuk workstation artis. Pendekatan ini menghilangkan kebutuhan akan infrastruktur lokal, memungkinkan kolaborasi global, dan menyediakan penskalaan tanpa batas untuk pekerjaan interaktif dan rendering. Namun, ini juga memperkenalkan tantangan baru dalam mengelola sumber daya cloud, memastikan akses latensi rendah ke data, dan mengintegrasikan workstation berbasis cloud dengan render farm.

Studio cloud-native yang khas memerlukan pendekatan terpadu untuk mengelola workstation cloud, penyimpanan bersama, infrastruktur rendering, dan penyebaran perangkat lunak di semua komponen ini. Pendekatan tradisional sering menghasilkan sistem yang kompleks dan dikelola secara manual yang berjuang untuk menyeimbangkan kinerja, biaya, dan fleksibilitas.

Penerapan Deadline Cloud untuk studio cloud-native dapat diimplementasikan menggunakan:

- Pengajuan pekerjaan alur kerja interaktif melalui pengirim DCC terintegrasi di workstation cloud
- Manajemen aplikasi melalui Deadline Saluran conda yang dikelola Cloud merender node
- Lisensi berbasis penggunaan secara otomatis dikonfigurasi untuk perangkat lunak yang didukung
- Akses penyimpanan khusus menggunakan FSx Windows File Server untuk data proyek bersama
- Pemantauan melalui aplikasi monitor Deadline Cloud
- Manajemen infrastruktur menggunakan armada yang dikelola layanan

Pendekatan ini memungkinkan seniman untuk bekerja di workstation berbasis cloud dengan akses langsung ke penyimpanan bersama berkinerja tinggi dan mengirimkan pekerjaan dengan mulus ke pertanian Deadline Cloud. Studio dapat mengelola penyebaran perangkat lunak di kedua workstation dan merender node menggunakan saluran conda yang sama, memastikan konsistensi dan mengurangi overhead pemeliharaan.

Manfaat utama dari konfigurasi ini meliputi:

- Kolaborasi global dengan seniman dapat mengakses workstation dari mana saja
- Lingkungan perangkat lunak yang konsisten di seluruh workstation dan node render
- Penyimpanan bersama berkinerja tinggi yang dapat diakses oleh workstation dan node render
- Penskalaan fleksibel dari sumber daya komputasi interaktif dan batch
- Manajemen terpusat dari semua infrastruktur studio di cloud

Konfigurasi penyimpanan dalam skenario ini biasanya melibatkan:

- FSx untuk Windows File Server untuk data proyek, dapat diakses oleh workstation cloud dan pekerja Deadline Cloud
- Profil penyimpanan di Deadline Cloud untuk mengelola pemetaan jalur antara workstation dan node render
- Pemasangan langsung FSx saham pada pekerja Deadline Cloud menggunakan titik akhir sumber daya VPC dan skrip konfigurasi host

Pendekatan cloud-native ini memungkinkan studio menghilangkan infrastruktur lokal, memungkinkan penskalaan cepat untuk proyek dalam berbagai ukuran sambil mempertahankan alur kerja artis yang sudah dikenal. Ini memberikan fleksibilitas untuk menggunakan campuran sumber daya yang dikelola layanan dan dikelola pelanggan, mengoptimalkan kemudahan manajemen dan persyaratan kinerja tertentu.

Dengan memanfaatkan workstation cloud bersama Deadline Cloud, studio dapat mencapai jalur produksi yang terintegrasi dan dapat diakses secara global yang dapat diskalakan secara mulus dari tim kecil hingga produksi besar.

ECommerce Otomasi

Platform e-commerce modern membutuhkan pembuatan aset otomatis dalam skala besar untuk memberikan visualisasi produk yang kaya di jutaan item. Pendekatan tradisional akan membutuhkan investasi infrastruktur yang signifikan untuk memproses volume besar model 3D ke dalam media produk standar, seringkali menghasilkan sistem yang kurang disediakan yang membuat backlog pemrosesan atau sistem yang disediakan secara berlebihan dengan kapasitas idle.

Alur kerja e-commerce otomatis yang khas perlu menangani pemrosesan unggahan produk, validasi model 3D, manajemen render pertanian, pemrosesan keluaran, dan integrasi dengan sistem informasi produk. Mengelola alur kerja ini secara tradisional memerlukan koordinasi beberapa aplikasi rendering, sumber daya komputasi, dan jalur pemrosesan data sambil memastikan kualitas yang konsisten dan mempertahankan efisiensi biaya dalam skala besar.

Penyebaran Deadline Cloud untuk otomatisasi e-commerce dapat diimplementasikan menggunakan:

- Pengajuan pekerjaan alur kerja otomatis melalui integrasi API kustom dalam aplikasi konsumsi e-niaga yang ada
- Definisi pekerjaan khusus yang disesuaikan dengan visualisasi produk standar

- Manajemen aplikasi melalui Deadline Saluran conda yang dikelola Cloud
- Lisensi berbasis penggunaan secara otomatis dikonfigurasi untuk perangkat lunak yang didukung
- Integrasi Amazon S3 langsung untuk manajemen aset
- Aplikasi pemantauan khusus terintegrasi dengan sistem manajemen produk yang ada
- Armada yang dikelola layanan untuk penskalaan elastis

Pendekatan ini memungkinkan pemrosesan ribuan produk per hari, secara otomatis menghasilkan visualisasi produk standar seperti animasi meja putar. Infrastruktur yang dikelola layanan secara otomatis menskalakan untuk memenuhi permintaan variabel sambil mempertahankan efisiensi biaya melalui penggunaan kembali pekerja dan penerapan aplikasi yang dioptimalkan.


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 Pelanggan

Perangkat lunak pembuatan konten digital tradisional (DCC) biasanya mengharuskan pengguna untuk mempertahankan infrastruktur rendering mereka sendiri atau proses render secara lokal di workstation mereka, yang mengarah ke investasi perangkat keras yang signifikan atau waktu tunggu yang lama yang mengganggu alur kerja kreatif. Untuk vendor perangkat lunak, menyediakan kemampuan rendering cloud yang secara tradisional diperlukan untuk membangun dan memelihara infrastruktur dan sistem penagihan yang kompleks.

Penyebaran Deadline Cloud yang terintegrasi ke dalam perangkat lunak B2C memungkinkan rendering cloud yang mulus langsung di dalam antarmuka yang sudah dikenal pengguna. Integrasi ini menggabungkan:

- Pengajuan pekerjaan alur kerja interaktif tertanam dalam aplikasi DCC
- Batas waktu saluran conda yang dikelola Cloud untuk penerapan aplikasi render
- Lisensi berbasis penggunaan dikonfigurasi secara otomatis
- Manajemen aset melalui lampiran pekerjaan dengan penyimpanan yang dikelola vendor
- Pemantauan khusus terintegrasi langsung di antarmuka DCC
- Armada yang dikelola layanan dibagikan di seluruh pengguna

Pendekatan ini memungkinkan pengguna akhir untuk mengirimkan render ke cloud dengan satu klik dari dalam perangkat lunak mereka, tanpa mengelola akun, infrastruktur, atau pengaturan yang kompleks. Vendor perangkat lunak memelihara lingkungan multi-penyewa di mana:

- Pengguna mengautentikasi melalui kredensi perangkat lunak yang ada
- Pekerjaan secara otomatis dialihkan ke antrian khusus per pengguna
- Aset diisolasi dengan aman menggunakan awalan penyimpanan yang dikendalikan oleh IAM
- Penagihan ditangani melalui sistem vendor yang ada
- Status dan output Job dialirkan langsung kembali ke aplikasi pengguna

Pendekatan armada bersama memastikan kinerja optimal dengan mempertahankan kumpulan pekerja yang hangat, meminimalkan waktu startup sambil memaksimalkan pemanfaatan sumber daya di seluruh basis pengguna. Konfigurasi ini memungkinkan vendor perangkat lunak untuk menawarkan rendering cloud sebagai fitur produk yang mulus daripada layanan terpisah yang memerlukan pengaturan atau akun tambahan.

Pengguna akhir mendapat manfaat dari:

- Pengajuan sekali klik dari antarmuka yang mereka kenal
- Pay-as-you-go Harga Tanpa Manajemen Infrastruktur
- Waktu startup pekerjaan yang cepat melalui infrastruktur bersama
- Pengunduhan otomatis dan pengorganisasian render yang telah selesai
- Pengalaman yang konsisten di semua platform

Pola integrasi ini memungkinkan vendor perangkat lunak untuk menyediakan kemampuan rendering tingkat perusahaan ke seluruh basis pengguna mereka sambil mempertahankan pengalaman sederhana dan ramah konsumen yang terasa asli untuk aplikasi mereka.


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

Apa itu beban kerja Deadline Cloud

Dengan AWS Deadline Cloud, Anda dapat mengirimkan pekerjaan untuk menjalankan aplikasi Anda di cloud dan memproses data untuk produksi konten atau wawasan yang penting bagi bisnis Anda. Deadline Cloud menggunakan Open Job Description (OpenJD) sebagai sintaks untuk template pekerjaan, spesifikasi yang dirancang untuk kebutuhan pipeline komputasi visual tetapi berlaku untuk banyak kasus penggunaan lainnya. Beberapa contoh beban kerja termasuk rendering grafis komputer, simulasi fisika, dan fotogrametri.

Skala beban kerja dari bundel pekerjaan sederhana yang dikirimkan pengguna ke antrian dengan CLI atau GUI yang dibuat secara otomatis, ke plugin pengirim terintegrasi yang secara dinamis menghasilkan bundel pekerjaan untuk beban kerja yang ditentukan aplikasi.

Bagaimana beban kerja muncul dari produksi

Untuk memahami beban kerja dalam konteks produksi dan cara mendukungnya dengan Deadline Cloud, pertimbangkan bagaimana hasilnya. Produksi mungkin melibatkan pembuatan efek visual, animasi, game, citra katalog produk, rekonstruksi 3D untuk pemodelan informasi bangunan (BIM), dan banyak lagi. Konten ini biasanya dibuat oleh tim spesialis artistik atau teknis yang menjalankan berbagai aplikasi perangkat lunak dan skrip khusus. Anggota tim meneruskan data antara satu sama lain menggunakan pipa produksi. Banyak tugas yang dilakukan oleh pipeline melibatkan perhitungan intensif yang akan memakan waktu berhari-hari jika dijalankan di workstation pengguna.

Beberapa contoh tugas dalam jaringan pipa produksi ini meliputi:

- Menggunakan aplikasi fotogrametri untuk memproses foto yang diambil dari set film untuk merekonstruksi mesh digital bertekstur.
- Menjalankan simulasi partikel dalam adegan 3D untuk menambahkan lapisan detail ke efek visual ledakan untuk acara televisi.
- Memasak data untuk level game ke dalam formulir yang diperlukan untuk rilis eksternal dan menerapkan pengaturan optimasi dan kompresi.
- Merender satu set gambar untuk katalog produk termasuk variasi warna, latar belakang, dan pencahayaan.
- Menjalankan skrip yang dikembangkan khusus pada model 3D untuk menerapkan tampilan yang dibuat khusus dan disetujui oleh sutradara film.

Tugas-tugas ini melibatkan banyak parameter untuk disesuaikan untuk mendapatkan hasil artistik atau untuk menyempurnakan kualitas output. Seringkali ada GUI untuk memilih nilai-nilai parameter dengan tombol atau menu untuk menjalankan proses secara lokal dalam aplikasi. Ketika pengguna menjalankan proses, aplikasi dan mungkin komputer host itu sendiri tidak dapat digunakan untuk melakukan operasi lain karena menggunakan status aplikasi dalam memori dan dapat mengkonsumsi semua sumber daya CPU dan memori komputer host.

Dalam banyak kasus prosesnya cepat. Selama produksi, kecepatan proses melambat ketika persyaratan untuk kualitas dan kompleksitas naik. Tes karakter yang memakan waktu 30 detik selama pengembangan dapat dengan mudah berubah menjadi 3 jam ketika diterapkan pada karakter produksi akhir. Melalui perkembangan ini, beban kerja yang memulai kehidupan di dalam GUI dapat tumbuh terlalu besar untuk dipasang. Memportingnya ke Deadline Cloud dapat meningkatkan produktivitas pengguna yang menjalankan proses ini karena mereka mendapatkan kembali kendali penuh atas workstation mereka dan dapat melacak lebih banyak iterasi dari monitor Deadline Cloud.

Ada dua tingkat dukungan yang harus dituju ketika mengembangkan dukungan untuk beban kerja di Deadline Cloud:

- Bongkar beban kerja dari workstation pengguna ke farm Deadline Cloud tanpa paralelisme atau percepatan. Ini mungkin kurang memanfaatkan sumber daya komputasi yang tersedia di pertanian, tetapi kemampuan untuk mengalihkan operasi panjang ke sistem pemrosesan batch memungkinkan pengguna untuk menyelesaikan lebih banyak pekerjaan dengan workstation mereka sendiri.
- Mengoptimalkan paralelisme beban kerja sehingga memanfaatkan skala horizontal Deadline Cloud farm untuk menyelesaikan dengan cepat.

Ada kalanya jelas bagaimana membuat beban kerja berjalan secara paralel. Misalnya, setiap frame render grafis komputer dapat dilakukan secara independen. Namun, penting untuk tidak terjebak pada paralelisme ini. Alih-alih, pahami bahwa menurunkan beban kerja yang berjalan lama ke Deadline Cloud memberikan manfaat yang signifikan, bahkan ketika tidak ada cara yang jelas untuk membagi beban kerja.

Bahan-bahan beban kerja

Untuk menentukan beban kerja Deadline Cloud, terapkan paket pekerjaan yang dikirimkan pengguna ke antrian dengan Deadline Cloud CLI. Sebagian besar pekerjaan dalam membuat bundel pekerjaan adalah menulis template pekerjaan, tetapi ada lebih banyak faktor seperti bagaimana menyediakan

aplikasi yang dibutuhkan beban kerja. Berikut adalah hal-hal penting yang perlu dipertimbangkan saat menentukan beban kerja untuk Deadline Cloud:

- Aplikasi untuk dijalankan. Pekerjaan harus dapat meluncurkan proses aplikasi, dan oleh karena itu memerlukan instalasi aplikasi yang tersedia serta lisensi apa pun yang digunakan aplikasi, seperti akses ke server lisensi mengambang. Ini biasanya bagian dari konfigurasi pertanian, dan tidak tertanam dalam bundel pekerjaan itu sendiri.
  - Konfigurasikan pekerjaan menggunakan lingkungan antrian
  - Connect armada yang dikelola pelanggan ke titik akhir lisensi
- Definisi parameter Job. Pengalaman pengguna dalam mengirimkan pekerjaan sangat dipengaruhi oleh parameter yang disediakannya. Contoh parameter termasuk file data, direktori, dan konfigurasi aplikasi.
  - Nilai parameter elemen untuk bundel pekerjaan
- Aliran data file. Ketika pekerjaan berjalan, ia membaca input dari file yang disediakan oleh pengguna, kemudian menulis outputnya sebagai file baru. Untuk bekerja dengan lampiran pekerjaan dan fitur pemetaan jalur, pekerjaan harus menentukan jalur direktori atau file tertentu untuk input dan output ini.
  - Menggunakan file dalam pekerjaan Anda
- Skrip langkah. Skrip langkah menjalankan biner aplikasi dengan opsi baris perintah yang tepat untuk menerapkan parameter pekerjaan yang disediakan. Ini juga menangani detail seperti pemetaan jalur jika file data beban kerja menyertakan referensi jalur absolut, bukan referensi jalur relatif.
  - Elemen template Job untuk bundel pekerjaan

Portabilitas beban kerja

Beban kerja portabel ketika dapat berjalan di beberapa sistem yang berbeda tanpa mengubahnya setiap kali Anda mengirimkan pekerjaan. Misalnya, ini mungkin berjalan di farm render berbeda yang memiliki sistem file bersama yang berbeda yang dipasang, atau pada sistem operasi yang berbeda seperti Linux atauWindows. Saat Anda menerapkan bundel pekerjaan portabel, lebih mudah bagi pengguna untuk menjalankan pekerjaan di pertanian spesifik mereka, atau menyesuaikannya untuk kasus penggunaan lainnya.

Berikut adalah beberapa cara Anda dapat membuat bundel pekerjaan Anda portabel.

- Tentukan sepenuhnya file data input yang dibutuhkan oleh beban kerja, menggunakan parameter `PATH` pekerjaan dan referensi aset dalam bundel pekerjaan. Pendekatan ini membuat pekerjaan

portabel ke peternakan berdasarkan sistem file bersama dan ke peternakan yang membuat salinan data input, seperti fitur lampiran pekerjaan Deadline Cloud.

- Buat referensi jalur file untuk file input pekerjaan dapat dipindahkan dan dapat digunakan pada sistem operasi yang berbeda. Misalnya ketika pengguna mengirimkan pekerjaan dari Windows workstation untuk dijalankan pada Linux armada.
  - Gunakan referensi jalur file relatif, jadi jika direktori yang berisi mereka dipindahkan ke lokasi yang berbeda, referensi masih diselesaikan. Beberapa aplikasi, seperti Blender, mendukung pilihan antara jalur relatif dan absolut.
  - Jika Anda tidak dapat menggunakan jalur relatif, dukung metadata pemetaan jalur OpenJD dan terjemahkan jalur absolut sesuai dengan cara Deadline Cloud menyediakan file ke pekerjaan.
- Menerapkan perintah dalam pekerjaan menggunakan skrip portabel. Python dan bash adalah dua contoh bahasa scripting yang dapat digunakan dengan cara ini. Anda harus mempertimbangkan untuk menyediakan keduanya di semua host pekerja armada Anda.
  - Gunakan biner interpreter skrip, seperti `python` atau`bash`, dengan nama file skrip sebagai argumen. Pendekatan ini bekerja pada semua sistem operasi termasukWindows, dibandingkan dengan menggunakan file skrip dengan bit eksekusi diaturLinux.
  - Tulis skrip bash portabel dengan menerapkan praktik ini:
    - Perluas parameter jalur template dalam tanda kutip tunggal untuk menangani jalur dengan spasi dan pemisah Windows jalur.
    - Saat berjalanWindows, perhatikan masalah yang terkait dengan terjemahan jalur otomatis MinGW. Misalnya, ia mengubah AWS CLI perintah seperti `aws logs tail /aws/deadline/...` menjadi perintah yang mirip dengan `aws logs tail "C:/Program Files/Git/aws/deadline/..."` dan tidak akan mengekor log dengan benar. Tetapkan variabel `MSYS_NO_PATHCONV=1` untuk menonaktifkan perilaku ini.
    - Dalam kebanyakan kasus, kode yang sama berfungsi pada semua sistem operasi. Ketika kode harus berbeda, gunakan `if/else` konstruksi untuk menangani kasus.

```
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
```

  - Anda dapat menulis skrip Python portabel menggunakan `pathlib` untuk menangani perbedaan jalur sistem file dan menghindari fitur khusus pengoperasian. Dokumentasi Python menyertakan

anotasi untuk ini, misalnya dalam dokumentasi pustaka sinyal. Linux-Dukungan fitur khusus ditandai sebagai “Ketersediaan: Linux.”

- Gunakan parameter pekerjaan untuk menentukan persyaratan aplikasi. Gunakan konvensi konsisten yang dapat diterapkan administrator pertanian di lingkungan antrian.
  - Misalnya, Anda dapat menggunakan `CondaPackages` dan/atau `RezPackages` parameter dalam pekerjaan Anda, dengan nilai parameter default yang mencantumkan nama paket aplikasi dan versi yang dibutuhkan pekerjaan. Kemudian, Anda dapat menggunakan salah satu contoh lingkungan antrian conda atau Rez untuk menyediakan lingkungan virtual untuk pekerjaan itu.

Memulai dengan sumber daya Deadline Cloud

Untuk mulai membuat solusi khusus untuk AWS Deadline Cloud, Anda harus menyiapkan sumber daya Anda. Ini termasuk pertanian, setidaknya satu antrian untuk pertanian, dan setidaknya satu armada pekerja untuk melayani antrian. Anda dapat membuat sumber daya menggunakan konsol Deadline Cloud, atau Anda dapat menggunakan. AWS Command Line Interface

Dalam tutorial ini, Anda akan menggunakan AWS CloudShell untuk membuat peternakan pengembang sederhana dan menjalankan agen pekerja. Anda kemudian dapat mengirimkan dan menjalankan pekerjaan sederhana dengan parameter dan lampiran, menambahkan armada yang dikelola layanan, dan membersihkan sumber daya pertanian Anda setelah selesai.

Bagian berikut memperkenalkan Anda ke berbagai fitur Deadline Cloud, dan bagaimana mereka berfungsi dan bekerja sama. Mengikuti langkah-langkah ini berguna untuk mengembangkan dan menguji beban kerja dan penyesuaian baru.

Untuk petunjuk cara menyiapkan farm menggunakan konsol, lihat Memulai di Panduan Pengguna Cloud Tenggat Waktu.

Topik

- Buat pertanian Cloud Deadline
- Jalankan agen pekerja Deadline Cloud
- Kirim dengan Deadline Cloud
- Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud
- Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud
- Bersihkan sumber daya pertanian Anda di Deadline Cloud

Buat pertanian Cloud Deadline

Untuk membuat resource farm dan antrean developer di AWS Deadline Cloud, gunakan AWS Command Line Interface (AWS CLI), seperti yang ditunjukkan pada prosedur berikut. Anda juga akan membuat peran AWS Identity and Access Management (IAM) dan armada yang dikelola pelanggan (CMF) dan mengaitkan armada dengan antrian Anda. Kemudian Anda dapat mengonfigurasi AWS CLI dan mengonfirmasi bahwa peternakan Anda sudah diatur dan berfungsi seperti yang ditentukan.

Anda dapat menggunakan peternakan ini untuk menjelajahi fitur Deadline Cloud, kemudian mengembangkan dan menguji beban kerja, penyesuaian, dan integrasi pipeline baru.

Untuk membuat peternakan

1. Buka AWS CloudShell sesi. Anda akan menggunakan CloudShell jendela untuk memasukkan AWS Command Line Interface (AWS CLI) perintah untuk menjalankan contoh dalam tutorial ini. Biarkan CloudShell jendela tetap terbuka saat Anda melanjutkan.
2. Buat nama untuk peternakan Anda, dan tambahkan nama pertanian itu ke`~/.bashrc`. Ini akan membuatnya tersedia untuk sesi terminal lainnya.

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

3. Buat sumber daya pertanian, dan tambahkan ID pertaniannya ke`~/.bashrc`.

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

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

4. Buat sumber daya antrian, dan tambahkan ID antreannya `~/.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. Buat peran IAM untuk armada. Peran ini memberi host pekerja di armada Anda kredensi keamanan yang diperlukan untuk menjalankan pekerjaan dari antrian Anda.

```
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. Buat armada yang dikelola pelanggan (CMF), dan tambahkan ID armadanya ke. `~/.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. Kaitkan CMF dengan antrian Anda.

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

8. Instal antarmuka baris perintah Deadline Cloud.

```
pip install deadline
```

9. Untuk menyetel farm default ke ID farm dan antrean ke ID antrian yang Anda buat sebelumnya, gunakan perintah berikut.

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

10. (Opsional) Untuk mengonfirmasi bahwa peternakan Anda diatur sesuai dengan spesifikasi Anda, gunakan perintah berikut:

    - Daftar semua peternakan — **`deadline farm list`**
    - Buat daftar semua antrian di pertanian default — **`deadline queue list`**
    - Daftar semua armada di peternakan default — **`deadline fleet list`**
    - Dapatkan peternakan default — **`deadline farm get`**
    - Dapatkan antrian default — **`deadline queue get`**
    - Dapatkan semua armada yang terkait dengan antrian default — **`deadline fleet get`**

Langkah berikutnya

Setelah membuat peternakan, Anda dapat menjalankan agen pekerja Deadline Cloud di host di armada Anda untuk memproses pekerjaan. Lihat Jalankan agen pekerja Deadline Cloud.

Jalankan agen pekerja Deadline Cloud

Sebelum Anda dapat menjalankan pekerjaan yang Anda kirimkan ke antrian di peternakan pengembang Anda, Anda harus menjalankan agen pekerja AWS Deadline Cloud dalam mode pengembang pada host pekerja.

Sepanjang sisa tutorial ini, Anda akan melakukan AWS CLI operasi di peternakan pengembang Anda menggunakan dua AWS CloudShell tab. Di tab pertama, Anda dapat mengirimkan pekerjaan. Di tab kedua, Anda dapat menjalankan agen pekerja.

> **Note**
>
> Jika Anda membiarkan CloudShell sesi Anda menganggur selama lebih dari 20 menit, itu akan batas waktu dan menghentikan agen pekerja. Untuk memulai kembali agen pekerja, ikuti instruksi dalam prosedur berikut.

Sebelum Anda dapat memulai agen pekerja, Anda harus menyiapkan pertanian Deadline Cloud, antrian, dan armada. Lihat Buat pertanian Cloud Deadline.

Untuk menjalankan agen pekerja dalam mode pengembang

1. Dengan pertanian Anda masih terbuka di CloudShell tab pertama, buka CloudShell tab kedua, lalu buat `demoenv-persist` direktori `demoenv-logs` dan.

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

2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

   > **Note**
   >
   > Pada Windows, diperlukan bahwa file agen diinstal ke direktori paket situs global Python. Lingkungan virtual Python saat ini tidak didukung.

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

3. Untuk memungkinkan agen pekerja membuat direktori sementara untuk menjalankan pekerjaan, buat direktori:

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

4. Jalankan agen pekerja Deadline Cloud dalam mode pengembang dengan variabel `DEV_FARM_ID` dan `DEV_CMF_ID` yang Anda tambahkan ke`~/.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
```

   Saat agen pekerja menginisialisasi dan kemudian melakukan polling pada operasi `UpdateWorkerSchedule` API, output berikut ditampilkan:

```
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. Pilih CloudShell tab pertama Anda, lalu daftarkan pekerja di armada.

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

Output seperti berikut ini ditampilkan:

```
Displaying 1 of 1 workers starting at 0

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

Dalam konfigurasi produksi, agen pekerja Deadline Cloud memerlukan pengaturan beberapa pengguna dan direktori konfigurasi sebagai pengguna administratif di mesin host. Anda dapat mengganti pengaturan ini karena Anda menjalankan pekerjaan di peternakan pengembangan Anda sendiri, yang hanya dapat Anda akses.

Langkah selanjutnya

Sekarang agen pekerja berjalan di host pekerja Anda, Anda dapat mengirim pekerjaan ke pekerja Anda. Anda dapat:

- Kirim dengan Deadline Cloudmenggunakan bundel pekerjaan OpenJD sederhana.
- Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloudyang berbagi file antar workstation menggunakan sistem operasi yang berbeda.

Kirim dengan Deadline Cloud

Untuk menjalankan pekerjaan Deadline Cloud di host pekerja Anda, Anda membuat dan menggunakan paket pekerjaan Open Job Description (OpenJD) untuk mengonfigurasi pekerjaan. Bundel mengkonfigurasi pekerjaan, misalnya dengan menentukan file input untuk pekerjaan dan di mana untuk menulis output pekerjaan. Topik ini mencakup contoh cara Anda dapat mengonfigurasi bundel pekerjaan.

Sebelum Anda dapat mengikuti prosedur di bagian ini, Anda harus menyelesaikan yang berikut:

- Buat pertanian Cloud Deadline
- Jalankan agen pekerja Deadline Cloud

Untuk menggunakan AWS Deadline Cloud untuk menjalankan pekerjaan, gunakan prosedur berikut. Gunakan AWS CloudShell tab pertama untuk mengirimkan pekerjaan ke peternakan pengembang Anda. Gunakan CloudShell tab kedua untuk melihat output agen pekerja.

Topik

Kirim simple_job sampelnya

Setelah membuat peternakan dan menjalankan agen pekerja, Anda dapat mengirimkan simple_job sampel ke Deadline Cloud.

Untuk mengirimkan simple_job sampel ke Deadline Cloud

1. Pilih CloudShell tab pertama Anda.
2. Unduh sampel dari GitHub.

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

3. Arahkan ke direktori sampel bundel pekerjaan.

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

4. Kirim simple_job sampel.

```
deadline bundle submit simple_job
```

5. Pilih CloudShell tab kedua Anda untuk melihat output logging tentang panggilan`BatchGetJobEntities`, mendapatkan sesi, dan menjalankan tindakan sesi.

```
...
[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**
>
> Hanya output logging dari agen pekerja yang ditampilkan. Ada log terpisah untuk sesi yang menjalankan pekerjaan.

6. Pilih tab pertama Anda, lalu periksa file log yang ditulis agen pekerja.

   a. Arahkan ke direktori log agen pekerja dan lihat isinya.

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

   b. Cetak file log pertama yang dibuat oleh agen pekerja.

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

   File ini berisi output agen pekerja tentang bagaimana itu disebut Deadline Cloud API untuk membuat sumber daya pekerja di armada Anda, dan kemudian mengambil peran armada.

   c. Cetak output file log saat agen pekerja bergabung dengan armada.

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

   Log ini berisi output tentang semua tindakan yang diambil agen pekerja, tetapi tidak berisi output tentang antrian tempat ia menjalankan pekerjaan, kecuali untuk sumber daya tersebut IDs .

   d. Cetak file log untuk setiap sesi dalam direktori yang diberi nama sama dengan id sumber daya antrian.

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

   Jika pekerjaan berhasil, output file log akan mirip dengan yang berikut ini:

   ```
   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. Cetak informasi tentang pekerjaan itu.

   ```
   deadline job get
   ```

   Saat Anda mengirimkan pekerjaan, sistem menyimpannya sebagai default sehingga Anda tidak perlu memasukkan ID pekerjaan.

Kirim simple_job dengan parameter

Anda dapat mengirimkan pekerjaan dengan parameter. Dalam prosedur berikut, Anda mengedit simple_job template untuk menyertakan pesan khusus, mengirimkansimple_job, lalu mencetak file log sesi untuk melihat pesan.

Untuk mengirimkan simple_job sampel dengan parameter

1. Pilih CloudShell tab pertama Anda, lalu arahkan ke direktori sampel bundel pekerjaan.

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

2. Cetak isi simple_job template.

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

`parameterDefinitions`Bagian dengan `Message` parameter akan terlihat seperti berikut:

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

3. Kirim simple_job sampel dengan nilai parameter, lalu tunggu pekerjaan selesai berjalan.

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

4. Untuk melihat pesan kustom, lihat file log sesi terbaru.

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

Buat bundel pekerjaan simple_file_job dengan file I/O

Pekerjaan render perlu membaca definisi adegan, merender gambar darinya, dan kemudian menyimpan gambar itu ke file output. Anda dapat mensimulasikan tindakan ini dengan membuat pekerjaan menghitung hash input alih-alih merender gambar.

Untuk membuat bundel pekerjaan simple_file_job dengan file I/O

1. Pilih CloudShell tab pertama Anda, lalu arahkan ke direktori sampel bundel pekerjaan.

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

2. Buat salinan `simple_job` dengan nama baru`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Edit template pekerjaan sebagai berikut:

> **Note**
>
> Kami menyarankan Anda menggunakan nano langkah-langkah ini. Jika Anda lebih suka menggunakanVim, Anda harus mengatur mode tempel menggunakan`:set paste`.

a. Buka template di editor teks.

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

b. Tambahkan yang berikut ini`type`,`objectType`, dan `dataFlow`.`parameterDefinitions`

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

c. Tambahkan perintah `bash` script berikut ke akhir file yang membaca dari file input dan menulis ke file output.

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

Yang diperbarui `template.yaml` harus sama persis dengan yang berikut:

```
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**
>
> Jika Anda ingin menyesuaikan spasi di`template.yaml`, pastikan Anda menggunakan spasi alih-alih lekukan.

d. Simpan file, dan keluar dari editor teks.

4. Berikan nilai parameter untuk file input dan output untuk mengirimkan simple_file_job.

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

5. Cetak informasi tentang pekerjaan itu.

```
deadline job get
```

- Anda akan melihat output seperti berikut ini:

```
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
```

- Meskipun Anda hanya menyediakan jalur relatif, parameter memiliki jalur lengkap yang disetel. AWS CLI Menggabungkan direktori kerja saat ini ke jalur apa pun yang disediakan sebagai parameter saat jalur memiliki tipePATH.
- Agen pekerja yang berjalan di jendela terminal lain mengambil dan menjalankan pekerjaan. Tindakan ini membuat `hash.txt` file, yang dapat Anda lihat dengan perintah berikut.

  ```
  cat hash.txt
  ```

  Perintah ini akan mencetak output yang mirip dengan berikut ini.

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

Langkah selanjutnya

Setelah mempelajari cara mengirimkan pekerjaan sederhana menggunakan Deadline Cloud CLI, Anda dapat menjelajahi:

- Kirim pekerjaan dengan lampiran pekerjaan di Deadline Clouduntuk mempelajari cara menjalankan pekerjaan pada host yang menjalankan sistem operasi yang berbeda.
- Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Clouduntuk menjalankan pekerjaan Anda di host yang dikelola oleh Deadline Cloud.
- Bersihkan sumber daya pertanian Anda di Deadline Clouduntuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud

Banyak peternakan menggunakan sistem file bersama untuk berbagi file antara host yang mengirimkan pekerjaan dan yang menjalankan pekerjaan. Misalnya, pada `simple_file_job` contoh sebelumnya, sistem file lokal dibagi antara jendela AWS CloudShell terminal, yang berjalan di tab satu tempat Anda mengirimkan pekerjaan, dan tab dua tempat Anda menjalankan agen pekerja.

Sistem file bersama menguntungkan ketika workstation submitter dan host pekerja berada di jaringan area lokal yang sama. Jika Anda menyimpan data Anda di lokasi dekat workstation yang mengaksesnya, maka menggunakan farm berbasis cloud berarti Anda harus membagikan sistem file Anda melalui VPN latensi tinggi atau menyinkronkan sistem file Anda di cloud. Tak satu pun dari opsi ini mudah diatur atau dioperasikan.

AWS Deadline Cloud menawarkan solusi sederhana dengan lampiran pekerjaan, yang mirip dengan lampiran email. Dengan lampiran pekerjaan, Anda melampirkan data ke pekerjaan Anda. Kemudian, Deadline Cloud menangani detail transfer dan penyimpanan data pekerjaan Anda di bucket Amazon Simple Storage Service (Amazon S3).

Alur kerja pembuatan konten sering berulang, artinya pengguna mengirimkan pekerjaan dengan subset kecil file yang dimodifikasi. Karena bucket Amazon S3 menyimpan lampiran pekerjaan dalam penyimpanan yang dapat dialamatkan konten, nama setiap objek didasarkan pada hash data objek dan konten pohon direktori disimpan dalam format file manifes yang dilampirkan ke pekerjaan.

Sebelum Anda dapat mengikuti prosedur di bagian ini, Anda harus menyelesaikan yang berikut:

- Buat pertanian Cloud Deadline
- Jalankan agen pekerja Deadline Cloud

Untuk menjalankan pekerjaan dengan lampiran pekerjaan, selesaikan langkah-langkah berikut.

Topik

- Tambahkan konfigurasi lampiran pekerjaan ke antrian Anda
- Kirim simple_file_job dengan lampiran pekerjaan
- Memahami bagaimana lampiran pekerjaan disimpan di Amazon S3
- Langkah selanjutnya

Tambahkan konfigurasi lampiran pekerjaan ke antrian Anda

Untuk mengaktifkan lampiran pekerjaan dalam antrian Anda, tambahkan konfigurasi lampiran pekerjaan ke sumber daya antrian di akun Anda.

Untuk menambahkan konfigurasi lampiran pekerjaan ke antrian

1. Pilih CloudShell tab pertama Anda, lalu masukkan salah satu perintah berikut untuk menggunakan bucket Amazon S3 untuk lampiran pekerjaan.

- Jika Anda tidak memiliki bucket Amazon S3 pribadi yang sudah ada, Anda dapat membuat dan menggunakan bucket S3 baru.

```
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}
```

- Jika Anda sudah memiliki bucket Amazon S3 pribadi, Anda dapat menggunakannya *MY_BUCKET_NAME* dengan menggantinya dengan nama bucket Anda.

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

2. Setelah membuat atau memilih bucket Amazon S3, tambahkan nama bucket agar bucket tersedia `~/.bashrc` untuk sesi terminal lainnya.

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

3. Buat peran AWS Identity and Access Management (IAM) untuk antrian.

```
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. Perbarui antrian Anda untuk menyertakan pengaturan lampiran pekerjaan dan peran 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. Konfirmasikan bahwa Anda memperbarui antrian Anda.

```
deadline queue get
```

Output seperti berikut ini ditampilkan:

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

Kirim simple_file_job dengan lampiran pekerjaan

Saat Anda menggunakan lampiran pekerjaan, paket pekerjaan harus memberi Deadline Cloud informasi yang cukup untuk menentukan aliran data pekerjaan, seperti menggunakan parameter. PATH Dalam kasussimple_file_job, Anda mengedit `template.yaml` file untuk memberi tahu Deadline Cloud bahwa aliran data ada di file input dan file output.

Setelah menambahkan konfigurasi lampiran pekerjaan ke antrian, Anda dapat mengirimkan sampel simple_file_job dengan lampiran pekerjaan. Setelah Anda melakukan ini, Anda dapat melihat logging dan output pekerjaan untuk mengonfirmasi bahwa lampiran pekerjaan simple_file_job dengan berfungsi.

Untuk mengirimkan bundel pekerjaan simple_file_job dengan lampiran pekerjaan

1. Pilih CloudShell tab pertama Anda, lalu buka `JobBundle-Samples` direktori.
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```
3. Kirim simple_file_job ke antrian. Saat diminta untuk mengonfirmasi unggahan, masukkan**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```
4. Untuk melihat output log sesi transfer data lampiran pekerjaan, jalankan perintah berikut.

```
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. Buat daftar tindakan sesi yang dijalankan dalam sesi.

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

Output seperti berikut ini ditampilkan:

```
{
    "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"
                }
            }
        }
    ]
```

```
}
```

Tindakan sesi pertama mengunduh lampiran pekerjaan input, sedangkan tindakan kedua menjalankan tugas seperti pada langkah sebelumnya dan kemudian mengunggah lampiran pekerjaan keluaran.

6. Daftar direktori output.

```
ls *.txt
```

Output seperti `hash.txt` ada di direktori, tetapi `hash-jobattachments.txt` tidak ada karena file output dari pekerjaan belum diunduh.

7. Unduh output dari pekerjaan terbaru.

```
deadline job download-output
```

8. Lihat output dari file yang diunduh.

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

Output seperti berikut ini ditampilkan:

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

Memahami bagaimana lampiran pekerjaan disimpan di Amazon S3

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengunggah atau mengunduh data untuk lampiran pekerjaan, yang disimpan di bucket Amazon S3. Memahami bagaimana Deadline Cloud menyimpan lampiran pekerjaan di Amazon S3 akan membantu saat Anda mengembangkan beban kerja dan integrasi pipeline.

Untuk memeriksa bagaimana lampiran pekerjaan Deadline Cloud disimpan di Amazon S3

1. Pilih CloudShell tab pertama Anda, lalu buka direktori sampel bundel pekerjaan.

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

2. Periksa properti pekerjaan.

```
deadline job get
```

Output seperti berikut ini ditampilkan:

```
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
```

Bidang lampiran berisi daftar struktur manifes yang menjelaskan jalur data input dan output yang digunakan pekerjaan saat dijalankan. Lihatlah `rootPath` untuk melihat jalur direktori lokal pada mesin yang mengirimkan pekerjaan. Untuk melihat akhiran objek Amazon S3 yang berisi file manifes, tinjau. `inputManifestFile` File manifes berisi metadata untuk snapshot pohon direktori dari data input pekerjaan.

3. Cetak objek manifes Amazon S3 dengan cantik untuk melihat struktur direktori input untuk pekerjaan tersebut.

```
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 .
```

Output seperti berikut ini ditampilkan:

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

4. Buat awalan Amazon S3 yang menyimpan manifes untuk lampiran pekerjaan keluaran dan daftarkan objek di bawahnya.

```
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
```

Lampiran pekerjaan keluaran tidak direferensikan secara langsung dari sumber daya pekerjaan tetapi ditempatkan di bucket Amazon S3 berdasarkan sumber daya pertanian. IDs

5. Dapatkan kunci objek manifes terbaru untuk id tindakan sesi tertentu, lalu cetak objek manifes dengan cantik.

```
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 .
```

Anda akan melihat properti file `hash-jobattachments.txt` dalam output seperti berikut ini:

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

Pekerjaan Anda hanya akan memiliki satu objek manifes per tugas yang dijalankan, tetapi secara umum dimungkinkan untuk memiliki lebih banyak objek per tugas yang dijalankan.

6. Lihat output penyimpanan Amazon S3 yang dapat dialamatkan konten di bawah awalan. `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 -
```

Output seperti berikut ini ditampilkan:

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

Langkah selanjutnya

Setelah mempelajari cara mengirimkan pekerjaan dengan lampiran menggunakan Deadline Cloud CLI, Anda dapat menjelajahi:

- Kirim dengan Deadline Clouduntuk mempelajari cara menjalankan pekerjaan menggunakan bundel OpenJD di host pekerja Anda.
- Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Clouduntuk menjalankan pekerjaan Anda di host yang dikelola oleh Deadline Cloud.
- Bersihkan sumber daya pertanian Anda di Deadline Clouduntuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud

AWS CloudShell tidak menyediakan kapasitas komputasi yang cukup untuk menguji beban kerja yang lebih besar. Ini juga tidak dikonfigurasi untuk bekerja dengan pekerjaan yang mendistribusikan tugas di beberapa host pekerja.

Alih-alih menggunakan CloudShell, Anda dapat menambahkan armada terkelola layanan Auto Scaling (SMF) ke peternakan pengembang Anda. SMF menyediakan kapasitas komputasi yang cukup untuk beban kerja yang lebih besar dan dapat menangani pekerjaan yang perlu mendistribusikan tugas pekerjaan di beberapa host pekerja.

Sebelum menambahkan SMF, Anda harus menyiapkan pertanian, antrian, dan armada Deadline Cloud. Lihat Buat pertanian Cloud Deadline.

Untuk menambahkan armada yang dikelola layanan ke peternakan pengembang Anda

1. Pilih AWS CloudShell tab pertama Anda, lalu buat armada yang dikelola layanan dan tambahkan ID armadanya. `.bashrc` Tindakan ini membuatnya tersedia untuk sesi terminal lainnya.

```
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. Kaitkan SMF dengan antrian Anda.

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

3. Kirim simple_file_job ke antrian. Saat diminta untuk mengonfirmasi unggahan, masukkan**y**.

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

4. Konfirmasikan SMF berfungsi dengan benar.

```
deadline fleet get
```

- Pekerja mungkin membutuhkan waktu beberapa menit untuk memulai. Ulangi `deadline fleet get` perintah sampai Anda dapat melihat bahwa armada sedang berjalan.
- Armada yang dikelola `queueFleetAssociationsStatus` untuk layanan akan. `ACTIVE`
- SMF `autoScalingStatus` akan berubah dari `GROWING` ke`STEADY`.

Status Anda akan terlihat mirip dengan yang berikut ini:

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

5. Lihat log untuk pekerjaan yang Anda kirimkan. Log ini disimpan dalam log di Amazon CloudWatch Logs, bukan sistem CloudShell file.

```
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
```

Langkah selanjutnya

Setelah membuat dan menguji armada yang dikelola layanan, Anda harus menghapus sumber daya yang Anda buat untuk menghindari biaya yang tidak perlu.

- Bersihkan sumber daya pertanian Anda di Deadline Clouduntuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Bersihkan sumber daya pertanian Anda di Deadline Cloud

Untuk mengembangkan dan menguji beban kerja baru dan integrasi pipeline, Anda dapat terus menggunakan Deadline Cloud developer farm yang Anda buat untuk tutorial ini. Jika Anda tidak lagi membutuhkan peternakan pengembang, Anda dapat menghapus sumber dayanya termasuk peran pertanian, armada, antrian, AWS Identity and Access Management (IAM), dan log di Amazon CloudWatch Logs. Setelah Anda menghapus sumber daya ini, Anda harus memulai tutorial lagi untuk menggunakan sumber daya. Untuk informasi selengkapnya, lihat Memulai dengan sumber daya Deadline Cloud.

Untuk membersihkan sumber daya pertanian pengembang

1. Pilih CloudShell tab pertama Anda, lalu hentikan semua asosiasi antrian-armada untuk antrian Anda.

```
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. Buat daftar asosiasi armada antrian.

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

   Anda mungkin perlu menjalankan kembali perintah sampai laporan output`"status": "STOPPED"`, maka Anda dapat melanjutkan ke langkah berikutnya. Proses ini bisa memakan waktu beberapa menit untuk menyelesaikannya.

```
{
    "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. Hapus semua asosiasi antrian-armada untuk antrian Anda.

```
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. Hapus semua armada yang terkait dengan antrian Anda.

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

5. Hapus antrian.

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

6. Hapus peternakan.

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

7. Hapus AWS sumber daya lain untuk peternakan Anda.

   a. Hapus peran armada 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. Hapus peran IAM antrian.

```
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. Hapus grup CloudWatch log Amazon Logs. Setiap antrian dan armada memiliki grup log mereka sendiri.

```
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"
```

Buat lowongan kerja untuk dikirimkan ke Deadline Cloud

Anda mengirimkan pekerjaan ke Deadline Cloud menggunakan paket pekerjaan. Paket pekerjaan adalah kumpulan file, termasuk template pekerjaan Open Job Description (OpenJD) dan file aset apa pun yang diperlukan untuk merender pekerjaan.

Template pekerjaan menjelaskan bagaimana pekerja memproses dan mengakses aset, dan menyediakan skrip yang dijalankan pekerja. Paket Job memungkinkan artis, direktur teknis, dan pengembang pipeline untuk dengan mudah mengirimkan pekerjaan kompleks ke Deadline Cloud dari workstation lokal atau farm render lokal mereka. Paket Job sangat berguna bagi tim yang mengerjakan efek visual skala besar, animasi, atau proyek rendering media lainnya yang membutuhkan sumber daya komputasi sesuai permintaan yang dapat diskalakan.

Anda dapat membuat bundel pekerjaan menggunakan sistem file lokal untuk menyimpan file dan editor teks untuk membuat template pekerjaan. Setelah membuat bundel, kirimkan pekerjaan ke Deadline Cloud menggunakan Deadline Cloud CLI atau alat seperti pengirim Deadline Cloud

Anda dapat menyimpan aset Anda dalam sistem file yang dibagikan di antara pekerja Anda, atau Anda dapat menggunakan lampiran pekerjaan Deadline Cloud untuk mengotomatiskan pemindahan aset ke bucket S3 tempat pekerja Anda dapat mengaksesnya. Lampiran Job juga membantu memindahkan output dari pekerjaan Anda kembali ke workstation Anda.

Bagian berikut memberikan petunjuk terperinci tentang membuat dan mengirimkan paket pekerjaan ke Deadline Cloud.

Topik

Templat Open Job Description (OpenJD) untuk Deadline Cloud

Paket pekerjaan adalah salah satu alat yang Anda gunakan untuk menentukan pekerjaan untuk AWS Deadline Cloud. Mereka mengelompokkan template Open Job Description (OpenJD) dengan informasi tambahan seperti file dan direktori yang digunakan pekerjaan Anda dengan lampiran pekerjaan. Anda menggunakan antarmuka baris perintah Deadline Cloud (CLI) untuk menggunakan bundel pekerjaan untuk mengirimkan pekerjaan agar antrian dapat dijalankan.

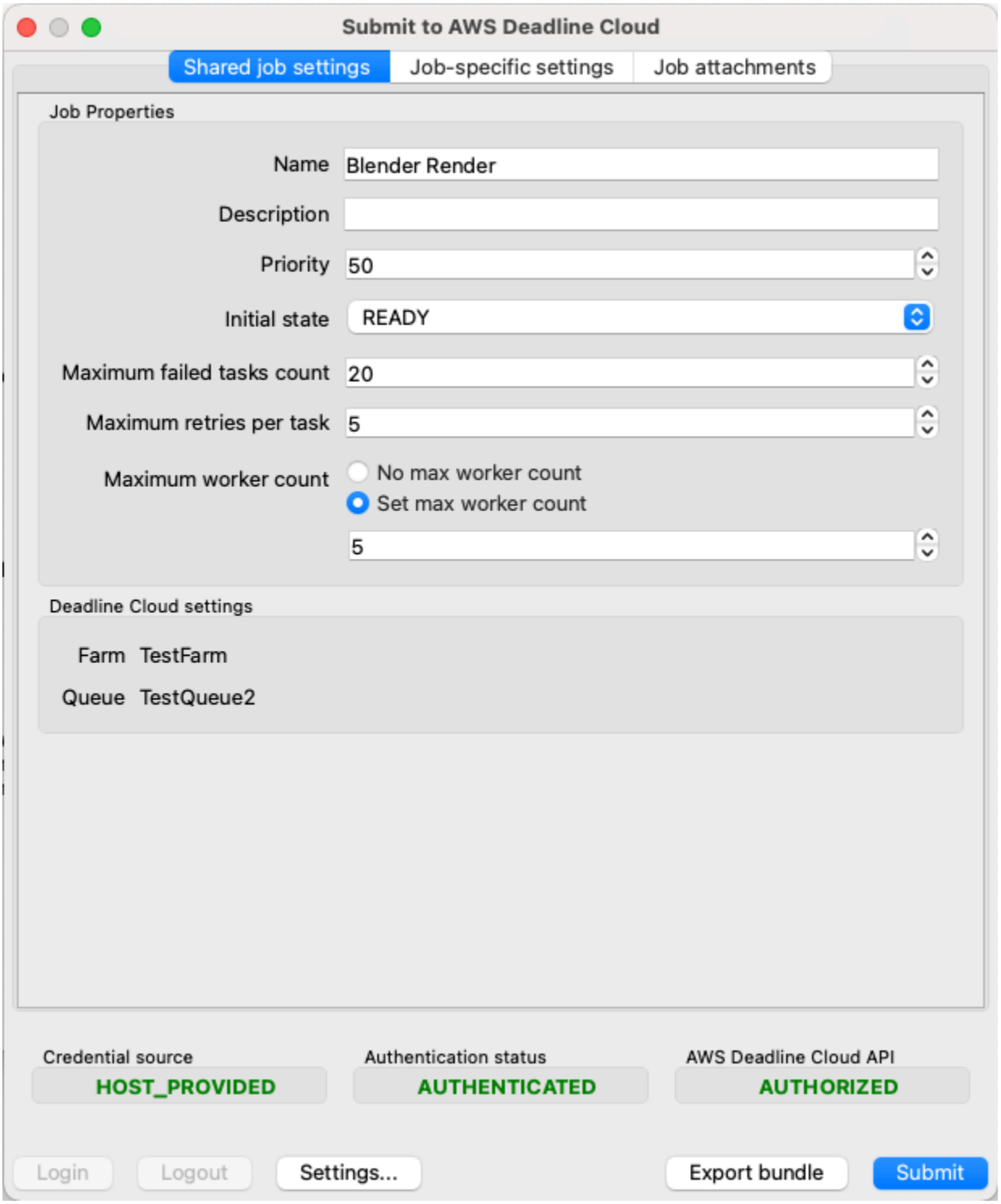
Bundel pekerjaan adalah struktur direktori yang berisi template pekerjaan OpenJD, file lain yang menentukan pekerjaan, dan file khusus pekerjaan yang diperlukan sebagai input untuk pekerjaan Anda. Anda dapat menentukan file yang menentukan pekerjaan Anda sebagai file YAMAL atau JSON.

Satu-satunya file yang diperlukan adalah salah satu `template.yaml` atau`template.json`. Anda juga dapat menyertakan file-file berikut:

```
/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
```

Gunakan bundel pekerjaan untuk pengiriman pekerjaan khusus dengan Deadline Cloud CLI dan lampiran pekerjaan, atau Anda dapat menggunakan antarmuka pengiriman grafis. Misalnya, berikut ini adalah sampel Blender dari GitHub. Untuk menjalankan sampel menggunakan perintah berikut di direktori sampel Blender:

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



Panel pengaturan khusus pekerjaan dihasilkan dari `userInterface` properti parameter pekerjaan yang ditentukan dalam templat pekerjaan.

Untuk mengirimkan pekerjaan menggunakan baris perintah, Anda dapat menggunakan perintah yang mirip dengan berikut ini

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

Atau Anda dapat menggunakan `deadline.client.api.create_job_from_job_bundle` fungsi dalam paket `deadline` Python.

Semua plugin pengirim pekerjaan yang disediakan dengan Deadline Cloud, seperti plugin Autodesk Maya, menghasilkan bundel pekerjaan untuk kiriman Anda dan kemudian gunakan paket Deadline Cloud Python untuk mengirimkan pekerjaan Anda ke Deadline Cloud. Anda dapat melihat bundel pekerjaan yang dikirimkan di direktori riwayat pekerjaan stasiun kerja Anda atau dengan menggunakan pengirim. Anda dapat menemukan direktori riwayat pekerjaan Anda dengan perintah berikut:

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

Ketika pekerjaan Anda berjalan pada pekerja Deadline Cloud, ia memiliki akses ke variabel lingkungan yang memberikan informasi tentang pekerjaan tersebut. Variabel lingkungan adalah:

Nama variabel	Available
DEADLINE_FARM_ID	Semua tindakan
DEADLINE_FLEET_ID	Semua tindakan
DEADLINE_WORKER_ID	Semua tindakan
DEADLINE_QUEUE_ID	Semua tindakan
DEADLINE_JOB_ID	Semua tindakan
DEADLINE_STEP_ID	Tindakan tugas
DEADLINE_SESSION_ID	Semua tindakan
DEADLINE_TASK_ID	Tindakan tugas

Nama variabel	Available
DEADLINE_SESSIONACTION_ID	Semua tindakan

Topik

Elemen template Job untuk bundel pekerjaan

Template pekerjaan mendefinisikan lingkungan runtime dan proses yang berjalan sebagai bagian dari pekerjaan Deadline Cloud. Anda dapat membuat parameter dalam template sehingga dapat digunakan untuk membuat pekerjaan yang hanya berbeda dalam nilai input, seperti fungsi dalam bahasa pemrograman.

Saat Anda mengirimkan pekerjaan ke Deadline Cloud, itu berjalan di lingkungan antrian apa pun yang diterapkan ke antrian. Lingkungan antrian dibangun menggunakan spesifikasi lingkungan eksternal Open Job Description (OpenJD). Untuk detailnya, lihat template Lingkungan di repositori OpenJD GitHub .

Untuk pengenalan yang membuat pekerjaan dengan template pekerjaan OpenJD, lihat Pengantar untuk membuat pekerjaan di repositori GitHub OpenJD. Informasi tambahan dapat ditemukan di Bagaimana pekerjaan dijalankan. Ada contoh template pekerjaan di dalam direktori GitHub repositori OpenJD. `samples`

Anda dapat menentukan template pekerjaan dalam format YAMAL (`template.yaml`) atau format JSON (`template.json`). Contoh di bagian ini ditampilkan dalam format YAMAL.

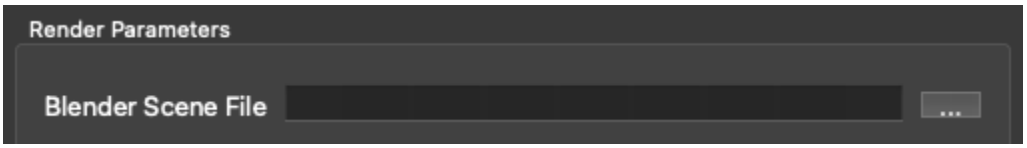
Misalnya, template pekerjaan untuk `blender_render` sampel mendefinisikan parameter input `BlenderSceneFile` sebagai jalur file:

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
```

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

`userInterface`Properti mendefinisikan perilaku antarmuka pengguna yang dihasilkan secara otomatis untuk kedua baris perintah menggunakan `deadline bundle gui-submit` perintah dan dalam plugin pengiriman pekerjaan untuk aplikasi seperti Autodesk Maya.

Dalam contoh ini, widget UI untuk memasukkan nilai untuk `BlenderSceneFile` parameter adalah dialog pemilihan file yang hanya menampilkan file. `.blend`



Untuk lebih banyak contoh penggunaan `userInteface` elemen, lihat contoh gui_control_showcase di repositori pada. deadline-cloud-samples GitHub

`dataFlow`Properti `objectType` dan mengontrol perilaku lampiran pekerjaan saat Anda mengirimkan pekerjaan dari bundel pekerjaan. Dalam hal ini, `objectType: FILE` dan `dataFlow:IN` berarti bahwa nilai `BlenderSceneFile` adalah file input untuk lampiran pekerjaan.

Sebaliknya, definisi `OutputDir` parameter memiliki `objectType: DIRECTORY` dan`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.
```

Nilai `OutputDir` parameter digunakan oleh lampiran pekerjaan sebagai direktori tempat pekerjaan menulis file output.

Untuk informasi selengkapnya tentang `dataFlow` properti `objectType` dan properti, lihat JobPathParameterDefinitionspesifikasi Open Job Description

Contoh template `blender_render` pekerjaan lainnya mendefinisikan alur kerja pekerjaan sebagai langkah tunggal dengan setiap frame dalam animasi yang dirender sebagai tugas terpisah:

```
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}}
```

Misalnya, jika nilai `Frames` parameternya`1-10`, ia mendefinisikan 10 tugas. Masing-masing memiliki tugas memiliki nilai yang berbeda untuk `Frame` parameter. Untuk menjalankan tugas:

1. Semua referensi variabel dalam `data` properti file tertanam diperluas, misalnya`--render-frame 1`.
2. Isi `data` properti ditulis ke file di direktori kerja sesi pada disk.
3. onRunPerintah tugas menyelesaikan `bash` *`location of embedded file`* dan kemudian berjalan.

Untuk informasi selengkapnya tentang file yang disematkan, sesi, dan lokasi yang dipetakan jalur, lihat Bagaimana pekerjaan dijalankan dalam spesifikasi Open Job Description.

Ada lebih banyak contoh template pekerjaan di repositori deadline-cloud-samples/job_bundles, serta sampel template yang disediakan dengan spesifikasi Open Job Descriptions.

Pembagian tugas untuk templat pekerjaan

Task chunking memungkinkan Anda mengelompokkan beberapa tugas ke dalam satu unit kerja yang disebut chunk. Dalam pekerjaan render, misalnya, ini berarti Deadline Cloud dapat mengirimkan beberapa frame bersama-sama, bukan satu frame per permintaan perintah. Ini mengurangi overhead aplikasi awal untuk setiap tugas dan mempersingkat total runtime pekerjaan. Untuk detailnya, lihat Menjalankan beberapa frame sekaligus di wiki OpenJD.

OpenJD mendukung ekstensi yang menambahkan fitur opsional ke template pekerjaan. Chunking tugas diaktifkan dengan menambahkan ekstensi. `TASK_CHUNKING` Untuk menggunakan chunking, tambahkan ekstensi ke template pekerjaan Anda dan gunakan tipe parameter `CHUNK[INT]` tugas. Kirim pekerjaan yang terpotong menggunakan perintah yang sama`deadline bundle submit`. Misalnya, template pekerjaan berikut merender frame dalam potongan 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
```

Dalam contoh ini, Deadline Cloud membagi 100 frame menjadi potongan-potongan seperti`1-10`,`11-20`, dan seterusnya. `{{Task.Param.Frame}}`Variabel meluas ke ekspresi rentang seperti`1-10`. Karena `rangeConstraint` diatur ke`CONTIGUOUS`, rentang selalu dalam `start-end` format. Skrip mem-parsing rentang ini dan meneruskan frame awal dan akhir ke Blender menggunakan `-e` opsi `-s` dan dengan`--render-anim`.

`chunks`Properti mendukung bidang-bidang berikut:

- `defaultTaskCount`— (Wajib) Berapa banyak tugas untuk digabungkan menjadi satu bagian. Nilai maksimumnya adalah 150.
- `rangeConstraint`— (Wajib) Jika`CONTIGUOUS`, potongan selalu merupakan rentang yang berdekatan seperti. `1-10` Jika`NONCONTIGUOUS`, potongan bisa menjadi set sewenang-wenang seperti. `1,3,7-10`
- `targetRuntimeSeconds`— (Opsional) Target runtime dalam hitungan detik untuk setiap potongan. Deadline Cloud dapat secara dinamis menyesuaikan ukuran potongan untuk mendekati target ini setelah beberapa potongan selesai.

Untuk contoh potongan tugas lainnya, termasuk contoh dasar dan Blender dengan potongan yang berdekatan dan tidak bersebelahan, lihat tugas memotong sampel di repositori sampel Deadline Cloud pada. GitHub

> **Persyaratan armada yang dikelola pelanggan**
>
> Task chunking memerlukan versi agen pekerja yang kompatibel. Jika Anda menggunakan armada yang dikelola pelanggan, pastikan agen pekerja Anda diperbarui sebelum mengirimkan pekerjaan dengan chunking. Armada yang dikelola layanan selalu menggunakan versi agen pekerja yang kompatibel.

> **Mengunduh output untuk pekerjaan yang terpotong**
>
> Saat Anda mengunduh output untuk satu tugas dalam pekerjaan yang dipotong, Deadline Cloud mengunduh output untuk seluruh potongan. Misalnya, jika frame 1-10 diproses bersama, mengunduh output untuk frame 3 mencakup semua frame 1-10. Fitur ini membutuhkan `deadline-cloud` versi 0.53.3 atau yang lebih baru.

Nilai parameter elemen untuk bundel pekerjaan

Anda dapat menggunakan file parameter untuk mengatur nilai beberapa parameter pekerjaan di template pekerjaan atau argumen permintaan CreateJoboperasi dalam bundel pekerjaan sehingga Anda tidak perlu menetapkan nilai saat mengirimkan pekerjaan. UI untuk pengiriman pekerjaan memungkinkan Anda untuk mengubah nilai-nilai ini.

Anda dapat menentukan template pekerjaan dalam format YAMAL (`parameter_values.yaml`) atau format JSON (`parameter_values.json`). Contoh di bagian ini ditampilkan dalam format YAMAL.

Di YAMAL, format file adalah:

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

Setiap elemen `parameterValues` daftar harus salah satu dari yang berikut:

- Parameter pekerjaan didefinisikan dalam template pekerjaan.
- Parameter pekerjaan yang ditentukan dalam lingkungan antrian untuk antrian yang Anda kirimkan pekerjaan ke..
- Parameter khusus diteruskan ke `CreateJob` operasi saat membuat pekerjaan.
  - `deadline:priority`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter prioritas.
  - `deadline:targetTaskRunStatus`— Nilai harus berupa string. Itu diteruskan ke `CreateJob` operasi sebagai parameter targetTaskRunStatus.
  - `deadline:maxFailedTasksCount`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter maxFailedTasksCount.
  - `deadline:maxRetriesPerTask`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter maxRetriesPerTugas.
  - `deadline:maxWorkercount`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai maxWorkerCountparameter.

Template pekerjaan selalu merupakan template daripada pekerjaan tertentu untuk dijalankan. File nilai parameter memungkinkan bundel pekerjaan untuk bertindak sebagai templat jika beberapa parameter tidak memiliki nilai yang ditentukan dalam file ini, atau sebagai pengiriman pekerjaan tertentu jika semua parameter memiliki nilai.

Misalnya, sampel blender_render tidak memiliki file parameter dan template pekerjaannya mendefinisikan parameter tanpa nilai default. Template ini harus digunakan sebagai template untuk membuat pekerjaan. Setelah Anda membuat pekerjaan menggunakan bundel pekerjaan ini, Deadline Cloud menulis bundel pekerjaan baru ke direktori riwayat pekerjaan.

Misalnya, ketika Anda mengirimkan pekerjaan dengan perintah berikut:

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

Bundel pekerjaan baru berisi `parameter_values.yaml` file yang berisi parameter yang ditentukan:

```
% 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
```

Anda dapat membuat pekerjaan yang sama dengan perintah berikut:

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

> **Note**
>
> Paket pekerjaan yang Anda kirimkan disimpan ke direktori riwayat pekerjaan Anda. Anda dapat menemukan lokasi direktori tersebut dengan perintah berikut:
>
> ```
> deadline config get settings.job_history_dir
> ```

Elemen referensi aset untuk bundel pekerjaan

Anda dapat menggunakan lampiran pekerjaan Deadline Cloud untuk mentransfer file bolak-balik antara workstation dan Deadline Cloud. File referensi aset mencantumkan file input dan direktori, serta direktori keluaran untuk lampiran Anda. Jika Anda tidak mencantumkan semua file dan direktori dalam file ini, Anda dapat memilihnya saat Anda mengirimkan pekerjaan dengan `deadline bundle gui-submit` perintah.

File ini tidak berpengaruh jika Anda tidak menggunakan lampiran pekerjaan.

Anda dapat menentukan template pekerjaan dalam format YAMAL (`asset_references.yaml`) atau format JSON (`asset_references.json`). Contoh di bagian ini ditampilkan dalam format YAMAL.

Di YAMAL, format file adalah:

```
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
```

Saat memilih file input atau output untuk diunggah ke Amazon S3, Deadline Cloud membandingkan jalur file dengan jalur yang tercantum dalam profil penyimpanan Anda. Setiap lokasi sistem file `SHARED` -type dalam profil penyimpanan mengabstraksi berbagi file jaringan yang dipasang di workstation dan host pekerja Anda. Deadline Cloud hanya mengunggah file yang tidak ada di salah satu pembagian file ini.

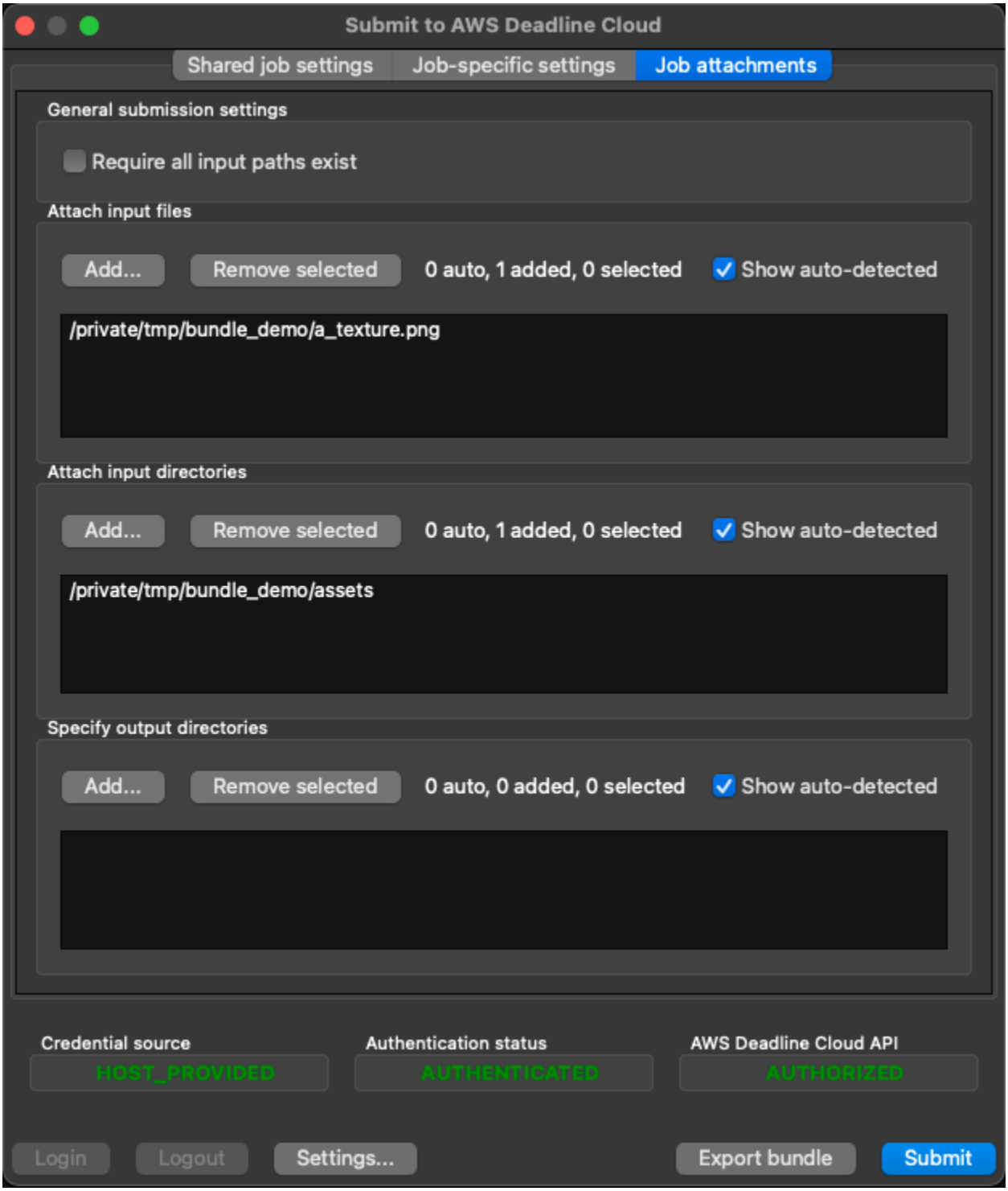
Untuk informasi selengkapnya tentang membuat dan menggunakan profil penyimpanan, lihat Penyimpanan bersama di Deadline Cloud di Panduan Pengguna Cloud AWS Deadline.

Example- File referensi aset yang dibuat oleh Deadline Cloud GUI

Gunakan perintah berikut untuk mengirimkan pekerjaan menggunakan sampel blender_render.

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

Tambahkan beberapa file tambahan ke pekerjaan di tab Lampiran Job:



Setelah mengirimkan pekerjaan, Anda dapat melihat `asset_references.yaml` file dalam bundel pekerjaan di direktori riwayat pekerjaan untuk melihat aset dalam file YAMAL:

```
% 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: []
```

Menggunakan file dalam pekerjaan Anda

Banyak pekerjaan yang Anda kirimkan ke AWS Deadline Cloud memiliki file input dan output. File input dan direktori output Anda mungkin terletak pada kombinasi sistem file bersama dan drive lokal. Pekerjaan perlu menemukan konten di lokasi tersebut. Deadline Cloud menyediakan dua fitur, lampiran pekerjaan dan profil penyimpanan yang bekerja sama untuk membantu pekerjaan Anda menemukan file yang mereka butuhkan.

Lampiran Job menawarkan beberapa manfaat

- Memindahkan file antar host menggunakan Amazon S3
- Transfer file dari stasiun kerja Anda ke host pekerja dan sebaliknya
- Tersedia untuk pekerjaan dalam antrian tempat Anda mengaktifkan fitur
- Terutama digunakan dengan armada yang dikelola layanan, tetapi juga kompatibel dengan armada yang dikelola pelanggan.

Gunakan profil penyimpanan untuk memetakan tata letak lokasi sistem file bersama di workstation dan host pekerja Anda. Pemetaan ini membantu pekerjaan Anda menemukan file dan direktori bersama ketika lokasinya berbeda antara workstation dan host pekerja Anda, seperti pengaturan lintas platform dengan workstation berbasis dan host pekerja Windows berbasis. Linux Peta profil penyimpanan konfigurasi sistem file Anda juga digunakan oleh lampiran pekerjaan untuk mengidentifikasi file yang diperlukan untuk berpindah antar host melalui Amazon S3.

Jika Anda tidak menggunakan lampiran pekerjaan, dan Anda tidak perlu memetakan ulang lokasi file dan direktori antara workstation dan host pekerja maka Anda tidak perlu memodelkan fileshares Anda dengan profil penyimpanan.

Topik

Contoh infrastruktur proyek

Untuk mendemonstrasikan penggunaan lampiran pekerjaan dan profil penyimpanan, siapkan lingkungan pengujian dengan dua proyek terpisah. Anda dapat menggunakan konsol Deadline Cloud untuk membuat sumber daya pengujian.

1. Jika Anda belum melakukannya, buat peternakan uji. Untuk membuat peternakan, ikuti prosedur di Buat peternakan.
2. Buat dua antrian untuk pekerjaan di masing-masing dari dua proyek. Untuk membuat antrian, ikuti prosedur di Buat antrian.

    a. Buat antrian pertama yang disebut**`Q1`**. Gunakan konfigurasi berikut, gunakan default untuk semua item lainnya.

    - Untuk lampiran pekerjaan, pilih Buat bucket Amazon S3 baru.
    - Pilih Aktifkan asosiasi dengan armada yang dikelola pelanggan.
    - Untuk menjalankan sebagai pengguna, masukkan **`jobuser`** untuk pengguna dan grup POSIX.
    - Untuk peran layanan antrian, buat peran baru bernama **`AssetDemoFarm-Q1-Role`**
    - Kosongkan kotak centang lingkungan antrian conda default.

    b. Buat antrian kedua yang disebut**`Q2`**. Gunakan konfigurasi berikut, gunakan default untuk semua item lainnya.

    - Untuk lampiran pekerjaan, pilih Buat bucket Amazon S3 baru.
    - Pilih Aktifkan asosiasi dengan armada yang dikelola pelanggan.
    - Untuk menjalankan sebagai pengguna, masukkan **`jobuser`** untuk pengguna dan grup POSIX.
    - Untuk peran layanan antrian, buat peran baru bernama **`AssetDemoFarm-Q2-Role`**
    - Kosongkan kotak centang lingkungan antrian conda default.

3. Buat satu armada yang dikelola pelanggan yang menjalankan pekerjaan dari kedua antrian. Untuk membuat armada, ikuti prosedur di Buat armada yang dikelola pelanggan. Gunakan konfigurasi berikut:

- Untuk Nama, gunakan**`DemoFleet`**.
- Untuk jenis Armada pilih Customer managed
- Untuk peran layanan Armada, buat peran baru bernama AssetDemoFarm-Fleet-Role.
- Jangan mengaitkan armada dengan antrian apa pun.

Lingkungan pengujian mengasumsikan bahwa ada tiga sistem file yang dibagi antara host menggunakan berbagi file jaringan. Dalam contoh ini, lokasi memiliki nama-nama berikut:

- `FSCommon`- berisi aset pekerjaan masukan yang umum untuk kedua proyek.
- `FS1`- berisi input dan output aset pekerjaan untuk proyek 1.
- `FS2`- berisi input dan output aset pekerjaan untuk proyek 2.

Lingkungan pengujian juga mengasumsikan bahwa ada tiga workstation, sebagai berikut:

- `WSAll`- Workstation Linux berbasis yang digunakan oleh pengembang untuk semua proyek. Lokasi sistem file bersama adalah:
  - `FSCommon`: `/shared/common`
  - `FS1`: `/shared/projects/project1`
  - `FS2`: `/shared/projects/project2`
- `WS1`- Sebuah workstation Windows berbasis yang digunakan untuk proyek 1. Lokasi sistem file bersama adalah:
  - `FSCommon`: `S:\`
  - `FS1`: `Z:\`
  - `FS2`: Tidak tersedia
- `WS1`- Workstation macOS berbasis yang digunakan untuk proyek 2. Lokasi sistem file bersama adalah:
  - `FSCommon`: `/Volumes/common`
  - `FS1`: Tidak tersedia
  - `FS2`: `/Volumes/projects/project2`

Terakhir, tentukan lokasi sistem file bersama untuk pekerja di armada Anda. Contoh-contoh berikut mengacu pada konfigurasi ini sebagai`WorkerConfig`. Lokasi bersama adalah:

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

Anda tidak perlu menyiapkan sistem file bersama, workstation, atau pekerja yang cocok dengan konfigurasi ini. Lokasi bersama tidak perlu ada untuk demonstrasi.

Profil penyimpanan dan pemetaan jalur

Gunakan profil penyimpanan untuk memodelkan sistem file di workstation dan host pekerja Anda. Setiap profil penyimpanan menjelaskan sistem operasi dan tata letak sistem file dari salah satu konfigurasi sistem Anda. Topik ini menjelaskan cara menggunakan profil penyimpanan untuk memodelkan konfigurasi sistem file host Anda sehingga Deadline Cloud dapat menghasilkan aturan pemetaan jalur untuk pekerjaan Anda, dan bagaimana aturan pemetaan jalur tersebut dihasilkan dari profil penyimpanan Anda.

Saat mengirimkan pekerjaan ke Deadline Cloud, Anda dapat memberikan ID profil penyimpanan opsional untuk pekerjaan tersebut. Profil penyimpanan ini menjelaskan sistem file workstation pengiriman. Ini menjelaskan konfigurasi sistem file asli yang digunakan jalur file dalam template pekerjaan.

Anda juga dapat mengaitkan profil penyimpanan dengan armada. Profil penyimpanan menjelaskan konfigurasi sistem file dari semua host pekerja di armada. Jika Anda memiliki pekerja dengan konfigurasi sistem file yang berbeda, pekerja tersebut harus ditugaskan ke armada yang berbeda di peternakan Anda.

Aturan pemetaan jalur menjelaskan bagaimana jalur harus dipetakan ulang dari cara mereka ditentukan dalam pekerjaan ke lokasi aktual jalur pada host pekerja. Deadline Cloud membandingkan konfigurasi sistem file yang dijelaskan dalam profil penyimpanan pekerjaan dengan profil penyimpanan armada yang menjalankan pekerjaan untuk mendapatkan aturan pemetaan jalur ini.

Topik

Model lokasi sistem file bersama dengan profil penyimpanan

Profil penyimpanan memodelkan konfigurasi sistem file dari salah satu konfigurasi host Anda. Ada empat konfigurasi host yang berbeda dalam infrastruktur proyek sampel. Dalam contoh ini Anda membuat profil penyimpanan terpisah untuk masing-masing. Anda dapat membuat profil penyimpanan menggunakan salah satu dari berikut ini:

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation sumber daya
- AWS konsol

Profil penyimpanan terdiri dari daftar lokasi sistem file yang masing-masing memberi tahu Deadline Cloud lokasi dan jenis lokasi sistem file yang relevan untuk pekerjaan yang dikirim dari atau dijalankan pada host. Profil penyimpanan seharusnya hanya memodelkan lokasi yang relevan untuk pekerjaan. Misalnya, `FSCommon` lokasi bersama terletak di workstation `WS1` di`S:\`, jadi lokasi sistem file yang sesuai adalah:

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

Gunakan perintah berikut untuk membuat profil penyimpanan untuk konfigurasi workstation `WS1``WS2`, `WS3` dan konfigurasi pekerja `WorkerConfig` menggunakan in AWS CLI: 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**
>
> Anda harus merujuk ke lokasi sistem file di profil penyimpanan Anda menggunakan nilai yang sama untuk `name` properti di semua profil penyimpanan di pertanian Anda. Deadline Cloud membandingkan nama untuk menentukan bahwa lokasi sistem file dari profil penyimpanan yang berbeda merujuk ke lokasi yang sama saat membuat aturan pemetaan jalur.

Konfigurasikan profil penyimpanan untuk armada

Anda dapat mengonfigurasi armada untuk menyertakan profil penyimpanan yang memodelkan lokasi sistem file pada semua pekerja di armada. Konfigurasi sistem file host dari semua pekerja dalam armada harus sesuai dengan profil penyimpanan armada mereka. Pekerja dengan konfigurasi sistem file yang berbeda harus berada dalam armada terpisah.

Untuk mengatur konfigurasi armada Anda agar menggunakan profil `WorkerConfig` penyimpanan, gunakan fitur AWS CLIin 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
    }
  }"
```

Konfigurasikan profil penyimpanan untuk antrian

Konfigurasi antrian mencakup daftar nama peka huruf besar/kecil dari lokasi sistem file bersama yang pekerjaan yang dikirimkan ke antrian memerlukan akses ke. misalnya, pekerjaan yang dikirimkan ke antrian Q1 memerlukan lokasi sistem file dan. `FSCommon` `FS1` Pekerjaan yang dikirimkan ke antrian Q2 memerlukan lokasi sistem file `FSCommon` dan`FS2`.

Untuk mengatur konfigurasi antrian agar memerlukan lokasi sistem file ini, gunakan skrip berikut:

```
# 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
```

Konfigurasi antrian juga mencakup daftar profil penyimpanan yang diizinkan yang berlaku untuk pekerjaan yang dikirimkan dan armada yang terkait dengan antrian tersebut. Hanya profil penyimpanan yang menentukan lokasi sistem file untuk semua lokasi sistem file yang diperlukan untuk antrian yang diizinkan dalam daftar antrian profil penyimpanan yang diizinkan.

Pekerjaan gagal jika Anda mengirimkannya dengan profil penyimpanan yang tidak ada dalam daftar profil penyimpanan yang diizinkan untuk antrian. Anda selalu dapat mengirimkan pekerjaan tanpa profil penyimpanan ke antrian. Konfigurasi workstation berlabel `WSAll` dan `WS1` keduanya memiliki lokasi sistem file yang diperlukan (`FSCommon`dan`FS1`) untuk antrian. `Q1` Mereka harus diizinkan untuk mengirimkan pekerjaan ke antrian. Demikian pula, konfigurasi workstation `WSAll` dan `WS2` memenuhi persyaratan untuk antrian. `Q2` Mereka harus diizinkan untuk mengirimkan pekerjaan ke antrian itu. Perbarui kedua konfigurasi antrian untuk memungkinkan pekerjaan dikirimkan dengan profil penyimpanan ini menggunakan skrip berikut:

```
# 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
```

Jika Anda menambahkan profil `WS2` penyimpanan ke daftar profil penyimpanan yang diizinkan untuk antrian, `Q1` itu gagal:

```
$ 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
```

Ini karena profil WS2 penyimpanan tidak berisi definisi untuk lokasi sistem file bernama antrian FS1 yang Q1 diperlukan.

Mengaitkan armada yang dikonfigurasi dengan profil penyimpanan yang tidak ada dalam daftar antrian profil penyimpanan yang diizinkan juga gagal. Contoh:

```
$ 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.
```

Untuk memperbaiki kesalahan, tambahkan profil penyimpanan yang diberi nama WorkerConfig ke daftar profil penyimpanan yang diizinkan untuk antrian Q1 dan antrian. Q2 Kemudian, kaitkan armada dengan antrian ini sehingga pekerja di armada dapat menjalankan pekerjaan dari kedua antrian.

```
# 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
```

Turunkan aturan pemetaan jalur dari profil penyimpanan

Aturan pemetaan jalur menjelaskan bagaimana jalur harus dipetakan ulang dari pekerjaan ke lokasi sebenarnya jalur pada host pekerja. Saat tugas dijalankan pada pekerja, profil penyimpanan dari pekerjaan tersebut dibandingkan dengan profil penyimpanan armada pekerja untuk mendapatkan aturan pemetaan jalur untuk tugas tersebut.

Deadline Cloud membuat aturan pemetaan untuk setiap lokasi sistem file yang diperlukan dalam konfigurasi antrian. Misalnya, pekerjaan yang dikirimkan dengan profil `WSAll` penyimpanan ke antrian `Q1` memiliki aturan pemetaan jalur:

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

Deadline Cloud membuat aturan untuk lokasi `FSComm` dan sistem `FS1` file, tetapi bukan lokasi sistem `FS2` file meskipun profil `WSAll` dan `WorkerConfig` penyimpanan ditentukan`FS2`. Ini karena daftar antrian `Q1` lokasi sistem file yang diperlukan adalah`["FSComm", "FS1"]`.

Anda dapat mengonfirmasi aturan pemetaan jalur yang tersedia untuk pekerjaan yang dikirimkan dengan profil penyimpanan tertentu dengan mengirimkan pekerjaan yang mencetak file aturan pemetaan jalur Open Job Description, lalu membaca log sesi setelah pekerjaan selesai:

```
# 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}}" ]
            }
          }
        }
      }
    ]
  }'
```

Jika Anda menggunakan Deadline Cloud CLI untuk mengirimkan pekerjaan, pengaturan `settings.storage_profile_id` konfigurasinya menetapkan profil penyimpanan yang akan dimiliki oleh pekerjaan yang dikirimkan dengan CLI. Untuk mengirimkan pekerjaan dengan profil `WSAll` penyimpanan, atur:

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

Untuk menjalankan pekerja yang dikelola pelanggan seolah-olah sedang berjalan di infrastruktur sampel, ikuti prosedur di Jalankan agen pekerja di Panduan Pengguna Cloud Tenggat Waktu untuk menjalankan pekerja. AWS CloudShell Jika Anda mengikuti instruksi tersebut sebelumnya, hapus `~/demoenv-persist` direktori `~/demoenv-logs` dan direktori terlebih dahulu. Juga, tetapkan nilai-nilai variabel `DEV_FARM_ID` dan `DEV_CMF_ID` lingkungan yang referensi arah sebagai berikut sebelum melakukannya:

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

Setelah pekerjaan berjalan, Anda dapat melihat aturan pemetaan jalur di file log pekerjaan:

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

Log berisi pemetaan untuk sistem `FSComm` file `FS1` dan file. Diformat ulang agar mudah dibaca, entri log terlihat seperti ini:

```
{
    "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"
        }
    ]
```

Anda dapat mengirimkan pekerjaan dengan profil penyimpanan yang berbeda untuk melihat bagaimana aturan pemetaan jalur berubah.

Gunakan lampiran pekerjaan untuk berbagi file

Gunakan lampiran pekerjaan untuk membuat file yang tidak ada di direktori bersama tersedia untuk pekerjaan Anda, dan untuk menangkap file output jika tidak ditulis ke direktori bersama. Lampiran Job menggunakan Amazon S3 untuk mengirim file antar host. File disimpan dalam bucket S3, dan Anda tidak perlu mengunggah file jika kontennya tidak berubah.

Anda harus menggunakan lampiran pekerjaan saat menjalankan pekerjaan pada armada yang dikelola layanan karena host tidak berbagi lokasi sistem file. Lampiran Job juga berguna dengan armada yang dikelola pelanggan ketika file input atau output pekerjaan disimpan pada sistem file jaringan bersama, seperti ketika bundel pekerjaan Anda berisi skrip shell atau Python.

Saat Anda mengirimkan paket pekerjaan dengan Deadline Cloud CLI atau pengirim Deadline Cloud, lampiran pekerjaan menggunakan profil penyimpanan pekerjaan dan lokasi sistem file yang diperlukan antrian untuk mengidentifikasi file input yang tidak ada di host pekerja dan harus diunggah ke Amazon S3 sebagai bagian dari pengiriman pekerjaan. Profil penyimpanan ini juga membantu Deadline Cloud mengidentifikasi file keluaran di lokasi host pekerja yang harus diunggah ke Amazon S3 sehingga tersedia untuk stasiun kerja Anda.

Contoh lampiran pekerjaan menggunakan konfigurasi profil pertanian, armada, antrian, dan penyimpanan dari dan. Contoh infrastruktur proyek Profil penyimpanan dan pemetaan jalur Anda harus melalui bagian-bagian itu sebelum yang satu ini.

Dalam contoh berikut, Anda menggunakan bundel pekerjaan sampel sebagai titik awal, lalu memodifikasinya untuk mengeksplorasi fungsionalitas lampiran pekerjaan. Paket Job adalah cara

terbaik bagi pekerjaan Anda untuk menggunakan lampiran pekerjaan. Mereka menggabungkan template pekerjaan Open Job Description dalam direktori dengan file tambahan yang mencantumkan file dan direktori yang dibutuhkan oleh pekerjaan menggunakan bundel pekerjaan. Untuk informasi selengkapnya tentang paket pekerjaan, lihatTemplat Open Job Description (OpenJD) untuk Deadline Cloud.

Mengirimkan file dengan pekerjaan

Dengan Deadline Cloud, Anda dapat mengaktifkan alur kerja pekerjaan untuk mengakses file input yang tidak tersedia di lokasi sistem file bersama di host pekerja. Lampiran Job memungkinkan rendering pekerjaan untuk mengakses file yang hanya berada di drive workstation lokal atau lingkungan armada yang dikelola layanan. Saat mengirimkan bundel pekerjaan, Anda dapat menyertakan daftar file input dan direktori yang diperlukan oleh pekerjaan. Deadline Cloud mengidentifikasi file yang tidak dibagikan ini, mengunggahnya dari mesin lokal ke Amazon S3, dan mengunduhnya ke host pekerja. Ini merampingkan proses mentransfer aset input ke render node, memastikan semua file yang diperlukan dapat diakses untuk eksekusi pekerjaan terdistribusi.

Anda dapat menentukan file untuk pekerjaan secara langsung di bundel pekerjaan, menggunakan parameter dalam template pekerjaan yang Anda berikan menggunakan variabel lingkungan atau skrip, dan menggunakan `assets_references` file pekerjaan. Anda dapat menggunakan salah satu metode ini atau kombinasi ketiganya. Anda dapat menentukan profil penyimpanan untuk bundel untuk pekerjaan sehingga hanya mengunggah file yang telah berubah di workstation lokal.

Bagian ini menggunakan contoh bundel pekerjaan GitHub untuk menunjukkan bagaimana Deadline Cloud mengidentifikasi file dalam pekerjaan Anda untuk diunggah, bagaimana file tersebut diatur di Amazon S3, dan bagaimana file tersebut tersedia untuk host pekerja yang memproses pekerjaan Anda.

Topik

- Bagaimana Deadline Cloud mengunggah file ke Amazon S3
- Bagaimana Deadline Cloud memilih file yang akan diunggah
- Bagaimana pekerjaan menemukan file input lampiran pekerjaan

Bagaimana Deadline Cloud mengunggah file ke Amazon S3

Contoh ini menunjukkan bagaimana Deadline Cloud mengunggah file dari workstation atau host pekerja Anda ke Amazon S3 sehingga file tersebut dapat dibagikan. Ini menggunakan bundel pekerjaan sampel dari GitHub dan Deadline Cloud CLI untuk mengirimkan pekerjaan.

Mulailah dengan mengkloning GitHubrepositori sampel Deadline Cloud ke AWS CloudShelllingkungan Anda, lalu salin bundel `job_attachments_devguide` pekerjaan ke direktori home Anda:

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

Instal Deadline Cloud CLI untuk mengirimkan bundel pekerjaan:

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

Bundel `job_attachments_devguide` pekerjaan memiliki satu langkah dengan tugas yang menjalankan skrip bash shell yang lokasi sistem filenya diteruskan sebagai parameter pekerjaan. Definisi parameter pekerjaan adalah:

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

`IN`Nilai `dataFlow` properti memberi tahu lampiran pekerjaan bahwa nilai `ScriptFile` parameter adalah masukan ke pekerjaan. Nilai `default` properti adalah lokasi relatif ke direktori bundel pekerjaan, tetapi juga bisa menjadi jalur absolut. Definisi parameter ini mendeklarasikan `script.sh` file dalam direktori bundel pekerjaan sebagai file input yang diperlukan agar pekerjaan dapat dijalankan.

Selanjutnya, pastikan bahwa Deadline Cloud CLI tidak memiliki profil penyimpanan yang dikonfigurasi kemudian kirimkan pekerjaan ke antrian: 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/
```

Output dari Deadline Cloud CLI setelah perintah ini dijalankan terlihat seperti:

```
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
```

Saat Anda mengirimkan pekerjaan, Deadline Cloud pertama-tama melakukan hash `script.sh` file dan kemudian mengunggahnya ke Amazon S3.

Deadline Cloud memperlakukan bucket S3 sebagai penyimpanan yang dapat dialamatkan konten. File diunggah ke objek S3. Nama objek berasal dari hash dari isi file. Jika dua file memiliki konten yang identik, mereka memiliki nilai hash yang sama terlepas dari di mana file tersebut berada atau apa namanya. Penyimpanan yang dapat dialamatkan konten ini memungkinkan Deadline Cloud untuk menghindari mengunggah file jika sudah tersedia.

Anda dapat menggunakan AWS CLI untuk melihat objek yang diunggah ke 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
```

Dua objek diunggah ke S3:

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— Isi dari`script.sh`. Nilai `87cb19095dd5d78fcaf56384ef0e6241` dalam kunci objek adalah hash dari isi file, dan ekstensi `xxh128` menunjukkan bahwa nilai hash dihitung sebagai 128 bit xxhash.
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— Objek manifes untuk pengajuan pekerjaan. Nilai`<farm-id>`,`<queue-id>`, dan `<guid>` merupakan pengidentifikasi pertanian Anda, pengidentifikasi antrian, dan nilai heksidesimal acak. Nilai `a1d221c7fd97b08175b3872a37428e8c` dalam contoh ini adalah nilai hash yang dihitung dari string`/home/cloudshell-user/job_attachments_devguide`, direktori tempat `script.sh` berada.

Objek manifes berisi informasi untuk file input pada jalur root tertentu yang diunggah ke S3 sebagai bagian dari pengiriman pekerjaan. Unduh file manifes ini (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`). Isinya mirip dengan:

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

Ini menunjukkan bahwa file `script.sh` telah diunggah, dan hash dari konten file itu. `87cb19095dd5d78fcaf56384ef0e6241` Nilai hash ini cocok dengan nilai dalam nama `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128` objek. Ini digunakan oleh Deadline Cloud untuk mengetahui objek mana yang akan diunduh untuk konten file ini.

Skema lengkap untuk file ini tersedia di GitHub.

Bila Anda menggunakan CreateJob operasi, Anda dapat mengatur lokasi objek manifes. Anda dapat menggunakan GetJoboperasi untuk melihat lokasi:

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

Bagaimana Deadline Cloud memilih file yang akan diunggah

File dan direktori yang dipertimbangkan oleh lampiran pekerjaan untuk diunggah ke Amazon S3 sebagai input ke pekerjaan Anda adalah:

- Nilai-nilai dari semua parameter pekerjaan `PATH` -type didefinisikan dalam template pekerjaan bundel pekerjaan dengan `dataFlow` nilai `IN` atau`INOUT`.
- File dan direktori terdaftar sebagai input dalam file referensi aset bundel pekerjaan.

Jika Anda mengirimkan pekerjaan tanpa profil penyimpanan, semua file yang dipertimbangkan untuk diunggah akan diunggah. Jika Anda mengirimkan pekerjaan dengan profil penyimpanan, file tidak akan diunggah ke Amazon S3 jika mereka berada di lokasi sistem file tipe penyimpanan profil `SHARED` penyimpanan yang juga diperlukan lokasi sistem file untuk antrian. Lokasi ini diharapkan tersedia di host pekerja yang menjalankan pekerjaan, jadi tidak perlu mengunggahnya ke S3.

Dalam contoh ini, Anda membuat lokasi sistem `SHARED` file `WSAll` di CloudShell lingkungan AWS Anda dan kemudian menambahkan file ke lokasi sistem file tersebut. Gunakan perintah berikut ini.

```
# 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
```

Selanjutnya, tambahkan file referensi aset ke bundel pekerjaan yang menyertakan semua file yang Anda buat sebagai input untuk pekerjaan tersebut. Gunakan perintah berikut ini.

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

Selanjutnya, konfigurasikan Deadline Cloud CLI untuk mengirimkan pekerjaan dengan `WSAll` profil penyimpanan, lalu kirimkan bundel pekerjaan:

```

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

deadline config set settings.storage_profile_id $WSALL_ID

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

Deadline Cloud mengunggah dua file ke Amazon S3 saat Anda mengirimkan pekerjaan. Anda dapat mengunduh objek manifes untuk pekerjaan dari S3 untuk melihat file yang diunggah:

```
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
```

Dalam contoh ini, ada satu file manifes dengan konten berikut:

```
{
    "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
}
```

Gunakan GetJob operasi untuk manifes untuk melihat bahwa `rootPath` adalah "/".

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

Jalur root untuk kumpulan file input selalu merupakan subpath umum terpanjang dari file-file tersebut. Jika pekerjaan Anda dikirim dari Windows sebagai gantinya dan ada file input tanpa subpath umum karena mereka berada di drive yang berbeda, Anda melihat jalur root terpisah pada setiap drive. Jalur dalam manifes selalu relatif terhadap jalur root manifes, sehingga file input yang diunggah adalah:

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— File skrip dalam bundel pekerjaan.

- `/shared/projects/project2/file.txt`— File di lokasi sistem `SHARED` file di profil `WSAll` penyimpanan yang tidak ada dalam daftar lokasi sistem file yang diperlukan untuk antrian`Q1`.

File di lokasi sistem file `FSCommon` (`/shared/common/file.txt`) dan `FS1` (`/shared/projects/project1/file.txt`) tidak ada dalam daftar. Ini karena lokasi sistem file tersebut berada `SHARED` di profil `WSAll` penyimpanan dan keduanya berada dalam daftar lokasi sistem file yang diperlukan dalam antrian`Q1`.

Anda dapat melihat lokasi sistem file yang dipertimbangkan `SHARED` untuk pekerjaan yang dikirimkan dengan profil penyimpanan tertentu dengan GetStorageProfileForQueue operasi. Untuk kueri profil penyimpanan `WSAll` untuk antrian `Q1` gunakan perintah berikut:

```
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
```

Bagaimana pekerjaan menemukan file input lampiran pekerjaan

Agar pekerjaan dapat menggunakan file yang diunggah Deadline Cloud ke Amazon S3 menggunakan lampiran pekerjaan, pekerjaan Anda memerlukan file-file tersebut yang tersedia melalui sistem file di host pekerja. Saat sesi untuk pekerjaan Anda berjalan di host pekerja, Deadline Cloud mengunduh file input untuk pekerjaan tersebut ke direktori sementara di drive lokal host pekerja dan menambahkan aturan pemetaan jalur untuk setiap jalur root pekerjaan ke lokasi sistem filenya di drive lokal.

Untuk contoh ini, mulai agen pekerja Deadline Cloud di CloudShell tab AWS. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan, lalu hapus log pekerjaan dari direktori log:

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

Skrip berikut memodifikasi bundel pekerjaan untuk menampilkan semua file di direktori kerja sementara sesi dan isi file aturan pemetaan jalur, dan kemudian mengirimkan pekerjaan dengan bundel yang dimodifikasi:

```
# 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/
```

Anda dapat melihat log pekerjaan yang dijalankan setelah dijalankan oleh pekerja di AWS CloudShell lingkungan Anda:

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

Log menunjukkan bahwa hal pertama yang terjadi dalam sesi adalah dua file input untuk pekerjaan yang diunduh ke pekerja:

```
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.
```

Berikutnya adalah output dari `script.sh` run by the job:

- File masukan yang diunggah saat pekerjaan dikirimkan terletak di bawah direktori yang namanya dimulai dengan “assetroot” di direktori sementara sesi.
- Jalur file input telah dipindahkan relatif ke direktori “assetroot” alih-alih relatif terhadap jalur root untuk manifes input pekerjaan (). `"/"`
- File aturan pemetaan jalur berisi aturan tambahan yang memetakan ulang `"/"` ke jalur absolut direktori “assetroot”.

Contoh:

```
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**
>
> Jika pekerjaan yang Anda kirimkan memiliki beberapa manifes dengan jalur root yang berbeda, ada direktori bernama “assetroot” yang berbeda untuk setiap jalur root.

Jika Anda perlu mereferensikan lokasi sistem file yang dipindahkan dari salah satu file input, direktori, atau lokasi sistem file Anda, Anda dapat memproses file aturan pemetaan jalur dalam pekerjaan Anda dan melakukan pemetaan ulang sendiri, atau menambahkan parameter PATH jenis pekerjaan ke template pekerjaan di bundel pekerjaan Anda dan meneruskan nilai yang Anda perlukan untuk memetakan ulang sebagai nilai parameter itu. Misalnya, contoh berikut memodifikasi bundel pekerjaan untuk memiliki salah satu parameter pekerjaan ini dan kemudian mengirimkan pekerjaan dengan lokasi sistem file `/shared/projects/project2` sebagai nilainya:

```
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
```

File log untuk menjalankan pekerjaan ini berisi outputnya:

```
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
```

Mendapatkan file output dari pekerjaan

Contoh ini menunjukkan bagaimana Deadline Cloud mengidentifikasi file output yang dihasilkan oleh pekerjaan Anda, memutuskan apakah akan mengunggah file tersebut ke Amazon S3, dan bagaimana Anda bisa mendapatkan file output tersebut di workstation Anda.

Gunakan bundel `job_attachments_devguide_output` pekerjaan alih-alih bundel `job_attachments_devguide` pekerjaan untuk contoh ini. Mulailah dengan membuat salinan bundel di AWS CloudShell lingkungan Anda dari tiruan repositori sampel GitHub Deadline Cloud:

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

Perbedaan penting antara bundel pekerjaan ini dan bundel `job_attachments_devguide` pekerjaan adalah penambahan parameter pekerjaan baru di templat pekerjaan:

```
...
parameterDefinitions:
```

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

`dataFlow`Properti parameter memiliki nilai`OUT`. Deadline Cloud menggunakan nilai parameter `dataFlow` pekerjaan dengan nilai `OUT` atau `INOUT` sebagai output dari pekerjaan Anda. Jika lokasi sistem file diteruskan sebagai nilai ke jenis parameter pekerjaan ini dipetakan ulang ke lokasi sistem file lokal pada pekerja yang menjalankan pekerjaan, maka Deadline Cloud akan mencari file baru di lokasi dan mengunggahnya ke Amazon S3 sebagai output pekerjaan.

Untuk melihat cara kerjanya, pertama-tama mulai agen pekerja Deadline Cloud di AWS CloudShell tab. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan. Kemudian hapus log pekerjaan dari direktori log:

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

Selanjutnya, kirimkan pekerjaan dengan bundel pekerjaan ini. Setelah pekerja berjalan dalam CloudShell proses Anda, lihat log:

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

deadline config set settings.storage_profile_id $WSALL_ID

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

Log menunjukkan bahwa file terdeteksi sebagai output dan diunggah ke 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.
```

Log juga menunjukkan bahwa Deadline Cloud membuat objek manifes baru di bucket Amazon S3 yang dikonfigurasi untuk digunakan oleh lampiran pekerjaan pada antrian. Q1 Nama objek manifes berasal dari pertanian, antrian, pekerjaan, langkah, tugas, stempel waktu, dan `sessionaction` pengidentifikasi tugas yang menghasilkan output. Unduh file manifes ini untuk melihat di mana Deadline Cloud menempatkan file output untuk tugas ini:

```
# 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 .
```

Manifes terlihat seperti:

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
```

```
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

Ini menunjukkan bahwa konten file output disimpan ke Amazon S3 dengan cara yang sama seperti file input pekerjaan disimpan. Mirip dengan file input, file output disimpan dalam S3 dengan nama objek yang berisi hash file dan awalan. `DeadlineCloud/Data`

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

Anda dapat mengunduh output pekerjaan ke workstation Anda menggunakan monitor Deadline Cloud atau Deadline Cloud CLI:

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

Nilai parameter `OutputDir` pekerjaan dalam pekerjaan yang dikirimkan adalah `./output_dir`, sehingga output diunduh ke direktori yang disebut `output_dir` dalam direktori bundel pekerjaan. Jika Anda menentukan jalur absolut atau lokasi relatif yang berbeda sebagai nilainya`OutputDir`, maka file output akan diunduh ke lokasi itu sebagai gantinya.

```
$ 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)
```

Menggunakan file dari langkah dalam langkah dependen

Contoh ini menunjukkan bagaimana satu langkah dalam pekerjaan dapat mengakses output dari langkah yang bergantung pada pekerjaan yang sama.

Untuk membuat output dari satu langkah tersedia untuk yang lain, Deadline Cloud menambahkan tindakan tambahan ke sesi untuk mengunduh output tersebut sebelum menjalankan tugas dalam sesi. Anda memberi tahu langkah mana untuk mengunduh output dengan mendeklarasikan langkah-langkah tersebut sebagai dependensi dari langkah yang perlu menggunakan output.

Gunakan bundel `job_attachments_devguide_output` pekerjaan untuk contoh ini. Mulailah dengan membuat salinan di AWS CloudShell lingkungan Anda dari klon repositori sampel GitHub Deadline Cloud. Ubah untuk menambahkan langkah dependen yang hanya berjalan setelah langkah yang ada dan menggunakan output langkah itu:

```
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
```

Pekerjaan yang dibuat dengan bundel pekerjaan yang dimodifikasi ini berjalan sebagai dua sesi terpisah, satu untuk tugas di langkah “Langkah” dan kemudian yang kedua untuk tugas di langkah "DependentStep”.

Pertama mulai agen pekerja Deadline Cloud di CloudShell tab. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan, lalu hapus log pekerjaan dari direktori log:

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

Selanjutnya, kirimkan pekerjaan menggunakan bundel `job_attachments_devguide_output` pekerjaan yang dimodifikasi. Tunggu sampai selesai berjalan pada pekerja di CloudShell lingkungan Anda. Lihatlah log untuk dua sesi:

```
# 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-*
```

Dalam log sesi untuk tugas di langkah bernama`DependentStep`, ada dua tindakan unduhan terpisah yang dijalankan:

```
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.
```

Tindakan pertama mengunduh `script.sh` file yang digunakan oleh langkah bernama “Langkah.” Tindakan kedua mengunduh output dari langkah itu. Deadline Cloud menentukan file mana yang akan diunduh dengan menggunakan manifes keluaran yang dihasilkan oleh langkah tersebut sebagai manifes masukan.

Di akhir log yang sama, Anda dapat melihat output dari langkah bernama "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
```

Buat batas sumber daya untuk pekerjaan

Pekerjaan yang dikirimkan ke Deadline Cloud mungkin bergantung pada sumber daya yang dibagi di antara beberapa pekerjaan. Misalnya, sebuah peternakan mungkin memiliki lebih banyak pekerja daripada lisensi mengambang untuk sumber daya tertentu. Atau server file bersama mungkin hanya dapat melayani data ke sejumlah pekerja terbatas pada saat yang bersamaan. Dalam beberapa kasus, satu atau lebih pekerjaan dapat mengklaim semua sumber daya ini, menyebabkan kesalahan karena sumber daya yang tidak tersedia ketika pekerja baru mulai.

Untuk membantu mengatasi hal ini, Anda dapat menggunakan batas untuk sumber daya terbatas ini. Deadline Cloud memperhitungkan ketersediaan sumber daya terbatas dan menggunakan informasi tersebut untuk memastikan bahwa sumber daya tersedia saat pekerja baru memulai sehingga pekerjaan memiliki kemungkinan gagal yang lebih rendah karena sumber daya yang tidak tersedia.

Batas dibuat untuk seluruh peternakan. Pekerjaan yang dikirimkan ke antrian hanya dapat memperoleh batas yang terkait dengan antrian. Jika Anda menentukan batas untuk pekerjaan yang tidak terkait dengan antrian, pekerjaan tersebut tidak kompatibel dan tidak akan berjalan.

Untuk menggunakan batas, Anda

- Buat batas

- Kaitkan batas dan antrian
- Kirim pekerjaan yang membutuhkan batasan

> **Note**
>
> Jika Anda menjalankan pekerjaan yang memiliki sumber daya terbatas dalam antrian yang tidak terkait dengan batas, pekerjaan itu dapat menghabiskan semua sumber daya. Jika Anda memiliki sumber daya terbatas, pastikan bahwa semua langkah dalam pekerjaan dalam antrian yang menggunakan sumber daya dikaitkan dengan batas.

Untuk batas yang ditentukan di peternakan, terkait dengan antrian, dan ditentukan dalam pekerjaan, salah satu dari empat hal dapat terjadi:

- Jika Anda membuat batas, kaitkan dengan antrian, dan tentukan batas dalam templat pekerjaan, pekerjaan akan berjalan dan hanya menggunakan sumber daya yang ditentukan dalam batas.
- Jika Anda membuat batas, tentukan dalam templat pekerjaan, tetapi jangan mengaitkan batas dengan antrian, pekerjaan ditandai tidak kompatibel dan tidak akan berjalan.
- Jika Anda membuat batas, jangan kaitkan dengan antrian, dan jangan tentukan batas dalam templat pekerjaan, pekerjaan berjalan tetapi tidak menggunakan batas.
- Jika Anda tidak menggunakan batas sama sekali, pekerjaan berjalan.

Jika Anda mengaitkan batas ke beberapa antrian, antrian berbagi sumber daya yang dibatasi oleh batas. Misalnya, jika Anda membuat batas 100, dan satu antrian menggunakan 60 sumber daya, antrian lain hanya dapat menggunakan 40 sumber daya. Ketika sumber daya dirilis, itu dapat diambil oleh tugas dari antrian apa pun.

Deadline Cloud menyediakan dua AWS CloudFormation metrik untuk membantu Anda memantau sumber daya yang disediakan oleh batas. Anda dapat memantau jumlah sumber daya saat ini yang digunakan dan jumlah maksimum sumber daya yang tersedia dalam batas. Untuk informasi selengkapnya, lihat Metrik batas sumber daya di Panduan Pengembang Cloud Batas Waktu.

Anda menerapkan batas untuk langkah pekerjaan dalam template pekerjaan. Saat Anda menentukan nama persyaratan jumlah batas di `amounts` bagian langkah dan batas yang sama `amountRequirementName` dikaitkan dengan antrian pekerjaan, tugas yang dijadwalkan untuk langkah ini dibatasi oleh batas sumber daya. `hostRequirements`

Jika sebuah langkah membutuhkan sumber daya yang dibatasi oleh batas yang tercapai, tugas dalam langkah itu tidak akan diambil oleh pekerja tambahan.

Anda dapat menerapkan lebih dari satu batas untuk langkah pekerjaan. Misalnya, jika langkah menggunakan dua lisensi perangkat lunak yang berbeda, Anda dapat menerapkan batas terpisah untuk setiap lisensi. Jika sebuah langkah membutuhkan dua batasan dan batas untuk salah satu sumber daya tercapai, tugas dalam langkah itu tidak akan diambil oleh pekerja tambahan sampai sumber daya tersedia.

Menghentikan dan menghapus batas

Saat Anda menghentikan atau menghapus asosiasi antara antrian dan batas, pekerjaan yang menggunakan batas menghentikan penjadwalan tugas dari langkah-langkah yang memerlukan batas ini dan memblokir pembuatan sesi baru untuk satu langkah.

Tugas yang berada dalam status READY tetap siap, dan tugas secara otomatis dilanjutkan dengan asosiasi antara antrian dan batas menjadi aktif kembali. Anda tidak perlu meminta pekerjaan apa pun.

Ketika Anda menghentikan atau menghapus asosiasi antara antrian dan batas, Anda memiliki dua pilihan tentang cara menghentikan menjalankan tugas:

- Hentikan dan batalkan tugas — Pekerja dengan sesi yang memperoleh batas membatalkan semua tugas.
- Berhenti dan selesaikan tugas yang sedang berjalan — Pekerja dengan sesi yang memperoleh batas menyelesaikan tugas mereka.

Ketika Anda menghapus batas menggunakan konsol, pekerja pertama-tama berhenti menjalankan tugas segera atau akhirnya ketika mereka selesai. Ketika asosiasi dihapus, hal berikut terjadi:

- Langkah-langkah yang membutuhkan batas ditandai tidak kompatibel.
- Seluruh pekerjaan yang berisi langkah-langkah tersebut dibatalkan, termasuk langkah-langkah yang tidak memerlukan batas.
- Pekerjaan ditandai tidak kompatibel.

Jika antrian yang terkait dengan batas memiliki armada terkait dengan kemampuan armada yang sesuai dengan jumlah persyaratan nama batas, armada tersebut akan terus memproses pekerjaan dengan batas yang ditentukan.

Buat batas

Anda membuat batas menggunakan konsol Deadline Cloud atau CreateLimit operasi di Deadline Cloud API. Batas didefinisikan untuk pertanian, tetapi terkait dengan antrian. Setelah Anda membuat batas, Anda dapat mengaitkannya dengan satu atau lebih antrian.

Untuk membuat batas

1. Dari dasbor Deadline Cloud console (Deadline Cloud console), pilih farm yang ingin Anda buat antrean.
2. Pilih peternakan untuk menambahkan batas, pilih tab Batas, lalu pilih Buat batas.
3. Berikan detail untuk batasnya. Nama persyaratan Jumlah adalah nama yang digunakan dalam template pekerjaan untuk mengidentifikasi batas. Itu harus dimulai dengan awalan **`amount.`** diikuti dengan nama jumlah. Nama persyaratan jumlah harus unik dalam antrian yang terkait dengan batas.
4. Jika Anda memilih Tetapkan jumlah maksimum, itu adalah jumlah total sumber daya yang diizinkan oleh batas ini. Jika Anda memilih Tidak ada jumlah maksimum, penggunaan sumber daya tidak terbatas. Bahkan ketika penggunaan sumber daya tidak terbatas, CloudWatch metrik `CurrentCount` Amazon dipancarkan sehingga Anda dapat melacak penggunaan. Untuk informasi selengkapnya, lihat CloudWatchmetrik di Panduan Pengembang Cloud Deadline.
5. Jika Anda sudah tahu antrian yang harus menggunakan batas, Anda dapat memilihnya sekarang. Anda tidak perlu mengaitkan antrian untuk membuat batas.
6. Pilih Buat batas.

Kaitkan batas dan antrian

Setelah Anda membuat batas, Anda dapat mengaitkan satu atau lebih antrian dengan batas. Hanya antrian yang terkait dengan batas yang menggunakan nilai yang ditentukan dalam batas.

Anda membuat asosiasi dengan antrian menggunakan konsol Deadline Cloud atau CreateQueueLimitAssociation operasi di Deadline Cloud API.

Untuk mengaitkan antrian dengan batas

1. Dari dasbor Deadline Cloud console (Deadline Cloud console), pilih farm tempat Anda ingin mengaitkan batas dengan antrian.
2. Pilih tab Limits, pilih limit untuk mengaitkan antrian dengan, lalu pilih Edit limit.

3. Di bagian antrian Associate, pilih antrian yang akan dikaitkan dengan batas.
4. Pilih Simpan perubahan.

Kirim pekerjaan yang membutuhkan batasan

Anda menerapkan batas dengan menentukannya sebagai persyaratan tuan rumah untuk pekerjaan atau langkah pekerjaan. Jika Anda tidak menentukan batas dalam satu langkah dan langkah itu menggunakan sumber daya terkait, penggunaan langkah tidak dihitung terhadap batas saat pekerjaan dijadwalkan..

Beberapa submitter Deadline Cloud memungkinkan Anda untuk menetapkan persyaratan host. Anda dapat menentukan nama persyaratan jumlah batas di pengirim untuk menerapkan batas.

Jika pengirim Anda tidak mendukung penambahan persyaratan host, Anda juga dapat menerapkan batasan dengan mengedit templat pekerjaan untuk pekerjaan itu.

Untuk menerapkan batas pada langkah pekerjaan dalam bundel pekerjaan

1. Buka template pekerjaan untuk pekerjaan menggunakan editor teks. Template pekerjaan terletak di direktori bundel pekerjaan untuk pekerjaan itu. Untuk informasi selengkapnya, lihat Paket Job di Panduan Pengembang Cloud Deadline.
2. Temukan definisi langkah untuk langkah yang menerapkan batas.
3. Tambahkan yang berikut ini ke definisi langkah. Ganti *amount.name* dengan nama persyaratan jumlah batas Anda. Untuk penggunaan umum, Anda harus menetapkan `min` nilai ke 1.

   YAML

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

   JSON

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

```
        }
    }
}
```

Anda dapat menambahkan beberapa batasan ke langkah pekerjaan sebagai berikut. Ganti *amount.name_1* dan *amount.name_2* dengan nama persyaratan jumlah batas Anda.

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. Simpan perubahan pada template pekerjaan.

Cara mengirimkan pekerjaan ke Deadline Cloud

Ada banyak cara berbeda untuk mengirimkan pekerjaan ke AWS Deadline Cloud. Bagian ini menjelaskan beberapa cara Anda dapat mengirimkan pekerjaan menggunakan alat yang disediakan oleh Deadline Cloud atau dengan membuat alat kustom Anda sendiri untuk beban kerja Anda.

- Dari terminal — untuk saat Anda pertama kali mengembangkan paket pekerjaan, atau saat pengguna mengirimkan pekerjaan merasa nyaman menggunakan baris perintah
- Dari skrip - untuk menyesuaikan dan mengotomatiskan beban kerja
- Dari aplikasi — untuk saat pekerjaan pengguna berada dalam aplikasi, atau ketika konteks aplikasi penting.

Contoh berikut menggunakan pustaka `deadline` Python dan alat baris `deadline` perintah. Keduanya tersedia dari PyPidan di-host GitHub.

Topik

- Kirim pekerjaan ke Deadline Cloud dari terminal
- Kirim pekerjaan ke Deadline Cloud menggunakan skrip
- Kirim pekerjaan dalam aplikasi

Kirim pekerjaan ke Deadline Cloud dari terminal

Dengan hanya menggunakan bundel pekerjaan dan Deadline Cloud CLI, Anda atau pengguna Anda yang lebih teknis dapat dengan cepat mengulangi penulisan bundel pekerjaan untuk menguji pengiriman pekerjaan. Gunakan perintah berikut untuk mengirimkan bundel pekerjaan:

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

Jika Anda mengirimkan bundel pekerjaan dengan parameter yang tidak memiliki default dalam bundel, Anda dapat menentukannya dengan opsi/. `-p --parameter`

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

Untuk daftar lengkap opsi yang tersedia, jalankan perintah bantuan:

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

Kirim pekerjaan ke Deadline Cloud menggunakan GUI

Deadline Cloud CLI juga dilengkapi dengan antarmuka pengguna grafis yang memungkinkan pengguna untuk melihat parameter yang harus mereka berikan sebelum mengirimkan pekerjaan. Jika

pengguna Anda memilih untuk tidak berinteraksi dengan baris perintah, Anda dapat menulis pintasan desktop yang membuka dialog untuk mengirimkan bundel pekerjaan tertentu:

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

Gunakan `--browse` opsi ini sehingga pengguna dapat memilih bundel pekerjaan:

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

Untuk daftar lengkap opsi yang tersedia, jalankan perintah bantuan:

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

Kirim pekerjaan ke Deadline Cloud menggunakan skrip

Untuk mengotomatiskan pengiriman pekerjaan ke Deadline Cloud, Anda dapat membuat skrip menggunakan alat seperti bash, Powershell, dan file batch.

Anda dapat menambahkan fungsionalitas seperti mengisi parameter pekerjaan dari variabel lingkungan atau aplikasi lain. Anda juga dapat mengirimkan beberapa pekerjaan berturut-turut, atau membuat skrip pembuatan bundel pekerjaan untuk dikirimkan.

Kirim pekerjaan menggunakan Python

Deadline Cloud juga memiliki pustaka Python open-source untuk berinteraksi dengan layanan. Kode sumber tersedia di GitHub.

Pustaka tersedia di pypi melalui pip ()`pip install deadline`. Ini adalah perpustakaan yang sama yang digunakan oleh alat Deadline Cloud CLI:

```
from deadline.client import api

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

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

Untuk membuat dialog seperti `deadline bundle gui-submit` perintah, Anda dapat menggunakan `show_job_bundle_submitter` fungsi dari `deadline.client.ui.job_bundle_submitter`.

Contoh berikut memulai aplikasi Qt dan menunjukkan pengirim bundel pekerjaan:

```
# 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)
```

Untuk membuat dialog Anda sendiri, Anda dapat menggunakan `SubmitJobToDeadlineDialog` kelas di `deadline.client.ui.dialogs.submit_job_to_deadline_dialog`. Anda dapat meneruskan nilai, menyematkan tab spesifik pekerjaan Anda sendiri, dan menentukan bagaimana bundel pekerjaan dibuat (atau diteruskan).

Kirim pekerjaan dalam aplikasi

Untuk memudahkan pengguna mengirimkan pekerjaan, Anda dapat menggunakan runtime scripting atau sistem plugin yang disediakan oleh aplikasi. Pengguna memiliki antarmuka yang akrab dan Anda dapat membuat alat canggih yang membantu pengguna saat mengirimkan beban kerja.

Sematkan bundel pekerjaan dalam aplikasi

Contoh ini menunjukkan pengiriman bundel pekerjaan yang Anda sediakan dalam aplikasi.

Untuk memberi pengguna akses ke bundel pekerjaan ini, buat skrip yang disematkan dalam item menu yang meluncurkan CLI Deadline Cloud.

Skrip berikut memungkinkan pengguna untuk memilih bundel pekerjaan:

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

Untuk menggunakan bundel pekerjaan tertentu dalam item menu sebagai gantinya, gunakan yang berikut ini:

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

Ini membuka dialog di mana pengguna dapat memodifikasi parameter pekerjaan, input, dan output, dan kemudian mengirimkan pekerjaan. Anda dapat memiliki item menu yang berbeda untuk bundel pekerjaan yang berbeda bagi pengguna untuk mengirimkan dalam aplikasi.

Jika job yang Anda kirimkan dengan bundel pekerjaan berisi parameter dan referensi aset yang serupa di seluruh kiriman, Anda dapat mengisi nilai default di bundel pekerjaan yang mendasarinya.

Dapatkan informasi dari aplikasi

Untuk menarik informasi dari aplikasi sehingga pengguna tidak perlu menambahkannya secara manual ke kiriman, Anda dapat mengintegrasikan Deadline Cloud dengan aplikasi sehingga pengguna Anda dapat mengirimkan pekerjaan menggunakan antarmuka yang sudah dikenal tanpa perlu keluar dari aplikasi atau menggunakan alat baris perintah.

Jika aplikasi Anda memiliki runtime scripting yang mendukung Python dan pyside/pyqt, Anda dapat menggunakan komponen GUI dari pustaka klien Deadline Cloud untuk membuat UI. Sebagai contoh, lihat Deadline Cloud untuk integrasi Maya pada GitHub.

Pustaka klien Deadline Cloud menyediakan operasi yang melakukan hal berikut untuk membantu Anda memberikan pengalaman pengguna terintegrasi yang kuat:

- Tarik parameter lingkungan antrian, parameter pekerjaan, dan referensi aset membentuk variabel lingkungan dan dengan memanggil SDK aplikasi.
- Atur parameter dalam bundel pekerjaan. Untuk menghindari memodifikasi bundel asli, Anda harus membuat salinan bundel dan mengirimkan salinannya.

Jika Anda menggunakan `deadline bundle gui-submit` perintah untuk mengirimkan bundel pekerjaan, Anda harus secara terprogram `asset_references.yaml` file `parameter_values.yaml` dan untuk meneruskan informasi dari aplikasi. Untuk informasi selengkapnya tentang file-file ini, lihatTemplat Open Job Description (OpenJD) untuk Deadline Cloud.

Jika Anda memerlukan kontrol yang lebih kompleks daripada yang ditawarkan oleh OpenJD, perlu mengabstraksi pekerjaan dari pengguna, atau ingin membuat integrasi cocok dengan gaya visual aplikasi, Anda dapat menulis dialog Anda sendiri yang memanggil pustaka klien Deadline Cloud untuk mengirimkan pekerjaan.

Jadwalkan pekerjaan di Deadline Cloud

Setelah Anda membuat pekerjaan, AWS Deadline Cloud menjadwalkannya untuk diproses pada satu atau beberapa armada yang terkait dengan antrian. Armada yang memproses tugas tertentu dipilih berdasarkan konfigurasi penjadwalan, kemampuan yang dikonfigurasi untuk armada, dan persyaratan tuan rumah dari langkah tertentu.

Bagian berikut memberikan rincian proses penjadwalan pekerjaan.

Konfigurasi penjadwalan

Anda dapat mengonfigurasi cara Deadline Cloud menjadwalkan pekerjaan dalam antrian dengan menyetel konfigurasi penjadwalan pada antrian. Konfigurasi penjadwalan mengontrol bagaimana pekerja didistribusikan di seluruh pekerjaan.

Anda dapat mengatur konfigurasi penjadwalan menggunakan konsol Deadline Cloud atau dengan memanggil atau. CreateQueueUpdateQueue APIs

Ada tiga konfigurasi penjadwalan yang tersedia:

- Prioritas, first-in-first-out (`priorityFifo`) - Menjadwalkan prioritas tertinggi, pekerjaan yang diajukan paling awal terlebih dahulu (default).
- Prioritas, seimbang (`priorityBalanced`) - Mendistribusikan pekerja secara merata di seluruh pekerjaan dengan prioritas tertinggi.
- Weighted, balanced (`weightedBalanced`) — Menggunakan formula tertimbang untuk menentukan bagaimana pekerja didistribusikan di seluruh pekerjaan.

Dalam semua konfigurasi penjadwalan, tugas yang sedang berlangsung berjalan hingga selesai sebelum keputusan penjadwalan baru dibuat. Jika Anda mengubah konfigurasi penjadwalan saat tugas sedang berjalan, perubahan hanya berlaku ketika pekerja ditugaskan berikutnya. Menjalankan tugas tidak terganggu atau dipindahkan.

Prioritas, first-in-first-out

Priority, first-in-first-out (`priorityFifo`) adalah konfigurasi penjadwalan default untuk antrian baru. Deadline Cloud menugaskan pekerja ke pekerjaan dengan prioritas tertinggi terlebih dahulu. Ketika beberapa pekerjaan memiliki prioritas yang sama, pekerjaan tertua (paling awal dikirimkan) menerima semua pekerja yang tersedia terlebih dahulu.

Gunakan prioritas FIFO ketika Anda ingin pemesanan pekerjaan yang ketat. Konfigurasi ini sesuai ketika pekerjaan harus diselesaikan satu per satu dalam urutan yang diajukan, seperti tahapan pipa berurutan atau pemrosesan batch di mana setiap pekerjaan harus diselesaikan sebelum pekerjaan berikutnya dimulai.

Konfigurasi ini tidak memiliki parameter tambahan.

Prioritas, seimbang

Priority, balanced (`priorityBalanced`) mendistribusikan pekerja secara merata di semua pekerjaan pada tingkat prioritas tertinggi. Ketika hanya satu pekerjaan yang ada pada prioritas tertinggi, Deadline Cloud menugaskan semua pekerja untuk pekerjaan itu. Ketika beberapa pekerjaan berbagi prioritas tertinggi, pekerja dibagi rata di antara mereka. Jika pekerja tidak dapat dibagi secara merata, pekerja tambahan didistribusikan di antara pekerjaan prioritas tertinggi.

Gunakan prioritas seimbang ketika beberapa artis atau pengguna mengirimkan pekerjaan pada prioritas yang sama dan setiap pengguna membutuhkan umpan balik segera. Konfigurasi ini memastikan bahwa tidak ada pekerjaan tunggal yang memonopoli semua pekerja yang tersedia, sehingga semua pengguna dialokasikan pekerja segera setelah pengiriman.

Jika pekerjaan memiliki lebih sedikit tugas yang tersisa daripada bagian pekerjanya, pekerja surplus didistribusikan kembali ke pekerjaan lain pada tingkat prioritas yang sama. Jika semua pekerjaan dengan prioritas tertinggi dialokasikan sepenuhnya, pekerja surplus mengalir ke pekerjaan di tingkat prioritas tertinggi berikutnya.

Konfigurasi ini memiliki parameter berikut:

`renderingTaskBuffer`

Mengontrol kelengketan pekerja. Seorang pekerja beralih dari pekerjaannya saat ini ke pekerjaan lain dengan prioritas yang sama hanya jika perbedaan dalam tugas rendering melebihi `renderingTaskBuffer` nilainya. Nilai yang lebih tinggi membuat pekerja pada pekerjaan mereka saat ini lebih lama, mengurangi peralihan konteks. Nilai default-nya adalah `1`.

Berbobot, seimbang

Weighted, balanced (`weightedBalanced`) menggunakan rumus untuk menghitung berat untuk setiap pekerjaan. Deadline Cloud menugaskan pekerja ke pekerjaan dengan bobot tertinggi terlebih dahulu. Jika beberapa pekerjaan memiliki bobot yang sama, pekerja didistribusikan di antara mereka.

Gunakan timbangan berbobot ketika Anda membutuhkan kontrol halus atas bagaimana pekerja didistribusikan di seluruh pekerjaan dengan berbagai prioritas, tingkat kesalahan, dan waktu pengiriman. Konfigurasi ini sesuai untuk lingkungan pertanian render kompleks di mana Anda ingin menyesuaikan keseimbangan antara prioritas pekerjaan, usia pekerjaan, penanganan kesalahan, dan kelengketan pekerja.

Bobot untuk setiap pekerjaan dihitung sebagai berikut:

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

`renderingTaskBuffer`Komponen diterapkan hanya jika pekerja saat ini sedang mengerjakan pekerjaan. Biasanya `renderingTaskWeight` diatur ke nilai negatif sehingga pekerjaan dengan pekerja yang ditugaskan menerima bobot yang lebih rendah, membawa pekerjaan lain ke depan antrian. `errorWeight`Itu juga biasanya negatif sehingga pekerjaan dengan kesalahan tidak diprioritaskan. Anda dapat menggunakan penggantian penjadwalan untuk pekerjaan prioritas minimum dan maksimum.

Konfigurasi ini memiliki parameter berikut:

`priorityWeight`

Bobot diterapkan pada prioritas pekerjaan. Nilai positif berarti pekerjaan dengan prioritas lebih tinggi dijadwalkan terlebih dahulu. Nilai default-nya adalah `100.0`. Rentang: `0` ke`10000`.

`errorWeight`

Bobot yang diterapkan pada hitungan kesalahan pekerjaan. Nilai negatif berarti pekerjaan tanpa kesalahan dijadwalkan terlebih dahulu. Nilai default-nya adalah `-10.0`. Rentang: `-10000` ke`10000`.

`submissionTimeWeight`

Bobot diterapkan pada waktu pengiriman pekerjaan (dalam detik). Nilai positif berarti pekerjaan yang diajukan sebelumnya dijadwalkan terlebih dahulu. Nilai default-nya adalah `3.0`. Rentang: `0` ke`10000`.

`renderingTaskWeight`

Bobot diterapkan pada jumlah tugas yang saat ini dirender untuk suatu pekerjaan. Nilai negatif berarti pekerjaan dengan lebih sedikit pekerja dijadwalkan berikutnya. Nilai default-nya adalah `-100.0`. Rentang: `-10000` ke`10000`.

`renderingTaskBuffer`

Jumlah tugas rendering sebelum bobot tugas rendering berlaku. Nilai positif membuat pekerja tetap pada pekerjaan mereka saat ini. Nilai default-nya adalah `1`. Rentang: `0` ke`1000`.

`maxPriorityOverride`

Tidak wajib. Ketika diatur k`alwaysScheduleFirst`, pekerjaan dengan prioritas maksimum (100) selalu dijadwalkan sebelum pekerjaan lain, terlepas dari rumus tertimbang. Ketika beberapa pekerjaan memiliki prioritas maksimum, ikatan diputus menggunakan rumus tertimbang standar. Ketika penggantian tidak ada, pekerjaan prioritas maksimum menggunakan formula tertimbang standar tanpa perlakuan khusus.

`minPriorityOverride`

Tidak wajib. Ketika diatur k`alwaysScheduleLast`, pekerjaan pada prioritas minimum (0) selalu dijadwalkan setelah pekerjaan lain, terlepas dari rumus tertimbang. Ketika beberapa pekerjaan memiliki prioritas minimum, ikatan diputus menggunakan rumus tertimbang standar. Ketika penggantian tidak ada, pekerjaan prioritas minimum menggunakan formula tertimbang standar tanpa perlakuan khusus.

Tentukan kompatibilitas armada

Setelah Anda membuat pekerjaan, Deadline Cloud memeriksa persyaratan host untuk setiap langkah dalam pekerjaan terhadap kemampuan armada yang terkait dengan antrian pekerjaan yang dikirimkan. Jika armada memenuhi persyaratan tuan rumah, pekerjaan itu dimasukkan ke `READY` negara bagian.

Jika ada langkah dalam pekerjaan yang memiliki persyaratan yang tidak dapat dipenuhi oleh armada yang terkait dengan antrian, status langkah diatur ke`NOT_COMPATIBLE`. Selain itu, sisa langkah dalam pekerjaan dibatalkan.

Kemampuan untuk armada ditetapkan pada tingkat armada. Bahkan jika seorang pekerja dalam armada memenuhi persyaratan pekerjaan, itu tidak akan diberikan tugas dari pekerjaan jika armadanya tidak memenuhi persyaratan pekerjaan.

Template pekerjaan berikut memiliki langkah yang menentukan persyaratan host untuk langkah tersebut:

```
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
```

Pekerjaan ini dapat dijadwalkan ke armada dengan kemampuan sebagai berikut:

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

```
}
```

Pekerjaan ini tidak dapat dijadwalkan ke armada dengan salah satu kemampuan berikut:

```
{
    "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.
```

Penskalaan armada

Ketika pekerjaan ditugaskan ke armada yang dikelola layanan yang kompatibel, armada diskalakan secara otomatis. Jumlah pekerja di armada berubah berdasarkan jumlah tugas yang tersedia untuk dijalankan armada.

Ketika pekerjaan ditugaskan ke armada yang dikelola pelanggan, pekerja mungkin sudah ada atau dapat dibuat menggunakan penskalaan otomatis berbasis peristiwa. Untuk informasi selengkapnya, lihat Menggunakan EventBridge untuk menangani peristiwa penskalaan otomatis di Panduan Pengguna Amazon EC2 Auto Scaling.

Sesi

Tugas dalam suatu pekerjaan dibagi menjadi satu atau lebih sesi. Pekerja menjalankan sesi untuk mengatur lingkungan, menjalankan tugas, dan kemudian meruntuhkan lingkungan. Setiap sesi terdiri dari satu atau lebih tindakan yang harus dilakukan seorang pekerja.

Saat pekerja menyelesaikan tindakan bagian, tindakan sesi tambahan dapat dikirim ke pekerja. Pekerja menggunakan kembali lingkungan yang ada dan lampiran pekerjaan dalam sesi untuk menyelesaikan tugas dengan lebih efisien.

Pada pekerja armada yang dikelola layanan, direktori sesi dihapus setelah sesi berakhir, tetapi direktori lain dipertahankan di antara sesi. Perilaku ini memungkinkan Anda menerapkan strategi caching untuk data yang dapat digunakan kembali di beberapa sesi. Untuk menyimpan data antar sesi, simpan di bawah direktori home pengguna yang menjalankan pekerjaan. Misalnya, paket conda di-cache di bawah direktori home pengguna pekerjaan `C:\Users\job-user\.conda-pkgs` di Windows pekerja dan `/home/job-user/.conda-pkgs` pekerja. Linux Data ini tetap tersedia sampai pekerja dimatikan.

Lampiran Job dibuat oleh pengirim yang Anda gunakan sebagai bagian dari paket pekerjaan Deadline Cloud CLI Anda. Anda juga dapat membuat lampiran pekerjaan menggunakan `--attachments` opsi untuk `create-job` AWS CLI perintah. Lingkungan didefinisikan di dua tempat: lingkungan antrian yang dilampirkan ke antrian tertentu, dan lingkungan pekerjaan dan langkah yang ditentukan dalam templat pekerjaan.

Ada empat jenis tindakan sesi:

- `syncInputJobAttachments`— Mengunduh lampiran pekerjaan input ke pekerja.
- `envEnter`— Melakukan `onEnter` tindakan untuk suatu lingkungan.
- `taskRun`— Melakukan `onRun` tindakan untuk suatu tugas.
- `envExit`— Melakukan `onExit` tindakan untuk suatu lingkungan.

Template pekerjaan berikut memiliki lingkungan langkah. Ini memiliki `onEnter` definisi untuk mengatur lingkungan langkah, `onRun` definisi yang mendefinisikan tugas yang akan dijalankan, dan `onExit` definisi untuk meruntuhkan lingkungan langkah. Sesi yang dibuat untuk pekerjaan ini akan mencakup `envEnter` tindakan, satu atau lebih `taskRun` tindakan, dan kemudian `envExit` tindakan.

```
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 }}
```

Tindakan sesi pipelining

Tindakan sesi pipelining memungkinkan penjadwal pra-menetapkan beberapa tindakan sesi ke pekerja. Pekerja kemudian dapat menjalankan tindakan ini secara berurutan, mengurangi atau menghilangkan waktu idle antar tugas.

Untuk membuat tugas awal, penjadwal membuat sesi dengan satu tugas, pekerja menyelesaikan tugas, dan kemudian penjadwal menganalisis durasi tugas untuk menentukan tugas masa depan.

Agar penjadwal efektif, ada aturan durasi tugas. Untuk tugas di bawah satu menit, penjadwal menggunakan pola pertumbuhan power-of-2. Misalnya, untuk tugas 1 detik, penjadwal memberikan 2 tugas baru, lalu 4, lalu 8. Untuk tugas lebih dari satu menit, penjadwal hanya menetapkan satu tugas baru dan pipelining tetap dinonaktifkan.

Untuk menghitung ukuran pipa, penjadwal melakukan hal berikut:

- Menggunakan durasi tugas rata-rata dari tugas yang diselesaikan
- Bertujuan untuk membuat pekerja sibuk selama satu menit
- Mempertimbangkan hanya tugas dalam sesi yang sama
- Tidak berbagi data durasi di seluruh pekerja

Dengan piplelining tindakan sesi, pekerja segera memulai tugas baru dan tidak ada waktu tunggu di antara permintaan penjadwal. Ini juga memberikan peningkatan efisiensi pekerja dan distribusi tugas yang lebih baik untuk proses yang berjalan lama.

Selain itu, jika ada pekerjaan prioritas baru yang lebih tinggi yang tersedia, pekerja akan menyelesaikan semua pekerjaan yang ditugaskan sebelumnya sebelum sesi saat ini berakhir dan sesi baru dari pekerjaan prioritas yang lebih tinggi ditugaskan.

Ketergantungan langkah

Deadline Cloud mendukung mendefinisikan dependensi antar langkah sehingga satu langkah menunggu hingga langkah lain selesai sebelum memulai. Anda dapat menentukan lebih dari satu ketergantungan untuk satu langkah. Langkah dengan ketergantungan tidak dijadwalkan sampai semua dependensinya selesai.

Jika template pekerjaan mendefinisikan ketergantungan melingkar, pekerjaan ditolak dan status pekerjaan disetel ke. `CREATE_FAILED`

Template pekerjaan berikut membuat pekerjaan dengan dua langkah. `StepB`tergantung pada`StepA`. `StepB`hanya berjalan setelah `StepA` selesai dengan sukses.

Setelah pekerjaan dibuat, `StepA` berada di `READY` negara bagian dan `StepB` berada di `PENDING` negara bagian. Setelah `StepA` selesai, `StepB` pindah ke `READY` negara bagian. Jika `StepA` gagal, atau `StepA` jika dibatalkan, `StepB` pindah ke `CANCELED` negara bagian.

Anda dapat mengatur ketergantungan pada beberapa langkah. Misalnya, jika `StepC` tergantung pada keduanya `StepA` dan`StepB`, `StepC` tidak akan dimulai sampai dua langkah lainnya selesai.

Dependensi langkah memiliki batasan berikut:

- Dependensi per langkah — Sebuah langkah dapat bergantung pada maksimum 128 langkah lainnya.
- Konsumen per langkah — Maksimal 32 langkah lain dapat bergantung pada satu langkah.

```
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!
```

Ubah pekerjaan di Deadline Cloud

Anda dapat menggunakan `update` perintah AWS Command Line Interface (AWS CLI) berikut untuk mengubah konfigurasi pekerjaan, atau untuk menetapkan status target pekerjaan, langkah, atau tugas:

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

Dalam contoh `update` perintah berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Example— Meminta pekerjaan

Semua tugas dalam pekerjaan beralih ke `READY` status, kecuali ada dependensi langkah. Langkah-langkah dengan dependensi beralih ke salah satu `READY` atau `PENDING` saat dipulihkan.

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

Example— Batalkan pekerjaan

Semua tugas dalam pekerjaan yang tidak memiliki status SUCCEEDED atau FAILED ditandaiCANCELED.

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

Example— Tandai pekerjaan gagal

Semua tugas dalam pekerjaan yang memiliki status SUCCEEDED dibiarkan tidak berubah. Semua tugas lainnya ditandaiFAILED.

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

Example— Tandai pekerjaan yang sukses

Semua tugas dalam pekerjaan pindah ke SUCCEEDED negara bagian.

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

Example— Menangguhkan pekerjaan

Tugas dalam pekerjaan diSUCCEEDED,CANCELED, atau FAILED negara bagian tidak berubah. Semua tugas lainnya ditandaiSUSPENDED.

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

Example— Mengubah prioritas pekerjaan

Memperbarui prioritas pekerjaan dalam antrian untuk mengubah urutan yang dijadwalkan. Pekerjaan prioritas yang lebih tinggi umumnya dijadwalkan terlebih dahulu.

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

Example— Ubah jumlah tugas gagal yang diizinkan

Memperbarui jumlah maksimum tugas yang gagal yang dapat dimiliki pekerjaan sebelum tugas yang tersisa dibatalkan.

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

Example— Ubah jumlah percobaan ulang tugas yang diizinkan

Memperbarui jumlah maksimum percobaan ulang untuk tugas sebelum tugas gagal. Tugas yang telah mencapai jumlah percobaan ulang maksimum tidak dapat diulang sampai nilai ini meningkat.

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

Example— Arsipkan pekerjaan

Memperbarui status siklus hidup pekerjaan ke. `ARCHIVED` Pekerjaan yang diarsipkan tidak dapat dijadwalkan atau diubah. Anda hanya dapat mengarsipkan pekerjaan yang ada di`FAILED`, `CANCELEDSUCCEEDED`, atau `SUSPENDED` negara bagian.

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

```
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example— Mengubah nama pekerjaan

Memperbarui nama tampilan pekerjaan. Panjang nama pekerjaan bisa mencapai 128 karakter.

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

Example— Mengubah deskripsi pekerjaan

Memperbarui deskripsi pekerjaan. Deskripsi dapat mencapai 2048 karakter. Untuk menghapus deskripsi yang ada, berikan string kosong.

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

Example— Meminta satu langkah

Semua tugas di langkah beralih ke `READY` status, kecuali ada dependensi langkah. Tugas dalam langkah-langkah dengan dependensi beralih ke salah satu `READY` atau`PENDING`, dan tugas dipulihkan.

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

Example— Batalkan langkah

Semua tugas dalam langkah yang tidak memiliki status `SUCCEEDED` atau `FAILED` ditandai`CANCELED`.

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

Example— Tandai langkah gagal

Semua tugas dalam langkah yang memiliki status SUCCEEDED dibiarkan tidak berubah. Semua tugas lainnya ditandaiFAILED.

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

Example— Tandai langkah sukses

Semua tugas dalam langkah ditandaiSUCCEEDED.

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

Example— Tangguhkan satu langkah

Tugas dalam langkah diSUCCEEDED,CANCELED, atau FAILED status tidak berubah. Semua tugas lainnya ditandaiSUSPENDED.

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

Example— Mengubah status tugas

Saat Anda menggunakan perintah `update-task` Deadline Cloud CLI, tugas beralih ke status yang ditentukan.

```
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
```

Membuat dan menggunakan armada yang dikelola pelanggan Deadline Cloud

Saat membuat armada yang dikelola pelanggan (CMF), Anda memiliki kendali penuh atas pipeline pemrosesan Anda. Anda menentukan lingkungan jaringan dan perangkat lunak untuk setiap pekerja. Deadline Cloud bertindak sebagai repositori dan penjadwal untuk pekerjaan Anda.

Pekerja dapat berupa instans Amazon Elastic Compute Cloud (Amazon EC2), pekerja di fasilitas lokasi bersama, atau pekerja lokal. Setiap pekerja harus menjalankan agen pekerja Deadline Cloud. Semua pekerja harus memiliki akses ke titik akhir layanan Deadline Cloud.

Topik berikut menunjukkan cara membuat CMF dasar menggunakan instans Amazon EC2.

Topik

Buat armada yang dikelola pelanggan

Untuk membuat armada yang dikelola pelanggan (CMF), selesaikan langkah-langkah berikut.

Deadline Cloud console

Untuk menggunakan konsol Deadline Cloud untuk membuat armada yang dikelola pelanggan

1. Buka konsol Deadline Cloud.
2. Pilih Peternakan. Daftar pajangan pertanian yang tersedia.
3. Pilih nama Peternakan tempat Anda ingin bekerja.

4. Pilih tab Armada, lalu pilih Buat armada.
5. Masukkan Nama untuk armada Anda.
6. (Opsional) Masukkan Deskripsi untuk armada Anda.
7. Pilih Pelanggan yang dikelola untuk jenis Armada.
8. Pilih akses layanan armada Anda.
    a. Sebaiknya gunakan opsi Buat dan gunakan peran layanan baru untuk setiap armada untuk kontrol izin yang lebih terperinci. Opsi ini dipilih secara default.
    b. Anda juga dapat menggunakan peran layanan yang ada dengan memilih Pilih peran layanan.
9. Tinjau pilihan Anda, lalu pilih Berikutnya.
10. Pilih sistem operasi untuk armada Anda. Semua pekerja armada harus memiliki sistem operasi yang sama.
11. Pilih arsitektur CPU host.
12. Pilih kemampuan perangkat keras vCPU dan memori minimum dan maksimum untuk memenuhi tuntutan beban kerja armada Anda.
13. Pilih jenis Auto Scaling. Untuk informasi selengkapnya, lihat Menggunakan EventBridge untuk menangani peristiwa Auto Scaling.
    - Tanpa penskalaan: Anda membuat armada lokal dan ingin memilih keluar dari Deadline Cloud Auto Scaling.
    - Rekomendasi penskalaan: Anda membuat armada Amazon Elastic Compute Cloud (Amazon EC2).
14. (Opsional) Pilih panah untuk memperluas bagian Tambahkan kemampuan.
15. (Opsional) Pilih kotak centang untuk Tambahkan kemampuan GPU - Opsional, lalu masukkan minimum dan maksimum GPUs dan memori.
16. Tinjau pilihan Anda, lalu pilih Berikutnya.
17. (Opsional) Tentukan kemampuan pekerja khusus, lalu pilih Berikutnya.
18. Menggunakan dropdown, pilih satu atau lebih antrian untuk dikaitkan dengan armada.

> **Note**
>
> Kami merekomendasikan untuk mengaitkan armada hanya dengan antrian yang semuanya berada dalam batas kepercayaan yang sama. Rekomendasi ini

memastikan batas keamanan yang kuat antara menjalankan pekerjaan pada pekerja yang sama.

19. Tinjau asosiasi antrian, lalu pilih Berikutnya.
20. (Opsional) Untuk lingkungan antrian Conda Default, kami akan membuat lingkungan untuk antrian Anda yang akan menginstal paket conda yang diminta oleh pekerjaan.

> **Note**
>
> Lingkungan antrian conda digunakan untuk menginstal paket conda yang diminta oleh pekerjaan. Biasanya, Anda harus menghapus centang pada lingkungan antrian conda pada antrian yang terkait dengan CMFs karena tidak CMFs akan memiliki perintah conda yang diperlukan diinstal secara default.

21. (Opsional) Tambahkan tag ke CMF Anda. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.
22. Tinjau konfigurasi armada Anda dan buat perubahan apa pun, lalu pilih Buat armada.
23. Pilih tab Armada, lalu catat ID Armada.

AWS CLI

Untuk menggunakan AWS CLI untuk membuat armada yang dikelola pelanggan

1. Buka terminal.
2. Buat `fleet-trust-policy.json` di editor baru.

    a. Tambahkan kebijakan IAM berikut, ganti *ITALICIZED* teks dengan ID AWS akun dan ID pertanian 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. Simpan perubahan Anda.

3. Buat`fleet-policy.json`.

a. Tambahkan kebijakan IAM berikut.

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. Simpan perubahan Anda.

4. Tambahkan peran IAM untuk digunakan pekerja di armada Anda.

```
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. Buat`create-fleet-request.json`.

a. Tambahkan kebijakan IAM berikut, ganti teks ITALICIZED dengan nilai CMF Anda.

> **Note**
>
> Anda dapat menemukan *`ROLE_ARN`* di`create-cmf-fleet.json`.

Untuk itu*OS_FAMILY*, Anda harus memilih salah satu`linux`, `macos` atau`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. Simpan perubahan Anda.

6. Buat armada Anda.

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

Pengaturan dan konfigurasi host pekerja

Host pekerja mengacu pada mesin host yang menjalankan pekerja Deadline Cloud. Bagian ini menjelaskan cara mengatur host pekerja dan mengonfigurasinya untuk kebutuhan spesifik Anda.

Setiap host pekerja menjalankan program yang disebut agen pekerja. Agen pekerja bertanggung jawab untuk:

- Mengelola siklus hidup pekerja.
- Sinkronisasi pekerjaan yang ditugaskan, kemajuan dan hasilnya.
- Memantau pekerjaan yang sedang berjalan.
- Meneruskan log ke tujuan yang dikonfigurasi.

Kami menyarankan Anda menggunakan agen pekerja Deadline Cloud yang disediakan. Agen pekerja adalah open source dan kami mendorong permintaan fitur, tetapi Anda juga dapat mengembangkan dan menyesuaikan agar sesuai dengan kebutuhan Anda.

Untuk menyelesaikan tugas di bagian berikut, Anda memerlukan yang berikut:

Linux

- LinuxInstans Amazon Elastic Compute Cloud (Amazon EC2) berbasis Amazon. Kami merekomendasikan Amazon Linux 2023.
- `sudo`hak istimewa
- Python 3.9 atau lebih tinggi

Windows

- WindowsInstans Amazon Elastic Compute Cloud (Amazon EC2) berbasis Amazon. Kami merekomendasikanWindows Server 2022.
- Akses administrator ke host pekerja
- Python 3.9 atau lebih tinggi diinstal untuk semua pengguna

Membuat dan mengkonfigurasi lingkungan virtual Python

Anda dapat membuat lingkungan virtual Python Linux jika Anda telah menginstal Python 3.9 atau lebih besar dan menempatkannya di lingkungan Anda. `PATH`

> **Note**
>
> PadaWindows, file agen harus diinstal ke direktori paket situs global Python. Lingkungan virtual Python saat ini tidak didukung.

Untuk membuat dan mengaktifkan lingkungan virtual Python

1. Buka terminal sebagai `root` pengguna (atau gunakan`sudo`/`su`).
2. Buat dan aktifkan lingkungan virtual Python.

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

Instal Deadline Agen pekerja Cloud

Setelah menyiapkan Python dan membuat lingkungan virtualLinux, instal paket Python agen pekerja Deadline Cloud.

Untuk menginstal paket Python agen pekerja

Linux

1. Buka terminal sebagai `root` pengguna (atau gunakan`sudo`/`su`).
2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

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

Windows

1. Buka prompt perintah administrator atau PowerShell terminal.
2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

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

Ketika host Windows pekerja Anda memerlukan nama jalur panjang (lebih dari 250 karakter), Anda harus mengaktifkan nama jalur panjang sebagai berikut:

Untuk mengaktifkan jalur panjang bagi host Windows pekerja

1. Pastikan bahwa kunci registri jalur panjang diaktifkan. Untuk informasi selengkapnya, lihat Setelan registri untuk mengaktifkan jalur log di situs web Microsoft.
2. Instal Windows SDK untuk Aplikasi Desktop C++ x86. Untuk informasi selengkapnya, lihat WindowsSDK di Pusat Windows Pengembang.
3. Buka lokasi instalasi Python di lingkungan Anda tempat agen pekerja diinstal. Nilai default-nya `C:\Program Files\Python311`. Ada file yang dapat dieksekusi bernama. `pythonservice.exe`
4. Buat file baru yang disebut `pythonservice.exe.manifest` di lokasi yang sama. Tambahkan yang berikut ini:

```
<?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. Buka prompt perintah dan jalankan perintah berikut di lokasi file manifes yang Anda buat:

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

Anda akan melihat output yang serupa dengan yang berikut:

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

Pekerja sekarang dapat mengakses jalur panjang. Untuk membersihkan, hapus `pythonservice.exe.manifest` file dan hapus instalan SDK.

Konfigurasikan agen pekerja Cloud Deadline

Anda dapat mengonfigurasi pengaturan agen pekerja Deadline Cloud dengan tiga cara. Kami menyarankan Anda menggunakan pengaturan sistem operasi dengan menjalankan `install-deadline-worker` alat.

Agen pekerja tidak mendukung berjalan sebagai pengguna domain di Windows. Untuk menjalankan pekerjaan sebagai pengguna domain, Anda dapat menentukan akun pengguna domain saat mengonfigurasi pengguna antrian untuk menjalankan pekerjaan. Untuk informasi selengkapnya, lihat langkah 7 dalam antrian Deadline Cloud di Panduan Pengguna Cloud AWS Deadline.

Argumen baris perintah - Anda dapat menentukan argumen saat menjalankan agen pekerja Deadline Cloud dari baris perintah. Beberapa pengaturan konfigurasi tidak tersedia melalui argumen baris perintah. Untuk melihat semua argumen baris perintah yang tersedia, masukkan`deadline-worker-agent --help`.

Variabel lingkungan — Anda dapat mengonfigurasi agen pekerja Deadline Cloud dengan menyetel variabel lingkungan yang dimulai dengan`DEADLINE_WORKER_`. Misalnya, untuk melihat semua argumen baris perintah yang tersedia, Anda dapat menggunakan `export DEADLINE_WORKER_VERBOSE=true` untuk mengatur output agen pekerja ke verbose. Untuk contoh dan informasi lebih lanjut, lihat `/etc/amazon/deadline/worker.toml.example` di Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` diWindows.

File konfigurasi — Ketika Anda menginstal agen pekerja, itu membuat file konfigurasi yang terletak di `/etc/amazon/deadline/worker.toml` on Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml` onWindows. Agen pekerja memuat file konfigurasi ini saat dimulai. Anda dapat menggunakan contoh file konfigurasi (`/etc/amazon/deadline/worker.toml.example`aktif Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` aktifWindows) untuk menyesuaikan file konfigurasi agen pekerja default untuk kebutuhan spesifik Anda.

Terakhir, kami sarankan Anda mengaktifkan auto shutdown untuk agen pekerja setelah perangkat lunak Anda digunakan dan berfungsi seperti yang diharapkan. Hal ini memungkinkan armada pekerja untuk meningkatkan skala ketika diperlukan dan untuk menutup ketika pekerjaan selesai. Penskalaan otomatis membantu memastikan Anda hanya menggunakan sumber daya yang dibutuhkan. Untuk

mengaktifkan instance yang dimulai oleh grup penskalaan otomatis untuk dimatikan, Anda harus menambahkan `shutdown_on_stop=true` ke file `worker.toml` konfigurasi.

Untuk mengaktifkan auto shutdown

Sebagai **`root`** pengguna:

- Instal agen pekerja dengan parameter**`--allow-shutdown`**.

  Linux

  Masukkan:

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

  Windows

  Masukkan:

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

Buat pengguna dan grup pekerjaan

Bagian ini menjelaskan hubungan pengguna dan grup yang diperlukan antara pengguna agen dan yang `jobRunAsUser` ditentukan pada antrian Anda.

Agen pekerja Deadline Cloud harus dijalankan sebagai pengguna khusus agen khusus di host. Anda harus mengonfigurasi `jobRunAsUser` properti antrian Deadline Cloud sehingga pekerja akan menjalankan pekerjaan antrian sebagai pengguna dan grup sistem operasi tertentu. Konfigurasi ini berarti Anda dapat mengontrol izin sistem file bersama yang dimiliki pekerjaan Anda. Ini juga menyediakan sebagai batas keamanan penting antara pekerjaan Anda dan pengguna agen pekerja.

Linuxpengguna pekerjaan dan grup

Untuk mengatur pengguna agen pekerja lokal dan`jobRunAsUser`, pastikan Anda memenuhi persyaratan berikut. Jika Anda menggunakan Linux Pluggable Authentication Module (PAM) seperti Active Directory atau LDAP, prosedur Anda mungkin berbeda.

Pengguna agen pekerja dan `jobRunAsUser` grup bersama disetel saat Anda menginstal agen pekerja. Defaultnya adalah `deadline-worker-agent` dan`deadline-job-users`, tetapi Anda dapat mengubahnya saat menginstal agen pekerja.

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

Perintah harus dijalankan sebagai pengguna root.

- Masing-masing `jobRunAsUser` harus memiliki kelompok utama yang cocok. Membuat pengguna dengan `adduser` perintah biasanya membuat grup utama yang cocok.

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

- Kelompok utama `jobRunAsUser` adalah kelompok sekunder untuk pengguna agen pekerja. Grup bersama memungkinkan agen pekerja untuk membuat file tersedia untuk pekerjaan saat sedang berjalan.

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

- `jobRunAsUser`Harus menjadi anggota kelompok kerja bersama.

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

- Tidak `jobRunAsUser` boleh termasuk dalam kelompok utama pengguna agen pekerja. File sensitif yang ditulis oleh agen pekerja dimiliki oleh kelompok utama agen. Jika a `jobRunAsUser` adalah bagian dari grup ini, file agen pekerja dapat diakses oleh pekerjaan yang berjalan pada pekerja.
- Default Wilayah AWS harus sesuai dengan Wilayah peternakan tempat pekerja tersebut berada. Ini harus diterapkan ke semua `jobRunAsUser` akun pada pekerja.

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

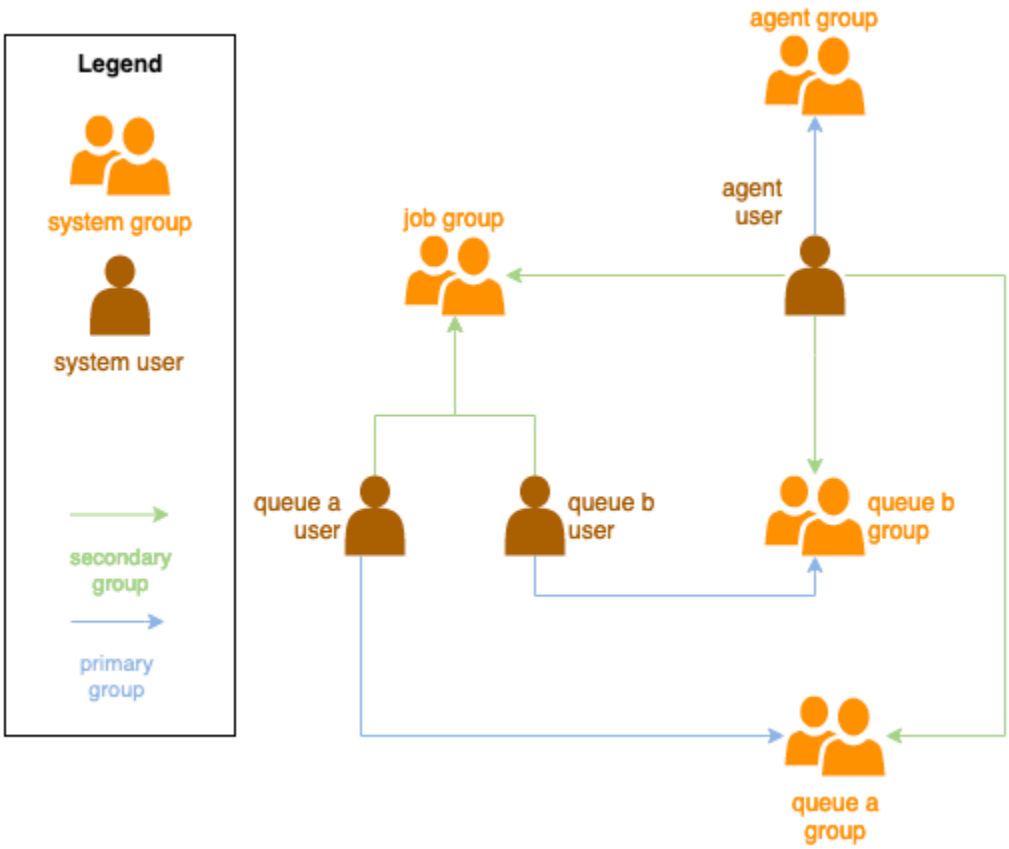
- Pengguna agen pekerja harus dapat menjalankan `sudo` perintah sebagai`jobRunAsUser`. Jalankan perintah berikut untuk membuka editor untuk membuat aturan sudoers baru:

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

Tambahkan yang berikut ini ke file:

```
# Allows the Deadline Cloud worker agent OS user to run commands
 # as the queue OS user without requiring a password.
deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
```

Diagram berikut menggambarkan hubungan antara pengguna agen dan `jobRunAsUser` pengguna dan grup untuk antrian yang terkait dengan armada.



Windowspengguna

Untuk menggunakan Windows pengguna sebagai`jobRunAsUser`, itu harus memenuhi persyaratan berikut:

- Semua `jobRunAsUser` pengguna antrian harus ada.
- Kata sandi mereka harus sesuai dengan nilai rahasia yang ditentukan dalam `JobRunAsUser` bidang antrian mereka. Untuk petunjuknya, lihat langkah 7 dalam antrian Deadline Cloud di Panduan Pengguna Cloud AWS Deadline.
- Agen-pengguna harus dapat masuk sebagai pengguna tersebut.

Mengamankan host pekerja Anda

Saat menyiapkan host pekerja Anda, ikuti praktik terbaik keamanan untuk melindungi informasi sensitif dan mempertahankan kontrol akses yang tepat.

Mengkonfigurasi izin folder log

Agen pekerja menulis file log yang mungkin berisi informasi sensitif dari skrip konfigurasi host dan eksekusi pekerjaan. `install-deadline-worker`Perintah membuat direktori log dengan izin aman. Jika Anda perlu membuat direktori secara manual sebelum instalasi, gunakan prosedur berikut untuk mencocokkan izin yang digunakan oleh armada yang dikelola layanan:

Linux

Untuk mengkonfigurasi izin direktori log pada Linux

1. Buat direktori log:

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

2. Setel pemilik dan grup ke pengguna agen pekerja:

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

3. Setel izin ke 750:

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

Izin ini memastikan bahwa hanya pengguna dan grup agen pekerja yang dapat mengakses file log, mencegah pengguna pekerjaan dan pengguna tidak sah lainnya membaca informasi yang berpotensi sensitif.

Windows

Untuk mengkonfigurasi izin direktori log pada Windows

1. Buka PowerShell terminal administrator.
2. Buat direktori log:

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

3. Konfigurasi terbatas ACLs untuk mengizinkan hanya pengguna agen pekerja dan Administrator:

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

Perintah ini membatasi akses ke direktori log hanya untuk pengguna agen pekerja dan grup Administrator, mencegah pengguna pekerjaan dan pengguna tidak sah lainnya membaca informasi yang berpotensi sensitif.

Mengelola akses ke rahasia pengguna Windows pekerjaan

Saat Anda mengonfigurasi antrian dengan a Windows`jobRunAsUser`, Anda harus menentukan AWS rahasia Secrets Manager. Nilai rahasia ini diharapkan menjadi objek yang dikodekan JSON dari bentuk:

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

Agar Pekerja menjalankan pekerjaan sesuai antrian yang dikonfigurasi`jobRunAsUser`, peran IAM armada harus memiliki izin untuk mendapatkan nilai rahasia. Jika rahasia dienkripsi menggunakan kunci KMS yang dikelola pelanggan, maka peran IAM armada juga harus memiliki izin untuk mendekripsi menggunakan kunci KMS.

Sangat disarankan untuk mengikuti prinsip hak istimewa paling sedikit untuk rahasia ini. Ini berarti bahwa akses untuk mengambil nilai rahasia dari antrian `jobRunAsUser` → `windows` → `passwordArn` harus:

- diberikan untuk peran armada ketika asosiasi antrian-armada dibuat antara armada dan antrian
- dicabut dari peran armada ketika asosiasi antrian-armada dihapus antara armada dan antrian

Selanjutnya, AWS rahasia Secrets Manager yang berisi `jobRunAsUser` kata sandi harus dihapus ketika tidak lagi digunakan.

Berikan akses ke rahasia kata sandi

Armada Cloud deadline memerlukan akses ke `jobRunAsUser` kata sandi yang disimpan dalam rahasia kata sandi antrian saat antrian dan armada terkait. Sebaiknya gunakan kebijakan sumber daya AWS Secrets Manager untuk memberikan akses ke peran armada. Jika Anda benar-benar mematuhi pedoman ini, lebih mudah untuk menentukan peran armada mana yang memiliki akses ke rahasia.

Untuk memberikan akses ke rahasia

1. Buka konsol AWS Secret Manager ke rahasia.
2. Di bagian Izin sumber daya, tambahkan pernyataan kebijakan formulir:

   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"
    }
  ]
}
```

Cabut akses ke rahasia kata sandi

Ketika armada tidak lagi memerlukan akses ke antrian, hapus akses ke rahasia kata sandi untuk antrian`jobRunAsUser`. Sebaiknya gunakan kebijakan sumber daya AWS Secrets Manager untuk memberikan akses ke peran armada. Jika Anda benar-benar mematuhi pedoman ini, lebih mudah untuk menentukan peran armada mana yang memiliki akses ke rahasia.

Untuk mencabut akses ke rahasia

1. Buka konsol AWS Secret Manager ke rahasia.
2. Di bagian Izin sumber daya, hapus pernyataan kebijakan formulir:

   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"
    }
  ]
}
```

Instal dan konfigurasikan perangkat lunak yang diperlukan untuk pekerjaan

Setelah menyiapkan agen pekerja Deadline Cloud, Anda dapat menyiapkan host pekerja dengan perangkat lunak apa pun yang diperlukan untuk menjalankan pekerjaan.

Saat Anda mengirimkan pekerjaan ke antrian dengan yang terkait`jobRunAsUser`, pekerjaan berjalan sebagai pengguna tersebut. Ketika pekerjaan dikirimkan dengan perintah yang bukan jalur absolut, perintah itu harus ditemukan di `PATH` pengguna itu.

Di Linux, Anda dapat menentukan `PATH` untuk pengguna di salah satu dari berikut ini:

- mereka `~/.bashrc` atau `~/.bash_profile`
- file konfigurasi sistem seperti `/etc/profile.d/*` dan `/etc/profile`
- skrip startup shell:`/etc/bashrc`.

Di Windows, Anda dapat menentukan `PATH` untuk pengguna di salah satu dari berikut ini:

- variabel lingkungan khusus pengguna mereka
- variabel lingkungan seluruh sistem

Instal adaptor alat pembuatan konten digital

Deadline Cloud menyediakan OpenJobDescription adaptor untuk menggunakan aplikasi pembuatan konten digital (DCC) populer. Untuk menggunakan adaptor ini dalam armada yang dikelola pelanggan, Anda harus menginstal perangkat lunak DCC dan adaptor aplikasi. Kemudian, pastikan program yang dapat dieksekusi perangkat lunak tersedia di jalur pencarian sistem (misalnya, dalam variabel `PATH` lingkungan).

Untuk memasang adaptor DCC pada armada yang dikelola pelanggan

1. Buka terminal.
    a. Di Linux, buka terminal sebagai `root` pengguna (atau gunakan`sudo`/`su`)
    b. WindowsAktif, buka prompt perintah administrator atau PowerShell terminal.
2. Instal paket adaptor Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

Mengkonfigurasi kredensial AWS

Fase awal dari siklus hidup pekerja adalah bootstrap. Pada fase ini, perangkat lunak agen pekerja menciptakan pekerja di armada Anda, dan memperoleh AWS kredensil dari peran armada Anda untuk operasi lebih lanjut.

AWS credentials for Amazon EC2

Untuk membuat peran IAM untuk Amazon EC2 dengan izin host pekerja Deadline Cloud

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran di panel navigasi, lalu pilih Buat peran.
3. Pilih AWS layanan.
4. Pilih EC2 sebagai Layanan atau kasus penggunaan, lalu pilih Berikutnya.
5. Untuk memberikan izin yang diperlukan, lampirkan kebijakan `AWSDeadlineCloud-WorkerHost` AWS terkelola.

On-premises AWS credentials

Pekerja lokal Anda menggunakan kredensil untuk mengakses Deadline Cloud. Untuk akses yang paling aman, sebaiknya gunakan IAM Roles Anywhere untuk mengautentikasi pekerja Anda. Untuk informasi selengkapnya, lihat Peran IAM Di Mana Saja.

Untuk pengujian, Anda dapat menggunakan kunci akses pengguna IAM untuk AWS kredensil. Kami menyarankan Anda menetapkan kedaluwarsa untuk pengguna IAM dengan menyertakan kebijakan inline yang membatasi.

> **Important**
>
> Perhatikan peringatan berikut:
>
> - JANGAN gunakan kredensi root akun Anda untuk mengakses AWS sumber daya. Kredensi ini menyediakan akses akun yang tidak terbatas dan sulit dicabut.

- JANGAN menaruh kunci akses literal atau informasi kredensi dalam file aplikasi Anda. Jika Anda melakukannya, Anda membuat risiko secara tidak sengaja mengekspos kredensialnya jika, misalnya, Anda mengunggah proyek ke repositori publik.
- JANGAN sertakan file yang berisi kredensil di area proyek Anda.
- Amankan kunci akses Anda. Jangan berikan kunci akses Anda kepada pihak yang tidak berwenang, bahkan untuk membantu menemukan pengenal akun Anda. Dengan melakukan tindakan ini, Anda mungkin memberi seseorang akses permanen ke akun Anda.
- Ketahuilah bahwa kredensil apa pun yang disimpan dalam file AWS kredensial bersama disimpan dalam teks biasa.

Untuk detail selengkapnya, lihat Praktik terbaik untuk mengelola kunci AWS akses di Referensi AWS Umum.

Mmebuat pengguna IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Pengguna dan pilih Buat pengguna.
3. Beri nama pengguna. Kosongkan kotak centang untuk Menyediakan akses pengguna ke Konsol Manajemen AWS, lalu pilih Berikutnya.
4. Pilih Lampirkan kebijakan secara langsung.
5. Dari daftar kebijakan izin, pilih AWSDeadlineCloud-WorkerHostkebijakan, lalu pilih Berikutnya.
6. Tinjau detail pengguna dan kemudian pilih Buat pengguna.

Batasi akses pengguna ke jendela waktu terbatas

Kunci akses pengguna IAM apa pun yang Anda buat adalah kredensi jangka panjang. Untuk memastikan bahwa kredensil ini kedaluwarsa jika salah penanganan, Anda dapat membuat kredensil ini terikat waktu dengan membuat kebijakan inline yang menentukan tanggal setelah kunci tidak lagi valid.

1. Buka pengguna IAM yang baru saja Anda buat. Di tab Izin, pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.

2. Di editor JSON, tentukan izin berikut. Untuk menggunakan kebijakan ini, ganti nilai `aws:CurrentTime` stempel waktu dalam kebijakan contoh dengan waktu dan tanggal Anda sendiri.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Buat kunci akses

1. Pada halaman detail pengguna, pilih tab Security credentials. Pada bagian Access key, pilih Buat access key.
2. Tunjukkan bahwa Anda ingin menggunakan kunci untuk Lainnya, lalu pilih Berikutnya, lalu pilih Buat kunci akses.
3. Pada halaman Ambil kunci akses, pilih Tampilkan untuk mengungkapkan nilai kunci akses rahasia pengguna Anda. Anda dapat menyalin kredensialnya atau mengunduh file.csv.

Simpan kunci akses pengguna

- Menyimpan kunci akses pengguna dalam file AWS kredensial pengguna agen pada sistem host pekerja:
  - PadaLinux, file tersebut terletak di `~/.aws/credentials`
  - PadaWindows, file tersebut terletak di `%USERPROFILE\.aws\credentials`

Ganti kunci berikut:

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> Ketika Anda tidak lagi membutuhkan pengguna IAM ini, kami sarankan Anda menghapusnya agar selaras dengan praktik terbaik AWS keamanan. Kami menyarankan Anda meminta pengguna manusia Anda untuk menggunakan kredensi sementara AWS IAM Identity Centersaat mengakses. AWS

Aliran data host pekerja untuk armada yang dikelola pelanggan

Topik ini menjelaskan koneksi jaringan yang dibuat oleh host pekerja AWS Deadline Cloud (Deadline Cloud) selama operasi, termasuk titik akhir yang dihubungi, protokol yang digunakan, dan data yang ditransmisikan. Informasi ini berlaku untuk pekerja armada yang dikelola pelanggan (CMF), termasuk instans Amazon Elastic Compute Cloud (Amazon EC2) dan pekerja lokal. Gunakan informasi ini untuk mengonfigurasi aturan firewall, membuat titik akhir VPC, melakukan audit keamanan, atau merencanakan kebijakan jaringan untuk host pekerja Anda. Untuk informasi tentang jaringan armada yang dikelola layanan, lihat. Inter-network privasi lalu lintas

Semua komunikasi pekerja hanya keluar. Host pekerja memulai semua koneksi—Anda tidak perlu mengizinkan koneksi masuk apa pun. Semua koneksi menggunakan HTTPS (TLS 1.2 atau yang lebih baru) melalui port 443.

Topik ini mencakup bagian-bagian berikut:

- the section called “Titik akhir dan protokol”
- the section called “Operasi API yang digunakan oleh pekerja”
- the section called “Data lain yang dikirimkan”
- the section called “Opsi konektivitas pribadi”

Titik akhir dan protokol

Tabel berikut mencantumkan titik akhir AWS layanan yang terhubung dengan host pekerja selama operasi. Untuk daftar lengkap titik akhir regional untuk setiap layanan, lihat titik akhir Layanan dan kuota di Referensi Umum.AWS

Referensi titik akhir host pekerja

AWS layanan	Titik akhir	Pelabuhan/ Protokol	Tujuan	Diperlukan
Batas Waktu Cloud (penjadwalan)	`scheduling.deadline. [Region].amazonaws.com`	443/ HTTPS	Pendaftaran pekerja, pemungutan suara tugas, pembaruan status, pertukaran kredensyal, pengambilan entitas pekerjaan. Lihat the section called “Operasi API yang digunakan oleh pekerja”.	Selalu
CloudWatch Log Amazon (CloudWatch Log)	`logs.[Region].amazonas.com`	443/ HTTPS	Agen pekerja dan pengiriman log sesi.	Selalu
Amazon Simple Storage Service (Amazon S3)	`s3.[Region].amazonaws.com`	443/ HTTPS	Unggah dan unduh lampiran pekerjaan.	Jika menggunakan lampiran pekerjaan

Jika pekerjaan Anda menggunakan AWS layanan lain, Anda mungkin juga perlu mengizinkan koneksi keluar ke titik akhir layanan tersebut.

Operasi API yang digunakan oleh pekerja

Semua operasi API berikut menggunakan `scheduling.deadline.[Region].amazonaws.com` endpoint. Untuk skema permintaan dan respons lengkap dari setiap operasi, lihat Referensi API Cloud Deadline.

Fase bootstrap

Ketika tuan rumah pekerja dimulai, agen pekerja mendaftar dengan armada. Kredensyal bootstrap memerlukan izin dalam kebijakan `AWSDeadlineCloud-WorkerHost` AWS terkelola, atau izin khusus yang setara. Fase bootstrap menggunakan operasi API berikut:

- `CreateWorker`— Mendaftarkan pekerja dengan armada. Mengirim nama host dan alamat IP. Menerima ID pekerja.
- `AssumeFleetRoleForWorker`— Memperoleh kredensi peran armada. Menerima AWS kredensi sementara yang digunakan agen pekerja untuk operasi selanjutnya.

Fase operasional

Setelah bootstrap, agen pekerja melakukan polling untuk sesi kerja dan proses. Peran armada memerlukan izin dalam kebijakan `AWSDeadlineCloud-FleetWorker` AWS terkelola, atau izin khusus yang setara, dan menggunakan operasi API berikut:

- `UpdateWorker`— Memperbarui status pekerja, misalnya `STOPPED` selama shutdown.
- `UpdateWorkerSchedule`— Jajak pendapat untuk tugas kerja. Mengirim pembaruan status tindakan sesi termasuk status penyelesaian, persentase kemajuan, pesan kemajuan, dan hash manifes keluaran. Menerima sesi yang ditetapkan (ID pekerjaan, ID antrian, tindakan sesi, konfigurasi log), permintaan pembatalan, status pekerja yang diinginkan, dan interval pembaruan.
- `BatchGetJobEntity`— Mengambil detail pekerjaan untuk pekerjaan yang ditugaskan. Mengirim pengidentifikasi entitas pekerjaan. Menerima detail pekerjaan, detail lingkungan, dan detail lampiran pekerjaan.
- `AssumeFleetRoleForWorker`— Menyegarkan kredensi peran armada secara berkala.
- `AssumeQueueRoleForWorker`— Memperoleh kredensi peran antrian yang dicakup ke antrian tertentu. Pekerja menggunakan kredensyal ini untuk mengakses lampiran pekerjaan di Amazon S3.

Data lain yang dikirimkan

Selain operasi API penjadwalan Deadline Cloud, host pekerja mengirimkan data berikut ke layanan lain AWS :

Data log

Agen pekerja mengirimkan log agen pekerja dan log sesi (stdout dan stderr dari proses pekerjaan) ke CloudWatch Log menggunakan operasi API. `PutLogEvents`

Lampiran Job

Pekerja mentransfer file input dan output melalui Amazon S3 menggunakan `GetObject` dan operasi `PutObject` API. Pekerja menggunakan kredensyal peran antrian yang diperoleh `AssumeQueueRoleForWorker` untuk akses ini.

Telemetri (opsional)

Agen pekerja mengirimkan metrik operasional seperti laporan kerusakan. Anda dapat memilih keluar dari koleksi telemetri. Untuk informasi selengkapnya, lihat Memilih keluar.

Opsi konektivitas pribadi

Anda dapat menggunakan AWS PrivateLink untuk menjaga lalu lintas antara host pekerja CMF dan Deadline Cloud dalam VPC Anda, tanpa melintasi internet publik. Untuk pekerja lokal, Anda dapat menggabungkan AWS PrivateLink dengan AWS Direct Connect (Direct Connect) atau koneksi VPN. Lihat informasi yang lebih lengkap di Akses AWS Deadline Cloud menggunakan titik akhir antarmuka (AWS PrivateLink).

Uji konfigurasi host pekerja Anda

Setelah Anda menginstal agen pekerja, menginstal perangkat lunak yang diperlukan untuk memproses pekerjaan Anda, dan mengonfigurasi AWS kredensi untuk agen pekerja, Anda harus menguji bahwa instalasi dapat memproses pekerjaan Anda sebelum membuat AMI untuk armada Anda. Anda harus menguji yang berikut:

- Agen pekerja Deadline Cloud dikonfigurasi dengan benar untuk dijalankan sebagai layanan sistem.
- Bahwa pekerja melakukan polling antrian terkait untuk bekerja.
- Bahwa pekerja berhasil memproses pekerjaan yang dikirim ke antrian yang terkait dengan armada.

Setelah Anda menguji konfigurasi dan berhasil memproses pekerjaan perwakilan, Anda dapat menggunakan pekerja yang dikonfigurasi untuk membuat AMI untuk EC2 pekerja Amazon, atau sebagai model untuk pekerja lokal Anda.

> **Note**
>
> Jika Anda menguji konfigurasi host pekerja dari armada penskalaan otomatis, Anda mungkin mengalami kesulitan menguji pekerja Anda dalam situasi berikut:
>
> - Jika tidak ada pekerjaan dalam antrian, Deadline Cloud menghentikan agen pekerja segera setelah pekerja mulai.
> - Jika agen pekerja dikonfigurasi untuk mematikan host saat berhenti, agen mematikan mesin jika tidak ada pekerjaan dalam antrian.
>
> Untuk menghindari masalah ini, gunakan armada pementasan yang tidak menskalakan otomatis untuk mengonfigurasi dan menguji pekerja Anda. Setelah menguji host pekerja, pastikan untuk mengatur ID armada yang benar sebelum memanggang AMI.

Untuk menguji konfigurasi host pekerja Anda

1. Jalankan agen pekerja dengan memulai layanan sistem operasi.

   Linux

   Dari shell root jalankan perintah berikut:

   ```
   systemctl start deadline-worker
   ```

   Windows

   Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

   ```
   sc.exe start DeadlineWorker
   ```

2. Pantau pekerja untuk memastikannya dimulai dan polling untuk pekerjaan.

   Linux

   Dari shell root jalankan perintah berikut:

```
systemctl status deadline-worker
```

Perintah harus mengembalikan respons seperti:

```
Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
```

Jika respons tidak terlihat seperti itu, periksa file log menggunakan perintah berikut:

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

```
sc.exe query DeadlineWorker
```

Perintah harus mengembalikan respons seperti:

```
STATE   : 4 RUNNING
```

Jika respons tidak berisi`RUNNING`, periksa file log pekerja. Buka dan administrator PowerShell prompt dan jalankan perintah berikut:

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. Kirim pekerjaan ke antrian yang terkait dengan armada Anda. Pekerjaan harus mewakili pekerjaan yang diproses armada.
4. Pantau kemajuan pekerjaan menggunakan monitor Deadline Cloud atau CLI. Jika pekerjaan gagal, periksa sesi dan log pekerja.
5. Perbarui konfigurasi host pekerja sesuai kebutuhan hingga pekerjaan selesai dengan sukses.
6. Ketika pekerjaan uji berhasil, Anda dapat menghentikan pekerja:

   Linux

   Dari shell root jalankan perintah berikut:

```
systemctl stop deadline-worker
```

Windows

Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

```
sc.exe stop DeadlineWorker
```

Buat sebuah Amazon Machine Image

Untuk membuat Amazon Machine Image (AMI) untuk digunakan dalam armada yang dikelola pelanggan (CMF EC2) Amazon Elastic Compute Cloud (Amazon), selesaikan tugas di bagian ini. Anda harus membuat EC2 instance Amazon sebelum melanjutkan. Untuk informasi selengkapnya, lihat Meluncurkan instans Anda di Panduan EC2 Pengguna Amazon untuk Instans Linux.

Important

Membuat AMI membuat snapshot dari volume terlampir EC2 instans Amazon. Perangkat lunak apa pun yang diinstal pada instance tetap ada sehingga instance, yang digunakan kembali saat Anda meluncurkan instance dari AMI. Kami merekomendasikan untuk mengadopsi strategi patching dan secara teratur memperbarui yang baru AMI dengan perangkat lunak yang diperbarui sebelum mendaftar ke armada Anda.

Siapkan EC2 instans Amazon

Sebelum Anda membangun sebuah AMI, Anda harus menghapus status pekerja. Negara pekerja tetap ada di antara peluncuran agen pekerja. Jika keadaan ini berlanjut ke AMI, maka semua instance yang diluncurkan darinya akan berbagi status yang sama.

Kami juga menyarankan Anda menghapus file log yang ada. File log dapat tetap berada di EC2 instans Amazon saat Anda menyiapkan AMI. Menghapus file-file ini meminimalkan kebingungan saat mendiagnosis kemungkinan masalah dalam armada pekerja yang menggunakan AMI.

Anda juga harus mengaktifkan layanan sistem agen pekerja sehingga agen pekerja Deadline Cloud diluncurkan saat Amazon EC2 dimulai.

Terakhir, kami sarankan Anda mengaktifkan auto shutdown agen pekerja. Hal ini memungkinkan armada pekerja untuk meningkatkan skala saat dibutuhkan dan dimatikan saat pekerjaan rendering selesai. Penskalaan otomatis ini membantu memastikan Anda hanya menggunakan sumber daya sesuai kebutuhan.

Untuk menyiapkan EC2 instance Amazon

1. Buka EC2 konsol Amazon.
2. Luncurkan EC2 instans Amazon. Untuk informasi selengkapnya, lihat Meluncurkan instans Anda.
3. Siapkan host untuk terhubung ke penyedia identitas Anda (iDP), lalu pasang sistem file bersama yang dibutuhkannya.
4. Ikuti tutorial untukInstal Deadline Agen pekerja Cloud, kemudianKonfigurasikan agen pekerja, danBuat pengguna dan grup pekerjaan.
5. Jika Anda sedang mempersiapkan AMI berdasarkan Amazon Linux 2023 untuk menjalankan perangkat lunak yang kompatibel dengan Platform Referensi VFX, Anda perlu memperbarui beberapa persyaratan. Untuk selengkapnya, lihat Kompatibilitas Platform Referensi VFX di Panduan Pengguna Cloud AWS Deadline.
6. Buka terminal.
   a. Di Linux, buka terminal sebagai `root` pengguna (atau gunakan`sudo`/`su`)
   b. Pada Windows, buka prompt perintah administrator atau PowerShell terminal.
7. Pastikan layanan pekerja tidak berjalan dan dikonfigurasi untuk memulai saat boot:
   a. Di Linux, jalankan

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. Pada Windows, lari

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. Hapus status pekerja.
   a. Di Linux, jalankan

```
rm -rf /var/lib/deadline/*
```

b. Pada Windows, lari

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Hapus file log.

a. Di Linux, jalankan

```
rm -rf /var/log/amazon/deadline/*
```

b. Pada Windows, lari

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. Pada Windows, disarankan untuk menjalankan aplikasi Amazon EC2 Launch Settings yang ditemukan di menu Start untuk menyelesaikan persiapan host akhir dan shutdown instance.

> **Note**
>
> Anda HARUS memilih Shutdown tanpa Sysprep dan tidak pernah memilih Shutdown dengan Sysprep. Mematikan dengan Sysprep akan menyebabkan semua pengguna lokal menjadi tidak dapat digunakan. Untuk informasi selengkapnya, lihat bagian Sebelum Anda Memulai topik Buat AMI kustom dari Panduan Pengguna untuk Instans Windows.

Membangun AMI

Untuk membangun AMI

1. Buka EC2 konsol Amazon.
2. Pilih Instans di panel navigasi, lalu pilih instans Anda.
3. Pilih status Instance, lalu Stop instance.
4. Setelah instance Dihentikan, pilih Tindakan.
5. Pilih Gambar dan template, lalu Buat gambar.
6. Masukkan nama Gambar.
7. (Opsional) Masukkan deskripsi untuk gambar Anda.

8. Pilih Buat citra.

Buat infrastruktur armada dengan grup Amazon EC2 Auto Scaling

Bagian ini menjelaskan cara membuat armada Auto Scaling Amazon EC2.

Gunakan template CloudFormation YAMB di bawah ini untuk membuat grup Amazon EC2 Auto Scaling (Auto Scaling), Amazon Virtual Private Cloud (Amazon VPC) dengan dua subnet, profil instans, dan peran akses instans. Ini diperlukan untuk meluncurkan instance menggunakan Auto Scaling di subnet.

Anda harus meninjau dan memperbarui daftar jenis instance agar sesuai dengan kebutuhan rendering Anda.

Untuk penjelasan lengkap tentang resource dan parameter yang digunakan dalam template CloudFormation YAMB, lihat referensi tipe resource Deadline Cloud di AWS CloudFormation Panduan Pengguna.

Untuk membuat armada Auto Scaling Amazon EC2

1. Gunakan contoh berikut untuk membuat CloudFormation template yang mendefinisikan`FarmID`,`FleetID`, dan `AMIId` parameter. Simpan template ke `.YAML` file di komputer lokal Anda.

```
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. Buka CloudFormation konsol di https://console.aws.amazon.com/cloudformation.

   Gunakan CloudFormation konsol untuk membuat tumpukan menggunakan instruksi untuk mengunggah file template yang Anda buat. Untuk informasi selengkapnya, lihat Membuat tumpukan di CloudFormation konsol di Panduan AWS CloudFormation Pengguna.

Note

- Kredensional dari peran IAM yang dilampirkan ke instans Amazon EC2 pekerja Anda tersedia untuk semua proses yang berjalan pada pekerja tersebut, yang mencakup pekerjaan. Pekerja harus memiliki hak istimewa paling sedikit untuk beroperasi: `deadline:CreateWorker` dan `deadline:AssumeFleetRoleForWorker.`
- Agen pekerja memperoleh kredensil untuk peran antrian dan mengonfigurasinya untuk digunakan dengan menjalankan pekerjaan. Peran profil instans Amazon EC2 tidak boleh menyertakan izin yang diperlukan oleh pekerjaan Anda.

Skalakan otomatis armada Amazon EC2 Anda dengan fitur rekomendasi skala Deadline Cloud

Deadline Cloud memanfaatkan grup Amazon EC2 Auto Scaling (Auto Scaling) untuk menskalakan armada yang dikelola pelanggan (CMF) Amazon EC2 secara otomatis. Anda perlu mengonfigurasi mode armada serta menerapkan infrastruktur yang diperlukan di akun Anda untuk membuat skala otomatis armada Anda. Infrastruktur yang Anda gunakan akan berfungsi untuk semua armada, jadi Anda hanya perlu mengaturnya sekali.

Alur kerja dasarnya adalah: Anda mengonfigurasi mode armada Anda ke skala otomatis, lalu Deadline Cloud akan mengirimkan EventBridge acara untuk armada tersebut setiap kali ukuran armada yang direkomendasikan berubah (satu peristiwa berisi id armada, ukuran armada yang direkomendasikan, dan metadata lainnya). Anda akan memiliki EventBridge aturan untuk memfilter acara yang relevan dan meminta Lambda untuk mengkonsumsinya. Lambda akan berintegrasi dengan Amazon EC2 Auto `AutoScalingGroup` Scaling untuk menskalakan armada Amazon EC2 secara otomatis.

Setel mode armada ke **`EVENT_BASED_AUTO_SCALING`**

Konfigurasikan mode armada Anda ke`EVENT_BASED_AUTO_SCALING`. Anda dapat menggunakan konsol untuk melakukan ini, atau menggunakan AWS CLI untuk langsung memanggil `CreateFleet` atau `UpdateFleet` API. Setelah mode dikonfigurasi, Deadline Cloud mulai mengirimkan EventBridge peristiwa setiap kali ukuran armada yang direkomendasikan berubah.

- Contoh `UpdateFleet` perintah:

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- Contoh `CreateFleet` perintah:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

Berikut ini adalah contoh yang `configuration.json` digunakan dalam perintah CLI di atas (`--configuration file://configuration.json`.

- Untuk mengaktifkan Auto Scaling pada armada Anda, Anda harus mengatur mode ke. `EVENT_BASED_AUTO_SCALING`
- `workerCapabilities`Ini adalah nilai default yang ditetapkan ke CMF saat Anda membuatnya. Anda dapat mengubah nilai-nilai ini jika Anda perlu meningkatkan sumber daya yang tersedia untuk CMF Anda.

Setelah Anda mengonfigurasi mode armada, Deadline Cloud mulai memancarkan acara rekomendasi ukuran armada untuk armada tersebut.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
```

```
        }
    }
}
```

Terapkan tumpukan Auto Scaling menggunakan template CloudFormation

Anda dapat mengatur EventBridge aturan untuk memfilter peristiwa, Lambda untuk mengkonsumsi peristiwa dan mengontrol Auto Scaling, dan antrean SQS untuk menyimpan peristiwa yang belum diproses. Gunakan CloudFormation template berikut untuk menyebarkan semuanya dalam tumpukan. Setelah Anda berhasil menyebarkan sumber daya, Anda dapat mengirimkan pekerjaan dan armada akan meningkat secara otomatis.

```
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
```

Lakukan pemeriksaan kesehatan armada

Setelah membuat armada Anda, Anda harus membuat pemeriksaan kesehatan khusus untuk memastikan armada Anda tetap sehat dan bebas dari keadaan macet untuk membantu mencegah biaya yang tidak perlu. Lihat Menerapkan pemeriksaan kesehatan armada Cloud Deadline di. GitHub Pemeriksaan kesehatan dapat menurunkan risiko perubahan yang tidak disengaja pada templat peluncuranAmazon Machine Image, atau konfigurasi jaringan Anda yang berjalan tanpa terdeteksi.

Konfigurasikan dan gunakan armada yang dikelola layanan Deadline Cloud

Service-managed fleet (SMF) adalah kumpulan pekerja yang dikelola oleh Deadline Cloud. SMF menghilangkan kebutuhan untuk mengelola penskalaan armada untuk permintaan pemrosesan atau mengurangi ukuran armada setelah tugas selesai.

Ketika SMF dikaitkan dengan antrian menggunakan lingkungan antrian conda default, Deadline Cloud mengonfigurasi pekerja di armada dengan paket perangkat lunak yang sesuai. Untuk aplikasi mitra yang didukung, lihat Lingkungan antrian conda default di Panduan Pengguna Cloud AWS Batas Waktu.

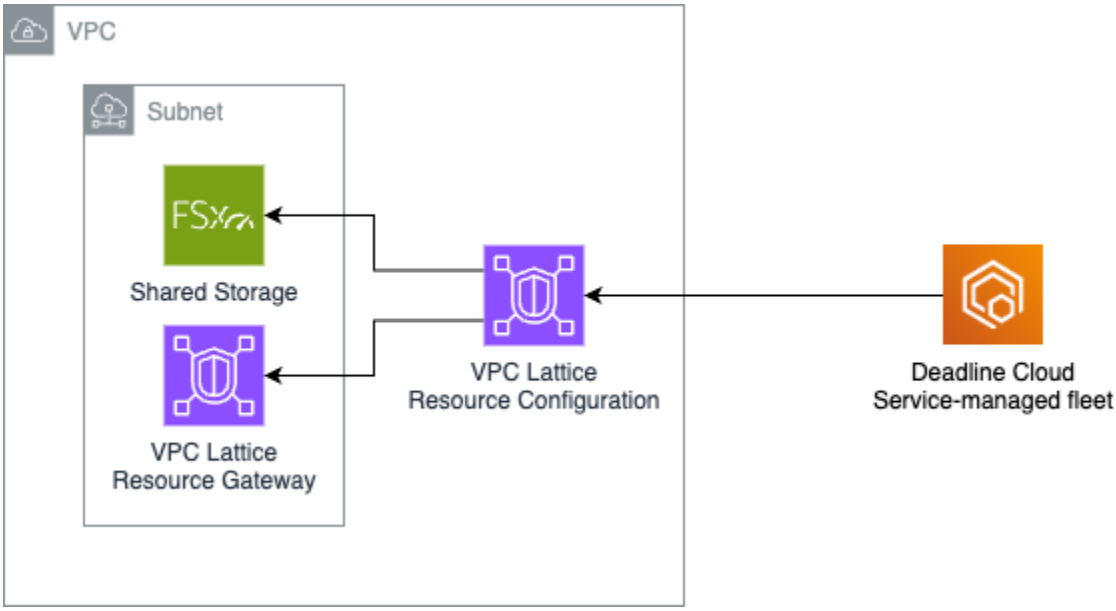
Dalam kebanyakan kasus, Anda tidak perlu mengubah SMF untuk memproses beban kerja Anda. Namun, beberapa situasi mungkin mengharuskan Anda membuat perubahan pada armada Anda.

> **Note**
>
> Untuk menginstal perangkat lunak khusus pada pekerja yang menggunakan skrip konfigurasi host, lihatJalankan skrip konfigurasi host dengan hak administrator.

Topik

Hubungkan sumber daya VPC ke SMF Anda dengan titik akhir sumber daya VPC

Dengan titik akhir sumber daya Amazon VPC untuk armada yang dikelola layanan Deadline Cloud (SMF), Anda dapat menghubungkan sumber daya VPC Anda seperti sistem file jaringan (NFS), server lisensi, dan database dengan pekerja Deadline Cloud Anda. Fitur ini memungkinkan Anda memanfaatkan platform Deadline Cloud yang dikelola sepenuhnya sambil berintegrasi dengan infrastruktur yang ada dalam VPC.



> **Tip**
>
> Untuk CloudFormation templat referensi yang menyiapkan FSx klaster Amazon dan menghubungkannya ke armada yang dikelola layanan, lihat smf_vpc_fsx di repositori sampel Deadline Cloud pada. GitHub

Cara kerja titik akhir sumber daya VPC

Titik akhir sumber daya VPC menggunakan VPC Lattice untuk membuat koneksi aman antara pekerja SMF Cloud Deadline Anda dan sumber daya di VPC Anda. Koneksinya searah, artinya pekerja dapat membuat koneksi ke sumber daya di VPC Anda dan mentransfer data bolak-balik, tetapi sumber daya di VPC Anda tidak dapat membuat koneksi ke pekerja.

Saat Anda menghubungkan sumber daya VPC ke armada yang dikelola layanan Deadline Cloud, pekerja Anda dapat mengakses sumber daya di VPC menggunakan nama domain pribadi. Selain itu, arus lalu lintas dari pekerja ke sumber daya VPC Anda melalui VPC Lattice, dan sumber daya di VPC Anda melihat lalu lintas yang berasal dari gateway sumber daya VPC Lattice.

Untuk mempelajari lebih lanjut, lihat panduan pengguna VPC Lattice.

Prasyarat

Sebelum menghubungkan sumber daya VPC ke armada yang dikelola layanan Deadline Cloud, pastikan Anda memiliki hal-hal berikut:

- AWS Akun dengan VPC yang berisi sumber daya yang ingin Anda sambungkan.
- Izin IAM untuk membuat dan mengelola sumber daya VPC Lattice.
- Peternakan Cloud Deadline dengan setidaknya satu armada yang dikelola layanan.
- Sumber daya VPC yang ingin Anda buat dapat diakses (FSx, NFS, server lisensi, dll.).

Siapkan titik akhir sumber daya VPC

Untuk menyiapkan titik akhir sumber daya VPC, Anda perlu membuat sumber daya di VPC Lattice dan AWS RAM, lalu menghubungkan sumber daya tersebut ke armada Anda di Deadline Cloud. Untuk menyiapkan titik akhir sumber daya VPC untuk SMF Anda, selesaikan langkah-langkah berikut.

1. Untuk membuat gateway sumber daya di VPC Lattice, lihat Membuat gateway sumber daya di panduan pengguna VPC Lattice.
2. Untuk membuat konfigurasi sumber daya di VPC Lattice, lihat Membuat konfigurasi sumber daya di panduan pengguna VPC Lattice.
3. Untuk berbagi sumber daya dengan armada Deadline Cloud Anda, buat pembagian sumber daya di AWS RAM. Lihat Membuat berbagi sumber daya untuk instruksi.

   Saat membuat pembagian sumber daya, untuk Prinsipal, pilih Prinsipal layanan dari menu tarik-turun, lalu masukkan. **`fleets.deadline.amazonaws.com`**
4. Untuk menghubungkan konfigurasi sumber daya dengan armada Deadline Cloud Anda, selesaikan langkah-langkah berikut.

   a. Jika Anda belum melakukannya, buka konsol Deadline Cloud.

   b. Di panel navigasi, pilih Peternakan, lalu pilih peternakan Anda.

   c. Pilih tab Armada, lalu pilih armada Anda.

   d. Pilih tab Konfigurasi.

   e. Di bawah titik akhir sumber daya VPC, pilih Edit.

   f. Pilih konfigurasi sumber daya yang Anda buat, lalu pilih Simpan perubahan.

Mengakses sumber daya VPC Anda

Setelah menghubungkan sumber daya VPC Anda ke armada Anda, pekerja dapat mengaksesnya menggunakan nama domain pribadi dalam format berikut: `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Domain ini bersifat pribadi dan hanya dapat diakses oleh pekerja (bukan dari internet atau workstation Anda).

Untuk memasang atau mengonfigurasi akses ke sumber daya VPC pada pekerja Anda, gunakan skrip konfigurasi host. Skrip konfigurasi host berjalan dengan hak administrator saat pekerja memulai, memungkinkan Anda memasang sistem file, mengonfigurasi pengaturan jaringan, atau melakukan tugas penyiapan lainnya.

Otentikasi dan keamanan

Untuk sumber daya yang memerlukan otentikasi, simpan kredensil dengan aman di AWS Secrets Manager, akses rahasia dari skrip konfigurasi host atau skrip pekerjaan Anda, dan terapkan izin sistem file yang sesuai untuk mengontrol akses. Pertimbangkan implikasi keamanan saat berbagi sumber daya di beberapa armada. Misalnya, jika dua armada terhubung ke penyimpanan bersama yang sama, pekerjaan yang berjalan pada satu armada mungkin dapat mengakses aset yang dibuat dari armada lainnya.

Pertimbangan teknis

Saat menggunakan titik akhir sumber daya VPC, pertimbangkan hal berikut:

- Koneksi hanya dapat dimulai dari pekerja ke sumber daya VPC, bukan dari sumber daya VPC ke pekerja.
- Setelah dibuat, koneksi tetap ada sampai diatur ulang, bahkan jika konfigurasi sumber daya terputus.
- Koneksi VPC Lattice menangani koneksi antara Availability Zones secara otomatis tanpa biaya tambahan. Gateway sumber daya Anda harus berbagi Availability Zone dengan resource VPC Anda, jadi sebaiknya konfigurasikan gateway sumber daya untuk menjangkau semua Availability Zone.
- Lalu lintas melalui titik akhir sumber daya VPC menggunakan Network Address Translation (NAT), yang tidak kompatibel dengan semua kasus penggunaan. Misalnya, Microsoft Active Directory (AD) tidak dapat terhubung melalui NAT.

Untuk informasi selengkapnya tentang kuota Kisi VPC, lihat Kuota untuk Kisi VPC.

Pemecahan masalah

Jika Anda mengalami masalah dengan titik akhir sumber daya VPC, periksa yang berikut ini.

- Jika Anda menerima pesan kesalahan seperti “mount.nfs: akses ditolak oleh server saat pemasangan,” Anda mungkin perlu memperbarui konfigurasi klien volume NFS Anda.
- Verifikasi pengaturan konfigurasi sumber daya Anda dengan menguji dari instans Amazon EC2 atau AWS CloudShell di VPC Anda.
- Uji koneksi Deadline Cloud Anda dengan pekerjaan CLI sederhana. Untuk informasi selengkapnya, lihat contoh Deadline Cloud di GitHub.
- Periksa pengaturan pada grup keamanan gateway sumber daya jika Anda mengalami kegagalan koneksi.
- Aktifkan log akses VPC untuk memantau koneksi.

Gunakan lampiran pekerjaan dengan armada yang dikelola layanan

Lampiran pekerjaan mentransfer file antara workstation dan pekerja Deadline Cloud menggunakan Amazon Simple Storage Service (Amazon S3). Anda dapat menggunakan lampiran pekerjaan sendiri atau bersama-sama dengan penyimpanan bersama untuk melampirkan data tambahan ke pekerjaan yang tidak dibagikan dengan pekerjaan lain, seperti skrip pekerjaan, file konfigurasi, atau aset proyek yang disimpan secara lokal.

Untuk informasi tentang cara kerja lampiran lowongan, lihat Lampiran pekerjaan di Panduan Pengguna Cloud Batas Waktu. Untuk detail tentang menentukan file input dan output dalam bundel pekerjaan, lihat. Gunakan lampiran pekerjaan untuk berbagi file

Pilih mode sistem file

Saat mengirimkan pekerjaan dengan lampiran, Anda dapat memilih cara pekerja memuat file dari Amazon S3 dengan menyetel properti: `fileSystem`

- DISALIN (default) — Download semua file ke disk lokal sebelum tugas dimulai. Terbaik ketika setiap tugas membutuhkan sebagian besar file input.

- VIRTUAL - Memasang sistem file virtual yang mengunduh file sesuai permintaan. Terbaik ketika tugas hanya membutuhkan subset file input. Hanya tersedia untuk pekerja SMF Linux.

> **Important**
>
> Caching mode VIRTUAL dapat meningkatkan konsumsi memori dan tidak dioptimalkan untuk semua beban kerja. Kami menyarankan Anda menguji beban kerja Anda sebelum menjalankan pekerjaan produksi.

Untuk informasi mendetail tentang mengonfigurasi mode sistem file, lihat Sistem file virtual di Panduan Pengguna Cloud Tenggat Waktu.

Optimalkan kinerja transfer

Kecepatan sinkronisasi file dari Amazon S3 ke pekerja SMF bergantung pada konfigurasi volume Amazon Elastic Block Store (Amazon EBS) Elastic Block Store (Amazon EBS) armada Anda. Secara default, pekerja SMF menggunakan volume Amazon EBS gp3 dengan pengaturan kinerja dasar. Untuk beban kerja dengan file input besar atau banyak file kecil, Anda dapat meningkatkan kecepatan transfer dengan meningkatkan throughput Amazon EBS dan pengaturan IOPS. Anda dapat memperbarui pengaturan ini menggunakan AWS Command Line Interface (AWS CLI).

Throughput (MiB/s)

Tingkat di mana data dapat dibaca dari atau ditulis ke volume. Defaultnya adalah 125 MiB/s, maximum is 1,000 MiB/s untuk volume gp3. Peningkatan untuk transfer file berurutan besar.

IOPS

Operasi input/output per detik. Default adalah 3.000 IOPS, maksimum 16.000 IOPS untuk volume gp3. Tingkatkan saat mentransfer banyak file kecil.

> **Note**
>
> Meningkatkan throughput Amazon EBS dan IOPS meningkatkan biaya armada. Untuk informasi harga, lihat harga Deadline Cloud.

Untuk memperbarui setelan Amazon EBS pada armada yang sudah ada menggunakan AWS CLI

- Jalankan perintah berikut:

```
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"}
    }
  }'
```

Unduh output pekerjaan

Setelah pekerjaan Anda selesai, unduh file output menggunakan Deadline Cloud CLI atau monitor Deadline Cloud (monitor AWS Deadline Cloud):

```
deadline job download-output --job-id job-0123456789abcdef0
```

Untuk mengunduh output secara otomatis saat pekerjaan selesai, lihat Unduhan otomatis di Panduan Pengguna Cloud Tenggat Waktu.

Menyebarkan dan mengkonfigurasi perangkat lunak khusus pada pekerja

AWS Deadline Cloud menyediakan beberapa metode untuk menerapkan dan mengonfigurasi perangkat lunak, plugin, dan alat khusus pada pekerja Anda. Metode yang Anda pilih tergantung pada kebutuhan Anda, seperti apakah Anda memerlukan hak administrator, seberapa sering perangkat lunak berubah, dan apakah perangkat lunak harus tersedia untuk semua pekerjaan atau hanya pekerjaan tertentu.

Pilih metode penerapan

Gunakan tabel berikut untuk memilih metode penyebaran yang tepat untuk kasus penggunaan Anda.

Kriteria	Lingkungan antrian	Skrip konfigurasi host	Paket conda kustom
Hak istimewa administrator diperlukan	Tidak	Ya	Tidak
Saat berjalan	Sesi dimulai	Startup pekerja	Sesi dimulai
Lingkup	Per antrian atau pekerjaan	Semua pekerja di armada	Per antrian atau pekerjaan
Dapat dikontrol dengan pengajuan pekerjaan	Ya	Tidak	Ya
Kompleksitas pengaturan	Rendah	Sedang	Tinggi
Terbaik untuk	Plugin sederhana , skrip, variabel lingkungan	Driver sistem, Docker, dudukan penyimpanan	Aplikasi kompleks dengan dependens i

Panduan keputusan cepat:

- Butuh hak administrator atau root? Gunakan skrip konfigurasi host.

- Plugin atau skrip sederhana tanpa hak admin? Gunakan lingkungan antrian.
- Aplikasi kompleks dengan kebutuhan kontrol versi? Buat paket conda kustom.

Konfigurasikan pekerjaan menggunakan lingkungan antrian

AWS Deadline Cloud menggunakan lingkungan antrian untuk mengonfigurasi perangkat lunak pada pekerja Anda. Lingkungan memungkinkan Anda untuk melakukan tugas yang memakan waktu, seperti mengatur dan merobohkan, sekali untuk semua tugas dalam satu sesi. Ini mendefinisikan tindakan untuk dijalankan pada pekerja ketika memulai atau menghentikan sesi. Anda dapat mengonfigurasi lingkungan untuk antrian, pekerjaan yang berjalan dalam antrian, dan langkah-langkah individual untuk suatu pekerjaan.

Anda mendefinisikan lingkungan sebagai lingkungan antrian atau lingkungan kerja. Buat lingkungan antrian dengan konsol Deadline Cloud atau dengan tenggat waktu: CreateQueueEnvironment operasi dan tentukan lingkungan pekerjaan di templat pekerjaan pekerjaan yang Anda kirimkan. Mereka mengikuti spesifikasi Open Job Description (OpenJD) untuk lingkungan. Untuk detailnya, lihat <Environment>spesifikasi OpenJDhttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment di. GitHub

Selain a `name` dan`description`, setiap lingkungan berisi dua bidang yang menentukan lingkungan pada host. File tersebut adalah:

- `script`— Tindakan yang diambil ketika lingkungan ini dijalankan pada pekerja.
- `variables`- Satu set name/value pasangan variabel lingkungan yang diatur saat memasuki lingkungan.

Anda harus menetapkan setidaknya satu `script` atau`variables`.

Anda dapat menentukan lebih dari satu lingkungan dalam template pekerjaan Anda. Setiap lingkungan diterapkan dalam urutan yang tercantum dalam template. Anda dapat menggunakan ini untuk membantu mengelola kompleksitas lingkungan Anda.

Lingkungan antrian default untuk Deadline Cloud menggunakan manajer paket conda untuk memuat perangkat lunak ke lingkungan, tetapi Anda dapat menggunakan manajer paket lainnya. Lingkungan default mendefinisikan dua parameter untuk menentukan perangkat lunak yang harus dimuat. Variabel ini diatur oleh pengirim yang disediakan oleh Deadline Cloud, meskipun Anda dapat mengaturnya dalam skrip dan aplikasi Anda sendiri yang menggunakan lingkungan default. File tersebut adalah:

- `CondaPackages`— Daftar spesifikasi paket conda yang dipisahkan ruang untuk dipasang untuk pekerjaan itu. Misalnya, pengirim Blender akan menambahkan `blender=3.6` bingkai render di Blender 3.6.
- `CondaChannels`— Daftar saluran conda yang dipisahkan ruang untuk menginstal paket dari. Untuk armada yang dikelola layanan, paket diinstal dari saluran. `deadline-cloud` Anda dapat menambahkan saluran lain.

Kontrol lingkungan kerja dengan lingkungan antrian OpenJD

Anda dapat menentukan lingkungan yang disesuaikan untuk pekerjaan rendering Anda menggunakan lingkungan antrian. Lingkungan antrian adalah template yang mengontrol variabel lingkungan, pemetaan file, dan pengaturan lain untuk pekerjaan yang berjalan dalam antrian tertentu. Ini memungkinkan Anda untuk menyesuaikan lingkungan eksekusi untuk pekerjaan yang dikirimkan ke antrian dengan persyaratan beban kerja Anda. AWS Deadline Cloud menyediakan tiga level bersarang di mana Anda dapat menerapkan lingkungan Open Job Description (OpenJD): antrian, pekerjaan, dan langkah. Dengan mendefinisikan lingkungan antrian, Anda dapat memastikan kinerja yang konsisten dan dioptimalkan untuk berbagai jenis pekerjaan, merampingkan alokasi sumber daya, dan menyederhanakan manajemen antrian.

Lingkungan antrian adalah template yang Anda lampirkan ke antrian di AWS akun Anda dari konsol AWS manajemen atau menggunakan. AWS CLI Anda dapat membuat satu lingkungan untuk antrian, atau Anda dapat membuat beberapa lingkungan antrian yang diterapkan untuk membuat lingkungan eksekusi. Pendekatan ini memungkinkan Anda untuk membuat dan menguji lingkungan dalam langkah-langkah untuk membantu memastikan bahwa itu bekerja dengan benar untuk pekerjaan Anda.

Lingkungan Job dan step didefinisikan dalam template pekerjaan yang Anda gunakan untuk membuat pekerjaan dalam antrian Anda. Sintaks OpenJD sama dalam berbagai bentuk lingkungan ini. Di bagian ini kami akan menunjukkannya di dalam template pekerjaan.

Topik

- Mengatur variabel lingkungan dalam lingkungan antrian
- Mengatur jalur di lingkungan antrian
- Jalankan proses daemon latar belakang dari lingkungan antrian

Mengatur variabel lingkungan dalam lingkungan antrian

Banyak aplikasi dan kerangka kerja menggunakan variabel lingkungan untuk mengontrol pengaturan fitur, tingkat logging, dan konfigurasi tampilan. Anda dapat menggunakan lingkungan Open Job Description (OpenJD) untuk mengatur variabel lingkungan yang diwarisi oleh setiap perintah tugas dalam cakupannya.

Lingkup variabel lingkungan

AWS Deadline Cloud menerapkan variabel lingkungan dari lingkungan antrian yang Anda lampirkan ke antrian. Dalam template pekerjaan, Anda juga dapat menentukan variabel lingkungan pada tingkat pekerjaan dan langkah menggunakan lingkungan OpenJD. Variabel yang didefinisikan pada lingkup yang lebih sempit mengganti variabel dengan nama yang sama dari lingkup yang lebih luas.

- Lingkungan antrian — Template yang Anda lampirkan ke antrian di Deadline Cloud. Variabel berlaku untuk semua pekerjaan yang dikirimkan ke antrian. Anda dapat mengatur variabel dengan `variables` peta untuk nilai tetap, atau menggunakan skrip untuk nilai dinamis.
- Job environment — Didefinisikan `jobEnvironments` di bawah dalam template pekerjaan. Variabel berlaku untuk semua langkah dan tugas dalam pekerjaan. Variabel tingkat pekerjaan mengesampingkan variabel tingkat antrian dengan nama yang sama.
- Lingkungan langkah - Didefinisikan `stepEnvironments` di bawah dalam template pekerjaan. Variabel hanya berlaku untuk tugas-tugas dalam langkah itu. Variabel tingkat langkah mengesampingkan variabel tingkat pekerjaan atau tingkat antrian dengan nama yang sama.

Mengatur variabel dalam lingkungan antrian

Anda dapat mengatur variabel lingkungan dalam lingkungan antrian menggunakan `variables` peta untuk nilai tetap, atau menggunakan `script` dengan `onEnter` tindakan untuk nilai dinamis.

Template lingkungan antrian berikut menggunakan `variables` peta untuk mengatur `QT_QPA_PLATFORM` variabel`offscreen`, yang memungkinkan aplikasi yang menggunakan Kerangka Qt untuk berjalan pada host pekerja tanpa tampilan interaktif.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Untuk nilai dinamis, seperti memodifikasi PATH atau mengaktifkan lingkungan virtual, gunakan skrip yang mencetak garis dalam format `openjd_env: VAR=value` ke stdout. `openjd_env:`Awalan diperlukan. Menggunakan`echo`,`export`, atau mekanisme shell lainnya tanpa awalan tidak menyebarkan variabel ke pekerjaan dan tugas.

Template lingkungan antrian berikut menetapkan `QT_QPA_PLATFORM` variabel menggunakan 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"
```

Untuk melampirkan lingkungan antrian ke antrian, gunakan konsol Deadline Cloud atau. AWS CLI Untuk informasi selengkapnya, lihat Membuat lingkungan antrian di Panduan Pengguna Cloud AWS Batas Waktu. AWS CLI Perintah berikut menciptakan lingkungan antrian dari file template.

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

Untuk contoh yang lebih kompleks, seperti membuat dan mengaktifkan lingkungan virtual conda, lihat contoh lingkungan antrian Deadline Cloud. GitHub

Menetapkan variabel dalam template pekerjaan

Dalam template pekerjaan, tambahkan `variables` peta ke `stepEnvironments` entri `jobEnvironments` atau. Setiap entri adalah pasangan kunci-nilai di mana kuncinya adalah nama variabel dan nilainya adalah nilai variabel.

Template pekerjaan berikut menetapkan variabel `QT_QPA_PLATFORM` lingkungan ke`offscreen`, yang memungkinkan aplikasi yang menggunakan Kerangka Qt untuk berjalan pada host pekerja tanpa tampilan interaktif.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

Anda dapat mengatur beberapa variabel dalam satu definisi lingkungan.

```
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
```

Anda dapat mereferensikan parameter pekerjaan dalam nilai variabel dengan menggunakan `{{Param.ParameterName}}` sintaks.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Untuk mengganti variabel tingkat pekerjaan untuk langkah tertentu, tentukan `stepEnvironments` entri dengan nama variabel yang sama. Contoh berikut mendefinisikan `JOB_PROJECT_ID` pada tingkat pekerjaan dengan nilai`project-12`, dan kemudian mengganti nilai pada tingkat langkah dengan. `step-project-12` Tugas dalam langkah menggunakan nilai tingkat langkah.

```
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
```

Cobalah: Menjalankan sampel variabel lingkungan

Repositori sampel Deadline Cloud menyertakan bundel pekerjaan yang menunjukkan pengaturan dan melihat variabel lingkungan. Template pekerjaan sampel mendefinisikan variabel pada tingkat pekerjaan dan langkah, kemudian menjalankan tugas yang mencetak hasil gabungan. Gunakan prosedur berikut untuk menjalankan sampel dan memeriksa hasilnya.

Prasyarat

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di konsol Deadline Cloud untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI AWS dan Deadline Cloud di workstation Anda, ikuti langkah-langkah di Mengatur pengirim Deadline Cloud.
3. Gunakan `git` untuk mengkloning repositori sampel GitHub Deadline Cloud.

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

Menjalankan sampel

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_vars` sampel.

   ```
   deadline bundle submit job_env_vars
   ```

2. Di monitor Deadline Cloud, pilih pekerjaan baru untuk memantau kemajuannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia, pekerjaan selesai dalam beberapa detik. Pilih tugas, lalu pilih Lihat log di menu kanan atas panel tugas.

Membandingkan tindakan sesi dengan definisinya

Tampilan log menunjukkan tiga tindakan sesi. Buka file job_env_vars/template.yaml di editor teks untuk membandingkan setiap tindakan dengan definisinya di template pekerjaan.

1. Pilih tindakan Peluncuran JobEnv sesi. Output log menunjukkan variabel lingkungan tingkat pekerjaan yang disetel.

```
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
```

   Baris berikut dari template pekerjaan menentukan lingkungan ini.

```
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. Pilih tindakan Peluncuran StepEnv sesi. Output log menunjukkan variabel tingkat langkah, termasuk yang diganti`JOB_PROJECT_ID`.

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   Baris berikut dari template pekerjaan menentukan lingkungan ini.

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Pilih tindakan sesi Task run. Output log menunjukkan variabel lingkungan gabungan yang tersedia untuk tugas. Perhatikan bahwa `JOB_PROJECT_ID` menggunakan nilai `step-project-12` tingkat langkah.

```
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
```

Mengatur jalur di lingkungan antrian

Gunakan lingkungan OpenJD untuk memberikan perintah baru di lingkungan. Pertama Anda membuat direktori yang berisi file skrip, dan kemudian menambahkan direktori itu ke variabel `PATH` lingkungan sehingga executable dalam skrip Anda dapat menjalankannya tanpa perlu menentukan jalur direktori setiap kali. Daftar variabel dalam definisi lingkungan tidak menyediakan cara untuk memodifikasi variabel, jadi Anda melakukan ini dengan menjalankan skrip sebagai gantinya. Setelah skrip mengatur segalanya dan memodifikasi`PATH`, ia mengekspor variabel ke runtime OpenJD dengan perintah. `echo "openjd_env: PATH=$PATH"`

Prasyarat

Lakukan langkah-langkah berikut untuk menjalankan paket pekerjaan sampel dengan variabel lingkungan dari repositori github sampel Deadline Cloud.

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di konsol Deadline Cloud untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI dan Deadline Cloud di workstation Anda, ikuti langkah-langkah di Mengatur pengirim Deadline Cloud dari panduan pengguna.
3. Gunakan `git` untuk mengkloning repositori sampel GitHub Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Jalankan sampel jalur

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_with_new_command` sampel.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. Di monitor Deadline Cloud, Anda akan melihat pekerjaan baru dan dapat memantau kemajuannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia untuk menjalankan tugas pekerjaan, pekerjaan selesai dalam beberapa detik. Pilih tugas, lalu pilih opsi Lihat log di menu kanan atas panel tugas.

   Di sebelah kanan adalah dua tindakan sesi, Launch RandomSleepCommand dan Task run. Penampil log di tengah jendela sesuai dengan tindakan sesi yang dipilih di sebelah kanan.

Bandingkan tindakan sesi dengan definisinya

Di bagian ini Anda menggunakan monitor Deadline Cloud untuk membandingkan tindakan sesi dengan di mana mereka didefinisikan dalam template pekerjaan. Ini berlanjut dari bagian sebelumnya.

Buka file job_env_with_new_command/template.yaml di editor teks. Bandingkan tindakan sesi dengan tempat mereka didefinisikan dalam template pekerjaan.

1. Pilih tindakan Peluncuran RandomSleepCommand sesi di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```
 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.
```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```
 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. Pilih tindakan Peluncuran StepEnv sesi di monitor Deadline Cloud. Anda melihat output log sebagai berikut.

```
 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. Baris berikut dari template pekerjaan menentukan tindakan ini.

```
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
```

Jalankan proses daemon latar belakang dari lingkungan antrian

Dalam banyak kasus penggunaan rendering, memuat aplikasi dan data adegan dapat memakan waktu yang cukup lama. Jika pekerjaan memuat ulang mereka untuk setiap frame, itu akan menghabiskan sebagian besar waktunya untuk overhead. Seringkali mungkin untuk memuat aplikasi sekali sebagai proses daemon latar belakang, memintanya memuat data adegan, dan kemudian mengirimkannya perintah melalui komunikasi antar-proses (IPC) untuk melakukan render.

Banyak integrasi Deadline Cloud open source menggunakan pola ini. Proyek Open Job Description menyediakan pustaka runtime adaptor dengan pola IPC yang kuat pada semua sistem operasi yang didukung.

Untuk mendemonstrasikan pola ini, ada bundel pekerjaan sampel mandiri yang menggunakan Python dan kode bash untuk mengimplementasikan daemon latar belakang dan IPC untuk tugas-tugas untuk berkomunikasi dengannya. Daemon diimplementasikan dengan Python, dan

mendengarkan SIGUSR1 sinyal POSIX kapan harus memproses tugas. Detail tugas diteruskan ke daemon dalam file JSON tertentu, dan hasil menjalankan tugas dikembalikan sebagai file JSON lain.

Prasyarat

Lakukan langkah-langkah berikut untuk menjalankan sample job bundle dengan proses daemon dari repositori github sampel Deadline Cloud.

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di konsol Deadline Cloud untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI dan Deadline Cloud di workstation Anda, ikuti langkah-langkah di Mengatur pengirim Deadline Cloud dari panduan pengguna.
3. Gunakan `git` untuk mengkloning repositori sampel GitHub Deadline Cloud.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Jalankan sampel daemon

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_daemon_process` sampel.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. Dalam aplikasi monitor Deadline Cloud, Anda akan melihat pekerjaan baru dan dapat memantau perkembangannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia untuk menjalankan tugas pekerjaan, itu selesai dalam waktu sekitar satu menit. Dengan salah satu tugas yang dipilih, pilih opsi Lihat log di menu kanan atas panel tugas.

   Di sebelah kanan ada dua tindakan sesi, Launch DaemonProcess dan Task run. Penampil log di tengah jendela sesuai dengan tindakan sesi yang dipilih di sebelah kanan.

   Pilih opsi Lihat log untuk semua tugas. Garis waktu menunjukkan sisa tugas yang berjalan sebagai bagian dari sesi, dan `Shut down DaemonProcess` tindakan yang keluar dari lingkungan.

Lihat log daemon

1. Di bagian ini Anda menggunakan monitor Deadline Cloud untuk membandingkan tindakan sesi dengan di mana mereka didefinisikan dalam template pekerjaan. Ini berlanjut dari bagian sebelumnya.

   Buka file job_env_daemon_process/template.yaml di editor teks. Bandingkan tindakan sesi dengan tempat mereka didefinisikan dalam template pekerjaan.

2. Pilih tindakan `Launch DaemonProcess` sesi di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```
 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
```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```
  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. Pilih salah satu tindakan Task run: N session di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```
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 ----------------------------------------------
```

Baris berikut dari template pekerjaan adalah apa yang menentukan tindakan ini. ```langkah:

```
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
```

Menyediakan aplikasi untuk pekerjaan Anda

Anda dapat menggunakan lingkungan antrian untuk memuat aplikasi untuk memproses pekerjaan Anda. Saat membuat armada yang dikelola layanan menggunakan konsol Deadline Cloud, Anda memiliki opsi untuk membuat lingkungan antrian yang menggunakan manajer paket conda untuk memuat aplikasi.

Jika Anda ingin menggunakan manajer paket yang berbeda, Anda dapat membuat lingkungan antrian untuk manajer itu. Untuk contoh menggunakan Rez, lihatGunakan manajer paket yang berbeda.

Deadline Cloud menyediakan saluran conda untuk memuat pilihan aplikasi rendering ke lingkungan Anda. Mereka mendukung pengirim yang disediakan Deadline Cloud untuk aplikasi pembuatan konten digital.

Anda juga dapat memuat perangkat lunak untuk conda-forge untuk digunakan dalam pekerjaan Anda. Contoh berikut menunjukkan templat pekerjaan menggunakan lingkungan antrian yang disediakan oleh Deadline Cloud untuk memuat aplikasi sebelum menjalankan pekerjaan.

Topik

Mendapatkan aplikasi dari saluran conda

Anda dapat membuat lingkungan antrian khusus untuk pekerja Deadline Cloud Anda yang menginstal perangkat lunak pilihan Anda. Contoh lingkungan antrian ini memiliki perilaku yang sama dengan lingkungan yang digunakan oleh konsol untuk armada yang dikelola layanan. Ini menjalankan conda langsung untuk menciptakan lingkungan.

Lingkungan menciptakan lingkungan virtual conda baru untuk setiap sesi Deadline Cloud yang berjalan pada pekerja, dan kemudian menghapus lingkungan ketika selesai.

Conda menyimpan paket yang diunduh sehingga tidak perlu diunduh lagi, tetapi setiap sesi harus menautkan semua paket ke lingkungan.

Lingkungan mendefinisikan tiga skrip yang berjalan saat Deadline Cloud memulai sesi pada pekerja. Skrip pertama berjalan ketika `onEnter` tindakan dipanggil. Ini memanggil dua lainnya untuk mengatur variabel lingkungan. Ketika skrip selesai berjalan, lingkungan conda tersedia dengan semua variabel lingkungan yang ditentukan ditetapkan.

Untuk versi terbaru dari contoh, lihat conda_queue_env_console_equivalent.yaml di repositori pada. deadline-cloud-samples GitHub

Jika Anda ingin menggunakan aplikasi yang tidak tersedia di saluran conda, Anda dapat membuat saluran conda di Amazon S3 dan kemudian membuat paket Anda sendiri untuk aplikasi itu. Lihat Buat saluran conda menggunakan S3 untuk mempelajari selengkapnya.

Dapatkan pustaka open source dari conda-forge

Bagian ini menjelaskan cara menggunakan pustaka sumber terbuka dari `conda-forge` saluran. Contoh berikut adalah template pekerjaan yang menggunakan paket `polars` Python.

Pekerjaan menetapkan `CondaChannels` parameter `CondaPackages` dan yang ditentukan dalam lingkungan antrian yang memberi tahu Deadline Cloud tempat mendapatkan paket.

Bagian dari template pekerjaan yang menetapkan parameter adalah:

```
- 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
```

Untuk versi terbaru dari contoh template pekerjaan lengkap, lihat stage_1_self_contained_template/template.yaml. Untuk versi terbaru dari lingkungan antrian yang memuat paket conda, lihat conda_queue_env_console_equivalent.yaml di repositori pada. deadline-cloud-samples GitHub

Dapatkan Blender dari saluran deadline-cloud

Contoh berikut menunjukkan template pekerjaan yang didapat Blender dari saluran `deadline-cloud` conda. Saluran ini mendukung pengirim yang disediakan Deadline Cloud untuk perangkat lunak pembuatan konten digital, meskipun Anda dapat menggunakan saluran yang sama untuk memuat perangkat lunak untuk Anda gunakan sendiri.

Untuk daftar perangkat lunak yang disediakan oleh `deadline-cloud` channel, lihat Lingkungan antrian default di Panduan Pengguna Cloud AWS Batas Waktu.

Pekerjaan ini menetapkan `CondaPackages` parameter yang ditentukan dalam lingkungan antrian untuk memberi tahu Deadline Cloud agar dimuat Blender ke lingkungan.

Bagian dari template pekerjaan yang menetapkan parameter adalah:

```
- 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.
```

Untuk versi terbaru dari contoh template pekerjaan lengkap, lihat blender_render/template.yaml. Untuk versi terbaru dari lingkungan antrian yang memuat paket conda, lihat conda_queue_env_console_equivalent.yaml di repositori pada. deadline-cloud-samplesGitHub

Gunakan manajer paket yang berbeda

Manajer paket default untuk Deadline Cloud adalah conda. Jika Anda perlu menggunakan manajer paket yang berbeda, sepertiRez, Anda dapat membuat lingkungan antrian kustom yang berisi skrip yang menggunakan manajer paket Anda sebagai gantinya.

Contoh lingkungan antrian ini memberikan perilaku yang sama dengan lingkungan yang digunakan oleh konsol untuk armada yang dikelola layanan. Ini menggantikan manajer paket conda dengan. Rez

Lingkungan mendefinisikan tiga skrip yang berjalan saat Deadline Cloud memulai sesi pada pekerja. Skrip pertama berjalan ketika `onEnter` tindakan dipanggil. Ini memanggil dua lainnya untuk mengatur variabel lingkungan. Ketika skrip selesai berjalan, Rez lingkungan tersedia dengan semua variabel lingkungan yang ditentukan ditetapkan.

Contoh ini mengasumsikan bahwa Anda memiliki armada yang dikelola pelanggan yang menggunakan sistem file bersama untuk paket Rez.

Untuk versi terbaru dari contoh, lihat rez_queue_env.yaml di repositori pada. deadline-cloud-samplesGitHub

Buat saluran conda menggunakan S3

Jika pekerjaan Anda perlu menjalankan aplikasi yang tidak tersedia di `conda-forge`saluran `deadline-cloud`atau, Anda dapat meng-host saluran conda khusus untuk melayani paket Anda sendiri. Saat Anda membuat antrean di konsol AWS Deadline Cloud (Deadline Cloud), konsol akan menambahkan lingkungan antrean conda secara default. Untuk membuat paket Anda tersedia untuk pekerjaan, tambahkan saluran kustom ke lingkungan antrian.

Saluran conda adalah konten yang di-host statis yang dapat Anda host dengan berbagai cara, termasuk di sistem file atau di bucket Amazon Simple Storage Service (Amazon S3). Jika Deadline Cloud farm menggunakan sistem file bersama untuk aset, Anda dapat menggunakan jalur apa pun di dalamnya sebagai nama saluran. Anda dapat meng-host saluran di bucket Amazon S3 untuk akses yang lebih luas menggunakan izin AWS Identity and Access Management (IAM).

Anda dapat membuat dan menguji paket secara lokal, lalu mempublikasikannya ke saluran. Membangun paket secara lokal adalah cara mudah untuk mulai mengulangi resep pembuatan paket tanpa pengaturan infrastruktur. Anda juga dapat menggunakan antrean pembuatan paket Deadline Cloud untuk membuat paket dan mempublikasikannya ke saluran. Antrian pembuatan paket menyederhanakan pemeliharaan paket untuk beberapa sistem operasi dan konfigurasi akselerator. Anda dapat memperbarui versi dan mengirimkan set lengkap paket build dari mana saja.

Anda dapat mengonfigurasi saluran untuk studio dan Deadline Cloud farm Anda dengan berbagai cara. Anda dapat memiliki satu saluran Amazon S3 dan mengonfigurasi semua workstation dan host pertanian Anda untuk menggunakannya. Anda juga dapat memiliki lebih dari satu saluran dan mengatur mirroring dengan AWS DataSync (DataSync). Misalnya, antrean pembuatan paket Deadline Cloud dapat dipublikasikan ke saluran Amazon S3 yang dicerminkan di tempat untuk workstation dan host farm lokal.

Topik

Membangun dan menguji paket secara lokal

Sebelum memublikasikan paket ke Amazon S3 atau menyiapkan CI/CD otomatisasi di pertanian Deadline Cloud, Anda dapat membuat dan menguji paket conda di workstation menggunakan saluran sistem file lokal. Pendekatan ini memungkinkan Anda dengan cepat mengulangi resep secara lokal dan memverifikasi paket.

`rattler-build publish`Perintah membangun resep, menyalin paket yang dihasilkan ke saluran, dan mengindeks saluran dalam satu langkah. Saat Anda menargetkan direktori sistem file lokal, `rattler-build` membuat dan menginisialisasi saluran secara otomatis jika direktori tidak ada.

Petunjuk berikut menggunakan resep sampel Blender 4,5 dari repositori sampel Deadline Cloud pada. GitHub Anda dapat mengganti resep yang berbeda dari repositori sampel atau menggunakan resep Anda sendiri.

Prasyarat

Sebelum Anda mulai, instal alat-alat berikut di workstation Anda:

- pixi — Manajer paket yang Anda gunakan untuk menginstal `rattler-build` dan menguji paket. Instal pixi dari pixi.sh.
- rattler-build — Alat pembuatan paket yang digunakan oleh resep Conda Deadline Cloud. Setelah Anda menginstal pixi, jalankan perintah berikut untuk menginstal`rattler-build`.

```
pixi global install rattler-build
```

- git — Diperlukan untuk mengkloning repositori sampel. OnWindows, git for Windows juga menyediakan `bash` shell, yang diperlukan oleh beberapa resep Windows sampel.

Membangun dan menerbitkan paket ke saluran lokal

Dalam prosedur ini, Anda mengkloning repositori sampel Deadline Cloud dan menggunakannya `rattler-build publish` untuk membangun dan mempublikasikan paket ke saluran sistem file lokal.

> **Note**
>
> Aplikasi besar dapat memerlukan puluhan GB ruang disk kosong untuk arsip sumber, file yang diekstraksi, dan membangun output. Pastikan Anda menggunakan disk dengan ruang yang cukup untuk output build paket.

Untuk membuat dan memublikasikan paket ke saluran lokal

1. Kloning repositori sampel Deadline Cloud.

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

2. Ubah ke direktori `conda_recipes`.

```
cd deadline-cloud-samples/conda_recipes
```

3. Jalankan perintah berikut untuk membangun resep Blender 4.5 dan mempublikasikan paket ke direktori saluran lokal.

   LinuxAktif danmacOS, jalankan perintah berikut.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to file://$HOME/my-conda-channel \
       --build-number=+1
   ```

   Pada Windows (cmd), jalankan perintah berikut.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml ^
       --to file://%USERPROFILE%/my-conda-channel ^
       --build-number=+1
   ```

   `rattler-build publish`Perintah melakukan tindakan berikut:

   - Membangun paket dari resep.
   - Membuat direktori saluran jika direktori tidak ada.
   - Menyalin file paket ke saluran.
   - Mengindeks saluran sehingga manajer paket dapat menemukan paket.

   Jika resep paket Anda bergantung pada paket dari saluran tertentu, seperti conda-forge, tambahkan `-c conda-forge` ke perintah.

> **Tentang nomor build**
>
> `--build-number=+1`Opsi ini secara otomatis memilih nomor build berikutnya berdasarkan apa yang sudah ada di saluran tujuan. Praktik terbaik adalah tidak pernah menimpa paket di saluran. Selalu buat ke nomor build baru jika paket tersebut memiliki nama file yang sama. Menggunakan `--build-number=+1` mencapai ini ketika Anda membangun ke saluran produksi atau saluran pementasan yang mencerminkan produksi.

> Jika Anda ingin mengontrol nomor build secara langsung, Anda dapat mengaturnya dengan nilai tertentu seperti`--build-number=7`. Jika Anda menghilangkan opsi, `rattler-build` gunakan nomor build yang ditentukan dalam `recipe.yaml` file.

Untuk informasi selengkapnya`rattler-build publish`, lihat dokumentasi publikasi rattler-build.

Membangun debugging

Jika build gagal, `rattler-build` mempertahankan direktori build sehingga Anda dapat menyelidikinya. Jalankan perintah berikut untuk membuka shell interaktif di lingkungan build dengan semua variabel lingkungan diatur sebagaimana adanya selama pembuatan.

```
rattler-build debug shell
```

Dari shell debug, Anda dapat memodifikasi file, menjalankan perintah build individual, dan menambahkan dependensi untuk mengisolasi masalah. Untuk informasi selengkapnya, lihat Mendebug build di dokumentasi rattler-build.

Menguji paket

Setelah Anda membangun dan menerbitkan paket, buat proyek pixi sementara. Gunakan proyek untuk menginstal paket dari saluran lokal dan memverifikasi bahwa itu berfungsi dengan benar.

Untuk menguji paket

1. Buat direktori pengujian sementara dan inisialisasi proyek pixi dengan saluran lokal.

   LinuxAktif danmacOS, jalankan perintah berikut.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://$HOME/my-conda-channel
   ```

   Pada Windows (cmd), jalankan perintah berikut.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://%USERPROFILE%/my-conda-channel
   ```

2. Tambahkan paket ke proyek.

```
pixi add blender=4.5
```

3. Verifikasi bahwa paket berfungsi dengan benar.

```
pixi run blender --version
```

`pixi run`Perintah mengaktifkan lingkungan conda untuk direktori proyek dan menjalankan perintah yang ditentukan di dalamnya. Lingkungan tetap ada di direktori proyek, sehingga Anda dapat menggunakan `pixi run` perintah yang sama dari terminal lain.

Ketika Anda puas dengan paket tersebut, Anda dapat mempublikasikan paket ke saluran conda Amazon S3 sehingga pekerja Deadline Cloud dapat menginstal paket tersebut. Lihat Menerbitkan paket ke saluran conda S3.

Menghapus paket dari saluran

Hindari menghapus paket dari saluran yang Anda gunakan untuk produksi, karena lockfiles mereferensikan paket tertentu dengan hash. Menghapus paket mencegah pembuatan ulang lingkungan dari file kunci tersebut. Untuk saluran pengembangan dan pengujian, Anda dapat menghapus paket tertentu dengan menghapus `.conda` file dari direktori saluran dan kemudian mengindeks ulang saluran. Pertama, instal`rattler-index`.

```
pixi global install rattler-index
```

Kemudian hapus file paket dan indeks ulang saluran.

LinuxAktif danmacOS, jalankan perintah berikut.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

Pada Windows (cmd), jalankan perintah berikut.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

File Package disimpan dalam subdirektori khusus platform seperti,, atau. `linux-64` `win-64` `osx-arm64` Buat daftar isi subdirektori ini untuk menemukan nama file yang tepat dari paket yang ingin Anda hapus.

Membersihkan

Setelah pengujian, Anda dapat menghapus proyek pengujian dan saluran lokal.

Untuk membersihkan sumber daya pengujian

1. Hapus direktori proyek uji.

   LinuxAktif danmacOS, jalankan perintah berikut.

   ```
   rm -rf package-test-env
   ```

   Pada Windows (cmd), jalankan perintah berikut.

   ```
   rmdir /s /q package-test-env
   ```

2. Hapus direktori saluran conda lokal.

   LinuxAktif danmacOS, jalankan perintah berikut.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Pada Windows (cmd), jalankan perintah berikut.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Opsional) Hapus direktori `rattler-build` output yang berisi file paket yang dibangun.

   LinuxAktif danmacOS, jalankan perintah berikut.

   ```
   rm -rf deadline-cloud-samples/conda_recipes/output
   ```

   Pada Windows (cmd), jalankan perintah berikut.

   ```
   rmdir /s /q deadline-cloud-samples\conda_recipes\output
   ```

Publikasikan paket ke saluran conda Amazon S3

Anda dapat memublikasikan paket conda ke bucket Amazon Simple Storage Service (Amazon S3) Service S3) sehingga pekerja AWS Deadline Cloud (Deadline Cloud) dapat menginstalnya untuk menjalankan pekerjaan. `rattler-build publish`Perintah ini bekerja dengan Amazon S3 dengan cara yang sama seperti dengan saluran sistem file lokal. Perintah dapat membuat resep dan mempublikasikan hasilnya, atau mempublikasikan file paket yang sudah Anda buat. Dalam kedua kasus tersebut, perintah mengunggah paket ke bucket dan mengindeks saluran dalam satu langkah.

`rattler-build publish`Perintah mengautentikasi dengan AWS menggunakan rantai kredensi standar, sehingga menggunakan AWS konfigurasi Anda seperti alat apa pun AWS . Untuk informasi selengkapnya tentang mengonfigurasi kredensil, lihat Pengaturan file konfigurasi dan kredensi di AWS Command Line Interface ()AWS CLI Panduan Pengguna.

Prasyarat

Sebelum Anda mempublikasikan paket ke Amazon S3, lengkapi prasyarat berikut:

- pixi dan rattler-build - Instal pixi dari pixi.sh, lalu instal. `rattler-build`

  ```
  pixi global install rattler-build
  ```

- git — Diperlukan untuk mengkloning repositori sampel. OnWindows, git for Windows juga menyediakan `bash` shell, yang diperlukan oleh beberapa resep Windows sampel.
- Bucket Amazon S3 — Bucket Amazon S3 untuk digunakan sebagai saluran conda. Anda dapat menggunakan bucket lampiran pekerjaan dari Deadline Cloud farm atau membuat bucket terpisah.
- AWS credentials — Konfigurasikan kredensional di workstation Anda menggunakan perintah atau perintah. `aws configure` `aws login` Untuk informasi selengkapnya, lihat Menyiapkan AWS CLI di AWS Command Line Interface Panduan Pengguna.
- Izin IAM — (Opsional) Untuk mengurangi cakupan izin yang dimiliki kredensil, Anda dapat menggunakan kebijakan AWS Identity and Access Management (IAM) yang hanya memberikan izin berikut di bucket Amazon S3 dan awalan saluran yang Anda gunakan (misalnya,): `/Conda/*`
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:DeleteObject`
  - `s3:ListBucket`
  - `s3:GetBucketLocation`

Menerbitkan paket ke saluran Amazon S3

Gunakan `rattler-build publish` dengan `s3://` target untuk mempublikasikan paket ke saluran conda Amazon S3 Anda. Jika saluran tidak ada di ember, `rattler-build` inisialisasi saluran secara otomatis. Sebelum Anda mulai, pastikan Anda telah menyelesaikan prasyarat.

Contoh berikut menerbitkan resep sampel Blender 4.5 dari repositori sampel Deadline Cloud pada. GitHub Anda dapat mengganti resep yang berbeda dari repositori sampel atau menggunakan resep Anda sendiri.

> **Note**
>
> Aplikasi besar dapat memerlukan puluhan GB ruang disk kosong untuk arsip sumber, file yang diekstraksi, dan membangun output. Pastikan Anda menggunakan disk dengan ruang yang cukup untuk output build paket.

Untuk mempublikasikan paket ke saluran Amazon S3

1. Kloning repositori sampel Deadline Cloud.

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

2. Ubah ke direktori `conda_recipes`.

   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Jalankan perintah berikut. Ganti *`amzn-s3-demo-bucket`* dengan nama bucket Anda.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
   Conda/Default --build-number=+1
   ```

   `/Conda/Default`Awalan mengatur saluran di dalam ember. Anda dapat menggunakan awalan yang berbeda, tetapi awalan harus konsisten di semua perintah dan konfigurasi antrian yang mereferensikan saluran.

Tentang nomor build

`--build-number=+1`Opsi ini secara otomatis memilih nomor build berikutnya berdasarkan apa yang sudah ada di saluran tujuan. Praktik terbaik adalah tidak pernah menimpa paket di saluran. Selalu buat ke nomor build baru jika paket tersebut memiliki nama file yang sama. Menggunakan `--build-number=+1` mencapai ini ketika Anda membangun ke saluran produksi atau saluran pementasan yang mencerminkan produksi.
Jika Anda ingin mengontrol nomor build secara langsung, Anda dapat mengaturnya dengan nilai tertentu seperti`--build-number=7`. Jika Anda menghilangkan opsi, `rattler-build` gunakan nomor build yang ditentukan dalam `recipe.yaml` file.

Jika resep paket Anda bergantung pada paket dari saluran tertentu, seperti conda-forge, tambahkan `-c conda-forge` ke perintah.

Anda juga dapat memublikasikan file paket yang sudah Anda buat, misalnya, `.conda` file dari build lokal. Ganti *`amzn-s3-demo-bucket`* dengan nama bucket Anda.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Menginisialisasi atau mengindeks ulang saluran

Saat Anda menggunakan `rattler-build publish` untuk mempublikasikan paket, perintah menginisialisasi saluran secara otomatis jika saluran belum ada. Dalam kebanyakan kasus, Anda tidak perlu menginisialisasi atau mengindeks ulang saluran secara manual.

Anda mungkin perlu menginisialisasi atau mengindeks ulang saluran secara manual dalam situasi berikut:

- Anda ingin membuat saluran kosong sebelum memublikasikan paket apa pun, misalnya, untuk memverifikasi bahwa lingkungan antrian Deadline Cloud Anda dapat terhubung ke saluran.
- Anda mengunggah atau menghapus `.conda` file secara langsung dengan alat Amazon S3 alih-alih `rattler-build publish` menggunakan, dan indeks saluran sudah kedaluwarsa.

Menginisialisasi saluran kosong

Untuk menginisialisasi saluran kosong, buat `repodata.json` file dan unggah ke `noarch` subdirektori awalan saluran. Ganti *amzn-s3-demo-bucket* dengan nama bucket Anda.

```
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
```

`/Conda/Default`Awalan harus sesuai dengan awalan saluran yang digunakan lingkungan antrian Anda. Setelah menginisialisasi saluran, Anda dapat mempublikasikan paket ke saluran dengan menggunakan`rattler-build publish`.

Mengindeks ulang saluran

Jika indeks saluran kedaluwarsa, gunakan `rattler-index` untuk membangun kembali indeks dari file paket di saluran. Pertama, instal`rattler-index`.

```
pixi global install rattler-index
```

Kemudian indeks ulang saluran. Ganti *amzn-s3-demo-bucket* dengan nama bucket Anda.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

Menguji paket

Setelah Anda mempublikasikan paket, buat proyek pixi sementara untuk memverifikasi bahwa paket berfungsi dengan benar. Proyek ini menginstal paket dari saluran Amazon S3.

Untuk menguji paket

1. Buat direktori pengujian sementara dan inisialisasi proyek pixi dengan saluran Amazon S3. Ganti *amzn-s3-demo-bucket* dengan nama bucket Anda.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. Tambahkan paket ke proyek.

```
pixi add blender=4.5
```

3. Verifikasi bahwa paket berfungsi dengan benar.

```
pixi run blender --version
```

`pixi run`Perintah mengaktifkan lingkungan conda untuk direktori proyek dan menjalankan perintah yang ditentukan di dalamnya. Lingkungan tetap ada di direktori proyek, sehingga Anda dapat menggunakan `pixi run` perintah yang sama dari terminal lain.

Menghapus paket dari saluran

Hindari menghapus paket dari saluran yang Anda gunakan untuk produksi, karena lockfiles mereferensikan paket tertentu dengan hash. Menghapus paket mencegah pembuatan ulang lingkungan dari file kunci tersebut. Untuk saluran pengembangan dan pengujian, Anda dapat menghapus paket tertentu dengan menghapus `.conda` file dari bucket dan kemudian mengindeks ulang saluran.

Hapus file paket dan kemudian indeks ulang saluran. Ganti *`amzn-s3-demo-bucket`* dengan nama bucket Anda.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

Setelah Anda menghapus file, indeks ulang saluran untuk memperbarui metadata saluran. Untuk petunjuk, lihat Mengindeks ulang saluran.

File Package disimpan dalam subdirektori khusus platform seperti,, atau. `linux-64` `win-64` `osx-arm64` Untuk daftar paket dalam subdirektori, jalankan perintah berikut.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

Membersihkan

Setelah pengujian, hapus direktori proyek uji.

Untuk membersihkan sumber daya pengujian

- Hapus direktori proyek uji.

LinuxAktif danmacOS, jalankan perintah berikut.

```
rm -rf package-test-env
```

Pada Windows (cmd), jalankan perintah berikut.

```
rmdir /s /q package-test-env
```

Membangun debugging

Jika build gagal, `rattler-build` mempertahankan direktori build sehingga Anda dapat menyelidikinya. Jalankan perintah berikut untuk membuka shell interaktif di lingkungan build dengan semua variabel lingkungan diatur sebagaimana adanya selama pembuatan.

```
rattler-build debug shell
```

Dari shell debug, Anda dapat memodifikasi file, menjalankan perintah build individual, dan menambahkan dependensi untuk mengisolasi masalah. Untuk informasi selengkapnya, lihat Mendebug build di dokumentasi rattler-build.

Membangun paket untuk platform lain

`rattler-build publish`Perintah membangun paket untuk sistem operasi workstation tempat perintah berjalan. Jika armada Deadline Cloud Anda menggunakan sistem operasi yang berbeda dari workstation Anda, atau jika paket Anda memiliki persyaratan host lain, Anda memiliki opsi berikut:

- Jalankan `rattler-build publish` pada host yang cocok dengan sistem operasi target. Misalnya, gunakan instans Amazon Elastic Compute Cloud (Amazon EC2) yang Linux berjalan untuk membuat paket untuk armada. Linux
- Gunakan antrian pembuatan paket Deadline Cloud untuk mengotomatiskan build di platform target. Lihat Membuat antrian pembuatan paket.
- (Advanced) Gunakan kompilasi silang untuk membangun paket untuk platform yang berbeda dari workstation Anda. Untuk informasi selengkapnya, lihat Cross-compilation dalam dokumentasi rattler-build.

Langkah selanjutnya

Setelah memublikasikan paket ke saluran conda Amazon S3, konfigurasikan antrian Deadline Cloud Anda untuk menggunakan saluran:

- Konfigurasikan izin antrian produksi untuk paket conda kustom — Berikan akses hanya-baca antrian produksi Anda ke saluran conda Amazon S3.
- Menambahkan saluran conda ke lingkungan antrian — Konfigurasikan lingkungan antrian untuk menginstal paket dari saluran conda Amazon S3.

Konfigurasikan izin antrian produksi untuk paket conda kustom

Antrian produksi Anda memerlukan izin hanya-baca ke `/Conda` awalan di bucket S3 antrian. Buka halaman AWS Identity and Access Management (IAM) untuk peran yang terkait dengan antrian produksi dan ubah kebijakan dengan yang berikut:

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrean pembuatan paket.
2. Pilih peran layanan antrian, lalu pilih Edit antrian.
3. Gulir ke bagian Peran layanan antrian, lalu pilih Lihat peran ini di konsol IAM.
4. Dari daftar kebijakan izin, pilih antrian AmazonDeadlineCloudQueuePolicyuntuk Anda.
5. Dari tab Izin, pilih Edit.
6. Tambahkan bagian baru ke peran layanan antrian seperti berikut ini. Ganti *amzn-s3-demo-bucket* dan *111122223333* dengan ember dan akun Anda sendiri.

```
{
   "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"
    }
   }
  },
```

Menambahkan saluran conda ke lingkungan antrian

Untuk menggunakan saluran conda S3, Anda perlu menambahkan lokasi `s3://amzn-s3-demo-bucket/Conda/Default` saluran ke `CondaChannels` parameter pekerjaan yang Anda kirimkan ke Deadline Cloud. Pengirim yang dilengkapi dengan Deadline Cloud menyediakan bidang untuk menentukan saluran dan paket conda kustom.

Anda dapat menghindari memodifikasi setiap pekerjaan dengan mengedit lingkungan antrian conda untuk antrian produksi Anda. Gunakan prosedur berikut:

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrean produksi.
2. Pilih tab lingkungan.
3. Pilih lingkungan antrian Conda, lalu pilih Edit.
4. Pilih editor JSON, dan kemudian dalam skrip, temukan definisi parameter untuk`CondaChannels`.
5. Edit baris `default: "deadline-cloud"` sehingga dimulai dengan saluran conda S3 yang baru dibuat:

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Armada yang dikelola layanan memungkinkan prioritas saluran fleksibel untuk conda secara default. Untuk pekerjaan yang meminta `blender=4.5` apakah Blender 4.5 ada di saluran baru dan `deadline-cloud` saluran, paket akan ditarik dari saluran mana pun yang pertama dalam daftar saluran. Jika versi paket tertentu tidak ditemukan di saluran pertama maka saluran berikutnya akan diperiksa untuk versi paket.

Untuk armada yang dikelola pelanggan, Anda dapat mengaktifkan penggunaan paket conda dengan menggunakan salah satu contoh lingkungan antrian conda di repositori sampel Deadline Cloud. GitHub

Buat paket conda untuk aplikasi atau plugin

Paket conda adalah arsip perangkat lunak terkompresi yang ditulis dalam bahasa apa pun. Conda mendukung berbagai kombinasi sistem operasi dan arsitektur, sehingga Anda dapat mengemas aplikasi lengkap sepertiBlender,Maya, dan Nuke bersama pustaka untuk Python dan bahasa lainnya. Untuk informasi selengkapnya tentang paket conda, lihat Paket dalam dokumentasi conda.

Untuk menggunakan paket conda, Anda menginstalnya ke lingkungan virtual. Lingkungan virtual conda memiliki direktori awalan tempat paket diinstal. Menginstal paket menggunakan hardlinking atau reflinking file saat didukung, sehingga membuat beberapa lingkungan dengan paket yang sama tidak menggunakan ruang disk tambahan yang signifikan. Untuk menggunakan lingkungan virtual, Anda mengaktifkannya untuk mengatur variabel lingkungan. Aktivasi menjalankan skrip yang disediakan paket, memberikan setiap paket kesempatan untuk memodifikasi PATH atau variabel lingkungan lainnya. Paket Conda biasanya berisi aplikasi atau pustaka, tetapi aktivasi fleksibel berarti mereka juga dapat menunjuk ke aplikasi yang diinstal pada sistem file bersama.

Membuat paket khusus melibatkan tiga tahap: resep berisi instruksi pembuatan, paket adalah artefak (`.conda`atau `.tar.bz2` file) yang dibangun, dan saluran menghosting paket untuk instalasi. `rattler-build publish`Perintah menangani ketiga langkah—dapat membangun resep ke dalam paket dan mempublikasikannya ke saluran, atau dapat mengambil artefak paket secara langsung untuk menerbitkannya.

Komunitas conda-forge memelihara resep paket untuk serangkaian perangkat lunak open source yang luas, dan menghosting artefak paket di saluran. `conda-forge` Anda dapat mengonfigurasi antrian untuk disertakan `conda-forge` sebagai sumber paket, lalu membangun paket khusus yang bergantung pada paket conda-forge untuk dijalankan. UntukLinux, conda-forge menghosting rantai alat kompiler lengkap termasuk dukungan CUDA, dengan opsi kompilasi dan penautan yang konsisten dipilih. Anda dapat menggunakan paket conda-forge sebagai dependensi dalam resep Anda sendiri, atau menginstalnya bersama paket kustom Anda di lingkungan yang sama.

Anda dapat menggabungkan seluruh aplikasi, termasuk dependensi, ke dalam paket conda. Paket Deadline Cloud menyediakan di saluran deadline-cloud untuk armada yang dikelola layanan menggunakan pendekatan pengemasan ulang biner ini. Ini mengatur file yang sama sebagai instalasi agar sesuai dengan lingkungan virtual conda.

> **Note**
>
> Aplikasi besar dapat memerlukan puluhan GB ruang disk kosong untuk arsip sumber, file yang diekstraksi, dan membangun output. Pastikan Anda menggunakan disk dengan ruang yang cukup untuk output build paket.

Package sebuah aplikasi

Saat mengemas ulang aplikasi untuk conda, ada dua tujuan:

- Sebagian besar file untuk aplikasi harus terpisah dari struktur lingkungan virtual conda utama. Lingkungan kemudian dapat mencampur aplikasi dengan paket dari sumber lain seperti conda-forge.
- Ketika lingkungan virtual conda diaktifkan, aplikasi harus tersedia dari variabel lingkungan PATH.

Untuk mengemas ulang aplikasi untuk conda

1. Tulis resep conda build yang menginstal aplikasi ke dalam subdirektori seperti. `$CONDA_PREFIX/opt/<application-name>` Ini memisahkannya dari direktori awalan standar seperti dan. `bin` `lib`
2. Tambahkan symlink atau luncurkan skrip `$CONDA_PREFIX/bin` untuk menjalankan binari aplikasi.

   Atau, buat skrip activate.d yang akan dijalankan `conda activate` perintah untuk menambahkan direktori biner aplikasi ke PATH. AktifWindows, di mana symlink tidak didukung di mana pun lingkungan dapat dibuat, gunakan peluncuran aplikasi atau aktifkan skrip.d sebagai gantinya.
3. Beberapa aplikasi bergantung pada pustaka yang tidak diinstal secara default pada armada yang dikelola layanan Deadline Cloud. Misalnya, sistem jendela X11 biasanya tidak diperlukan untuk pekerjaan non-interaktif, tetapi beberapa aplikasi masih mengharuskannya untuk berjalan tanpa antarmuka grafis. Anda harus memberikan dependensi tersebut dalam paket yang Anda buat.
4. Jika aplikasi mendukung plugin, berikan konvensi yang jelas bahwa paket plugin harus mengikuti untuk mengintegrasikan dengan aplikasi dalam lingkungan virtual. Misalnya, resep sampel Maya 2026 mendokumentasikan konvensi ini untuk Maya plugin.

5. Pastikan Anda mengikuti perjanjian hak cipta dan lisensi untuk aplikasi yang Anda paket. Sebaiknya gunakan bucket Amazon S3 pribadi untuk saluran conda Anda guna mengontrol distribusi dan membatasi akses paket ke peternakan Anda.

Contoh resep untuk paket di `deadline-cloud` saluran tersedia di repositori sampel Deadline Cloud pada. GitHub

Package sebuah plugin

Plugin aplikasi dapat dikemas sebagai paket conda mereka sendiri. Saat membuat paket plugin, ikuti panduan ini:

- Sertakan paket aplikasi host sebagai dependensi build dan run dalam resep `recipe.yaml` build. Gunakan batasan versi sehingga resep build hanya diinstal dengan paket yang kompatibel.
- Ikuti konvensi paket aplikasi host untuk mendaftarkan plugin.

Paket adaptor

Beberapa integrasi aplikasi Deadline Cloud menggunakan adaptor yang memperluas antarmuka aplikasi untuk menyederhanakan penulisan templat pekerjaan. Adaptor adalah antarmuka baris perintah dengan dukungan untuk menjalankan daemon latar belakang, status pelaporan, dan menerapkan pemetaan jalur. Untuk informasi selengkapnya, lihat Open Job Description Adaptor Runtime aktif. GitHub Misalnya, deadline-cloud-for-mayapada GitHub menyertakan GUI pengiriman pekerjaan terintegrasi dan Maya adaptor yang tersedia sebagai `maya-openjd` paket pada armada yang dikelola layanan.

Pengajuan Job dari Deadline Cloud submitter GUIs menyertakan nilai `CondaPackages` parameter yang menentukan paket conda untuk disertakan dalam lingkungan virtual untuk menjalankan pekerjaan. Nilai `CondaPackages` parameter untuk Maya biasanya terlihat seperti `maya=2026.* maya-openjd=0.15.* maya-mtoa` dan mungkin berisi entri alternatif untuk paket plugin. Ketika lingkungan antrian menyiapkan lingkungan virtual conda untuk menjalankan pekerjaan, itu menyelesaikan nama paket dan batasan versi ini agar kompatibel dan menambahkan semua paket ketergantungan yang perlu mereka jalankan. Setiap paket adaptor dan plugin menentukan apa yang kompatibel dengannya, termasuk versi manaMaya, versi Python, dan dependensi lainnya.

Untuk membuat paket adaptor Anda sendiri menggunakan sampel kami seperti resep maya-openjdGitHub, Anda dapat membangun paket untuk Python dan dependensi lain yang disediakan

oleh conda-forge. Anda mungkin perlu membuat tenggat waktu dan openjd-adaptor-runtimeresep terlebih dahulu untuk memenuhi dependensi.

Buat resep conda build untuk Blender

Blendergratis untuk digunakan dan mudah dikemas dengan conda, yang menjadikannya titik awal yang baik untuk mempelajari cara membuat paket conda untuk AWS Deadline Cloud (Deadline Cloud). BlenderYayasan menyediakan arsip aplikasi untuk beberapa sistem operasi. Resep sampel Blender 4,5 di repositori sampel Deadline Cloud pada GitHub paket arsip ini ke dalam paket conda.

Memahami resepnya

File recipe.yaml mendefinisikan metadata paket, sumber, dan opsi build dalam sintaks template URLs rattler-build. Resep menentukan nomor versi sekali dan menyediakan sumber yang berbeda URLs berdasarkan sistem operasi.

`build`Bagian dalam `recipe.yaml` mematikan relokasi biner dan pemeriksaan penautan objek bersama dinamis (DSO). Opsi ini mengontrol cara kerja paket saat diinstal ke lingkungan virtual conda di awalan direktori apa pun. Nilai default yang digunakan di `build` bagian ini dirancang untuk mengemas setiap perpustakaan ketergantungan secara terpisah, tetapi ketika mengemas ulang aplikasi biner, Anda perlu mengubahnya. Blendertidak memerlukan penyesuaian RPATH karena arsip aplikasi dibangun dengan mempertimbangkan relokasi. Lihat Membuat resep conda untuk Maya untuk contoh menambahkan relokasi.

Selama pembuatan paket, skrip build.sh atau build_win.sh berjalan untuk menginstal file ke lingkungan. Skrip ini menyalin file instalasi ke dalam`$PREFIX/opt/blender`, membuat symlink dari `$PREFIX/bin` (onLinux), dan mengatur skrip aktivasi yang mengkonfigurasi variabel lingkungan seperti. `BLENDER_LOCATION` AktifWindows, skrip aktivasi menambahkan Blender direktori ke PATH alih-alih membuat symlink.

Skrip Windows build menggunakan `bash` alih-alih `cmd.exe` file.bat untuk konsistensi di seluruh platform. Anda dapat menginstal git Windows`bash` untuk menyediakan pembuatan paket.

Resep ini juga menyertakan `deadline-cloud.yaml` file yang menentukan platform conda dan metadata untuk mengirimkan pekerjaan pembuatan paket otomatis ke Deadline Cloud. Untuk informasi selengkapnya, lihat Mengirimkan pekerjaan pembuatan paket.

Membangun Blender paket

Gunakan `rattler-build publish` untuk membuat resep Blender 4.5 dan mempublikasikan paket ke saluran. Anda dapat memublikasikan ke saluran sistem file lokal untuk pengujian atau langsung ke saluran Amazon S3 untuk penggunaan produksi. Jika Anda menyelesaikan penyiapan di Build dan test packages secara lokal, jalankan perintah berikut dari `conda_recipes` direktori.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Untuk opsi penerbitan lainnya:

- Untuk memublikasikan ke saluran Amazon S3, lihat Menerbitkan paket ke saluran conda S3.
- Untuk mengotomatiskan build menggunakan antrean pembuatan paket Deadline Cloud, lihat Mengotomatiskan build paket dengan Deadline Cloud.

Uji paket Anda dengan pekerjaan Blender render

Setelah Anda membangun paket Blender 4.5, Anda dapat mengujinya dengan pekerjaan render. Jika Anda tidak memiliki Blender adegan, unduh adegan Blender 3.5 - Cozy Kitchen dari halaman file Blender demo. Repositori sampel Deadline Cloud berisi paket `blender_render` pekerjaan dan lingkungan antrian conda yang dapat Anda gunakan untuk pengujian lokal dan cloud.

Menguji secara lokal

Anda dapat menjalankan template pekerjaan di workstation Anda menggunakan Open Job Description CLI. Instal CLI dengan. `pip`

```
pip install openjd-cli
```

Dari `job_bundles` direktori di repositori sampel, jalankan perintah berikut. Ganti */path/to/scene.blend* dengan jalur ke file Blender adegan Anda.

```
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
```

`--environment`Opsi ini menerapkan lingkungan antrian conda, yang menciptakan lingkungan virtual conda dengan paket yang ditentukan dalam. `CondaPackages` `CondaChannels`Parameter memberitahu lingkungan antrian di mana menemukan paket. Jika Anda memublikasikan ke saluran Amazon S3 alih-alih saluran lokal, ganti `file://` jalur dengan URL `s3://` saluran Anda.

Pengujian pada Deadline Cloud

Setelah mengonfigurasi antrian produksi untuk menggunakan saluran conda Amazon S3, Anda dapat mengirimkan pekerjaan render ke Deadline Cloud. Dari `job_bundles` direktori di repositori sampel, jalankan perintah berikut.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Gunakan monitor Deadline Cloud untuk melacak kemajuan pekerjaan. Di monitor, pilih tugas untuk pekerjaan itu dan pilih Lihat log. Pilih tindakan sesi Launch Conda untuk memverifikasi bahwa paket ditemukan di saluran Amazon S3.

Buat resep conda build untuk Autodesk Maya

Aplikasi komersial seperti Autodesk Maya memperkenalkan persyaratan pengemasan tambahan dibandingkan dengan aplikasi open source sepertiBlender. BlenderResep ini mengemas arsip sederhana yang dapat dipindahkan di bawah lisensi open source. Aplikasi komersial sering didistribusikan melalui installer dan memerlukan konfigurasi manajemen lisensi.

Pertimbangan untuk aplikasi komersial

Pertimbangan berikut berlaku saat mengemas aplikasi komersial. Detailnya menggambarkan bagaimana masing-masing berlaku. Maya

- Perizinan — Memahami hak lisensi dan pembatasan aplikasi. Anda mungkin perlu mengonfigurasi sistem manajemen lisensi. Baca FAQ Manfaat Autodesk Berlangganan tentang Hak Cloud untuk memahami hak cloud. Maya Autodeskproduk bergantung pada `ProductInformation.pit` file yang biasanya memerlukan akses administrator untuk mengkonfigurasi. Fitur produk untuk klien tipis memberikan alternatif yang dapat direlokasi. Lihat Lisensi Klien Tipis untuk Maya dan MotionBuilder untuk informasi lebih lanjut.

- Dependensi pustaka sistem — Beberapa aplikasi bergantung pada pustaka yang tidak diinstal pada host pekerja armada yang dikelola layanan. Mayatergantung pada perpustakaan termasuk freetype dan fontconfig. Ketika pustaka ini tersedia di manajer paket sistem, seperti `dnf` untuk AL2023, Anda dapat menggunakan manajer paket sebagai sumber. Karena paket RPM tidak dibangun untuk dapat dipindahkan, Anda perlu menggunakan alat seperti `patchelf` untuk menyelesaikan dependensi dalam awalan instalasi. Maya
- Akses administrator untuk instalasi — Beberapa installer memerlukan akses administrator. Armada yang dikelola layanan tidak menyediakan akses administrator, jadi Anda perlu menginstal aplikasi pada sistem terpisah dan membuat arsip file untuk pembuatan paket. WindowsPemasang untuk Maya membutuhkan pendekatan ini. README.md dalam resep mendokumentasikan prosedur berulang menggunakan instans Amazon Elastic Compute Cloud (Amazon EC2) yang baru diluncurkan.
- Integrasi plugin — Maya Paket sampel mendefinisikan `MAYA_NO_HOME=1` untuk mengisolasi aplikasi dari konfigurasi tingkat pengguna, dan menambahkan jalur pencarian modul `MAYA_MODULE_PATH` sehingga paket plugin dapat menempatkan `.mod` file dalam lingkungan virtual. Lihat resep sampel Maya 2026 untuk konvensi integrasi plugin lengkap.

Memahami resepnya

File recipe.yaml mendefinisikan metadata paket dalam sintaks template rattler-build. Tinjau bagian file berikut:

- sumber — Referensi arsip penginstal, termasuk hash sha256. PadaLinux, sumbernya adalah arsip Autodesk penginstal. PadaWindows, sumber mencakup arsip penginstal dan `cleanMayaForCloud.py` skrip dari Autodesk yang Maya mempersiapkan penyebaran cloud. Perbarui hash saat Anda mengubah file sumber, misalnya saat mengemas versi baru.
- build — Mematikan relokasi biner default dan pemeriksaan penautan DSO karena mekanisme otomatis tidak berfungsi dengan benar untuk pustaka dan direktori biner yang digunakan. Maya PadaLinux, resep termasuk `patchelf` sebagai dependensi build untuk menyetel relatif RPATHs secara manual.
- tentang — Metadata tentang aplikasi untuk menjelajah atau memproses konten saluran conda.

Skrip build (build.sh forLinux, build_win.sh forWindows) menyertakan komentar yang menjelaskan setiap langkah. Skrip melakukan tugas-tugas utama berikut:

- Ekstrak installer - Ekstrak file Maya instalasi ke awalan conda. WindowsSkrip Linux dan menangani ini secara berbeda karena format penginstal. Lihat skrip build untuk detailnya.
- Instal dependensi pustaka sistem — Linux Aktif, skrip mengunduh dan mengekstrak pustaka sistem yang Maya membutuhkan tetapi tidak ada pada host armada yang dikelola layanan. Skrip menyalin pustaka ini ke dalam Maya `lib` direktori sehingga tersedia dalam lingkungan conda.
- Setel relatif RPATHs dengan patchelf — AktifLinux, skrip digunakan `patchelf --add-rpath` untuk menambahkan jalur `$ORIGIN` -relatif ke pustaka bersama. Pendekatan ini mengikuti rekomendasi conda untuk tidak pernah digunakan `LD_LIBRARY_PATH` di lingkungan conda. Skrip menambal pustaka pada beberapa tingkat direktori (`liblib/python*/site-packages`,,`lib/python*/lib-dynload`) sehingga setiap perpustakaan dapat menemukan dependensinya relatif terhadap lokasinya sendiri. Resepnya mengikuti praktik pengaturan terbaik `DT_RUNPATH` alih-alih`DT_RPATH`, yang memungkinkan `LD_LIBRARY_PATH` untuk mengganti jalur pencarian saat diperlukan untuk debugging.
- Konfigurasi lisensi klien tipis - Skrip mengatur lisensi klien tipis seperti yang didokumentasikan oleh Autodesk sehingga `ProductInformation.pit` file dapat ditemukan di dalam lingkungan conda daripada memerlukan akses administrator tingkat sistem.
- Siapkan skrip aktivasi — Skrip membuat mengaktifkan dan menonaktifkan skrip yang mengatur variabel lingkungan termasuk`MAYA_LOCATION`,,, `MAYA_VERSION` dan. `MAYA_NO_HOME` `MAYA_MODULE_PATH` WindowsAktif, skrip menghasilkan keduanya `.sh` dan file `.bat` aktivasi karena lingkungan antrian sampel Deadline Cloud digunakan `bash` untuk mengaktifkan lingkungan aktif. Windows

Membangun Maya paket

Sebelum Anda membuat Maya paket, unduh Maya penginstal dari Autodesk akun Anda. UntukLinux, tempatkan arsip langsung ke `conda_recipes/archive_files` direktori. UntukWindows, ikuti prosedur di README.md untuk membuat arsip.

Gunakan `rattler-build publish` untuk membangun dan mempublikasikan paket. MayaResepnya membutuhkan `patchelf` ketergantungan buildLinux, yang tersedia dari conda-forge. Tambahkan `-c conda-forge` untuk membuat dependensi tersedia selama pembuatan. Dari `conda_recipes` direktori, jalankan perintah berikut.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
```

```
    -c conda-forge
```

Untuk opsi penerbitan lainnya:

- Untuk memublikasikan ke saluran Amazon S3, lihat Menerbitkan paket ke saluran conda S3.
- Untuk mengotomatiskan build menggunakan antrean pembuatan paket Deadline Cloud, lihat Mengotomatiskan build paket dengan Deadline Cloud. Untuk membangun keduanya Linux dan Windows paket, gunakan `--all-platforms` opsi dengan `submit-package-job` skrip.

Untuk merender sampel meja putar dengan Maya danArnold, buat paket MtoAplugin dan Mayaadaptor. Setelah memublikasikan ketiga paket, Anda dapat mengirimkan pekerjaan render pengujian menggunakan turntable withMaya/Arnoldjob bundle dari repositori sampel Deadline Cloud. Lihat Uji paket Anda dengan pekerjaan render Maya.

Buat resep conda build untuk adaptor Maya

`maya-openjd`Paket ini menyediakan adaptor yang terintegrasi Maya dengan pengiriman pekerjaan AWS Deadline Cloud (Deadline Cloud). Saat Anda mengirimkan pekerjaan Maya render menggunakan GUI submitter Deadline Cloud, `CondaPackages` parameter tersebut disertakan `maya-openjd` bersama paket. `maya` Adaptor menangani peluncuranMaya, mengkomunikasikan parameter render, dan mengelola siklus hidup aplikasi selama sesi pekerjaan. Untuk informasi selengkapnya tentang adaptor, lihat Paket adaptor.

Memahami resepnya

Resep sampel maya-openjd membangun adaptor dari deadline-cloud-for-mayapaket sumber yang diterbitkan ke PyPI. Recipe.yaml menginstal paket menggunakan `pip` ke dalam lingkungan conda.

Resepnya bergantung pada Python dan dua paket lain dari repositori sampel Deadline Cloud yang perlu Anda buat terlebih dahulu:

- deadline — Pustaka klien Deadline Cloud.
- openjd-adaptor-runtime— Runtime adaptor Open Job Description.

Python dan dependensi lainnya tersedia dari conda-forge, jadi tambahkan `-c conda-forge` ke `rattler-build publish` perintah saat Anda membuat paket adaptor.

Membangun paket adaptor

`maya-openjd`Paket ini bergantung pada dua paket lain dari repositori sampel Deadline Cloud. Bangun ketiga paket secara berurutan dari `conda_recipes` direktori. `-c conda-forge`Opsi pada setiap perintah adalah untuk memenuhi dependensi resep untuk pustaka Python dan Python.

Bangun `deadline` paketnya.

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Bangun `openjd-adaptor-runtime` paketnya.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Bangun `maya-openjd` paketnya.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

Untuk opsi penerbitan lainnya:

- Untuk memublikasikan ke saluran Amazon S3, lihat Menerbitkan paket ke saluran conda S3.
- Untuk mengotomatiskan build menggunakan antrean pembuatan paket Deadline Cloud, lihat Mengotomatiskan build paket dengan Deadline Cloud.

Buat resep build conda untuk plugin Autodesk Maya to Arnold (MtoA)

Maya to Arnold (MtoA)Plugin menambahkan Arnold renderer sebagai opsi di dalamnya. Maya Resep sampel MToA menunjukkan cara mengemas plugin sebagai paket conda terpisah yang terintegrasi dengan paket aplikasi host.

Memahami resepnya

Recipe.yaml menentukan dependensi pada `maya` paket untuk persyaratan build dan run. Ketergantungan ini menggunakan kendala versi sehingga plugin hanya diinstal dengan versi yang kompatibel. Maya

Resepnya menggunakan arsip sumber yang sama dengan Maya resepnya. Skrip build menginstal MtoA dan membuat `mtoa.mod` file di `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` direktori tempat Maya paket mengkonfigurasi. `MAYA_MODULE_PATH` Arnolddan Maya menggunakan teknologi perizinan yang sama, sehingga Maya paket tersebut sudah menyertakan informasi perizinan yang Arnold dibutuhkan.

Membangun MtoA paket

Bangun Maya paket sebelum Anda membuat MtoA paket, karena MtoA tergantung Maya pada waktu pembuatan. Gunakan `rattler-build publish` untuk membangun dan mempublikasikan paket. Dari `conda_recipes` direktori, jalankan perintah berikut.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

`rattler-build publish`Perintah menggunakan saluran target sebagai saluran prioritas tertinggi saat menyelesaikan dependensi, sehingga `maya` paket yang Anda terbitkan sebelumnya tersedia secara otomatis.

Untuk opsi penerbitan lainnya:

- Untuk memublikasikan ke saluran Amazon S3, lihat Menerbitkan paket ke saluran conda S3.
- Untuk mengotomatiskan build menggunakan antrean pembuatan paket Deadline Cloud, lihat Mengotomatiskan build paket dengan Deadline Cloud.

Uji paket Anda dengan pekerjaan Maya render

Setelah Anda membangunMaya,MtoA, dan `maya-openjd` paket, Anda dapat mengujinya dengan pekerjaan render. Repositori sampel Deadline Cloud berisi turntable denganMaya/Arnoldjob bundle yang membuat animasi menggunakan dan. Maya Arnold Paket pekerjaan juga digunakan FFmpeg untuk menyandikan video, yang tersedia dari `conda-forge` saluran.

Menguji secara lokal

Anda dapat menjalankan template pekerjaan di workstation Anda menggunakan Open Job Description CLI. Instal CLI dengan. `pip`

```
pip install openjd-cli
```

Dari `job_bundles` direktori di repositori sampel, jalankan perintah berikut. `ErrorOnArnoldLicenseFail=false`Parameter memberitahu Arnold untuk merender dengan tanda air alih-alih gagal ketika tidak ada lisensi yang tersedia.

```
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
```

`--environment`Opsi ini menerapkan lingkungan antrian conda, yang menciptakan lingkungan virtual conda dengan paket yang ditentukan dalam. `CondaPackages` `CondaChannels`Parameter ini mencakup saluran lokal untuk paket kustom Anda dan `conda-forge` untuk`ffmpeg`. Jika Anda memublikasikan ke saluran Amazon S3 alih-alih saluran lokal, ganti `file://` jalur dengan URL `s3://` saluran Anda.

Ketika pekerjaan selesai, output yang diberikan ada di direktori. `turntable_with_maya_arnold/output/`

Pengujian pada Deadline Cloud

Setelah mengonfigurasi antrian produksi untuk menggunakan saluran conda Amazon S3, kirimkan pekerjaan render ke Deadline Cloud. Tambahkan `conda-forge` saluran ke `CondaChannels` parameter di lingkungan antrian conda Anda untuk menyediakan sumber `ffmpeg` dan dependensi Python yang dibutuhkan adaptor. Dari `job_bundles` direktori di repositori sampel, jalankan perintah berikut.

```
deadline bundle submit turntable_with_maya_arnold
```

Gunakan monitor Deadline Cloud untuk melacak kemajuan pekerjaan. Di monitor, pilih tugas untuk pekerjaan itu dan pilih Lihat log. Pilih tindakan sesi Launch Conda untuk memverifikasi bahwa`maya`,`maya-mtoa`, dan `maya-openjd` paket ditemukan di saluran Amazon S3.

Otomatiskan pembuatan paket dengan Deadline Cloud

Untuk CI/CD alur kerja atau saat Anda perlu membuat paket untuk beberapa sistem operasi, Anda dapat membuat antrean pembuatan paket Deadline Cloud. Jadwal antrian membangun pekerjaan di armada Anda, yang membuat paket dan mempublikasikannya ke saluran conda Amazon Simple Storage Service (Amazon S3). Ini menyederhanakan pemeliharaan pembuatan paket berkelanjutan untuk rilis perangkat lunak di semua konfigurasi yang Anda butuhkan.

Anda dapat membuat antrean pembuatan paket menggunakan templat AWS CloudFormation (CloudFormation), atau secara manual dari konsol Deadline Cloud. CloudFormation Template menyebarkan pertanian lengkap dengan antrian produksi dan antrian pembuatan paket yang sudah dikonfigurasi. Membuat antrian dari konsol memberi Anda kontrol lebih besar atas pengaturan individual.

Buat antrian pembuatan paket dengan CloudFormation

Anda dapat menggunakan CloudFormation template untuk membuat Deadline Cloud farm yang menyertakan antrian pembuatan paket. Template mengonfigurasi antrian produksi dan antrian pembuatan paket dengan saluran conda Amazon S3 pribadi.

Sebelum Anda menerapkan template, buat bucket Amazon S3 untuk menyimpan lampiran pekerjaan dan saluran conda Anda. Anda dapat membuat ember dari konsol Amazon S3. Anda memerlukan nama bucket saat menerapkan template.

Untuk menyebarkan template CloudFormation

1. Unduh deadline-cloud-starter-farmtemplate -template.yaml dari repositori sampel Deadline Cloud pada. GitHub
2. Dari CloudFormation konsol, pilih Buat Tumpukan, lalu Dengan sumber daya baru (standar).
3. Pilih opsi untuk mengunggah file templat, lalu unggah `deadline-cloud-starter-farm-template.yaml` file tersebut.
4. Masukkan nama untuk tumpukan, seperti**`StarterFarm`**, dan berikan nama bucket Amazon S3 untuk lampiran pekerjaan dan saluran conda.
5. Ikuti langkah-langkah CloudFormation konsol untuk menyelesaikan pembuatan tumpukan.

Untuk informasi selengkapnya tentang parameter template dan opsi penyesuaian, lihat README starter farm di repositori sampel Deadline Cloud pada. GitHub

Buat antrian pembuatan paket dari konsol

Ikuti petunjuk di Buat antrean di Panduan Pengguna Cloud Deadline. Lakukan perubahan berikut:

- Pada langkah 5, pilih bucket Amazon S3 yang ada. Tentukan nama folder root seperti **`DeadlineCloudPackageBuild`** agar artefak build tetap terpisah dari lampiran Deadline Cloud normal Anda.
- Pada langkah 6, Anda dapat mengaitkan antrian pembuatan paket dengan armada yang sudah ada, atau Anda dapat membuat armada yang sama sekali baru jika armada Anda saat ini tidak cocok.
- Pada langkah 9, buat peran layanan baru untuk antrian pembuatan paket Anda. Anda akan memodifikasi izin untuk memberikan antrian izin yang diperlukan untuk mengunggah paket dan mengindeks ulang saluran conda.

Konfigurasikan izin antrian pembuatan paket

Untuk mengizinkan antrian pembuatan paket mengakses `/Conda` awalan di bucket Amazon S3 antrean, Anda harus mengubah peran antrian untuk memberikan akses. read/write Peran tersebut memerlukan izin berikut agar pekerjaan pembuatan paket dapat mengunggah paket baru dan mengindeks ulang saluran.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrean pembuatan paket.
2. Pilih peran layanan antrian, lalu pilih Edit antrian.
3. Gulir ke bagian Peran layanan antrian, lalu pilih Lihat peran ini di konsol IAM.
4. Dari daftar kebijakan izin, pilih antrian AmazonDeadlineCloudQueuePolicyuntuk Anda.
5. Dari tab Izin, pilih Edit.
6. Tambahkan bagian baru ke peran layanan antrian seperti berikut ini. Ganti *`amzn-s3-demo-bucket`* dan *`111122223333`* dengan ember dan akun Anda sendiri.

```
{
   "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"
    }
   }
  },
```

Kirim pekerjaan pembuatan paket

Setelah membuat antrean pembuatan paket dan mengonfigurasi izin antrian, Anda dapat mengirimkan pekerjaan untuk membangun paket conda. `submit-package-job`Skrip di repositori sampel Deadline Cloud saat GitHub mengirimkan pekerjaan build untuk resep conda.

Anda membutuhkan yang berikut ini:

- The Deadline Cloud CLI diinstal pada workstation Anda.
- Sesi login AWS Deadline Cloud monitor (Deadline Cloud monitor) yang aktif.
- Klon dari repositori sampel Deadline Cloud.

Untuk mengirimkan pekerjaan pembuatan paket

1. Buka GUI konfigurasi Deadline Cloud dan atur farm default dan antrean ke antrian pembuatan paket Anda.

```
deadline config gui
```

2. Ubah ke `conda_recipes` direktori di repositori sampel.

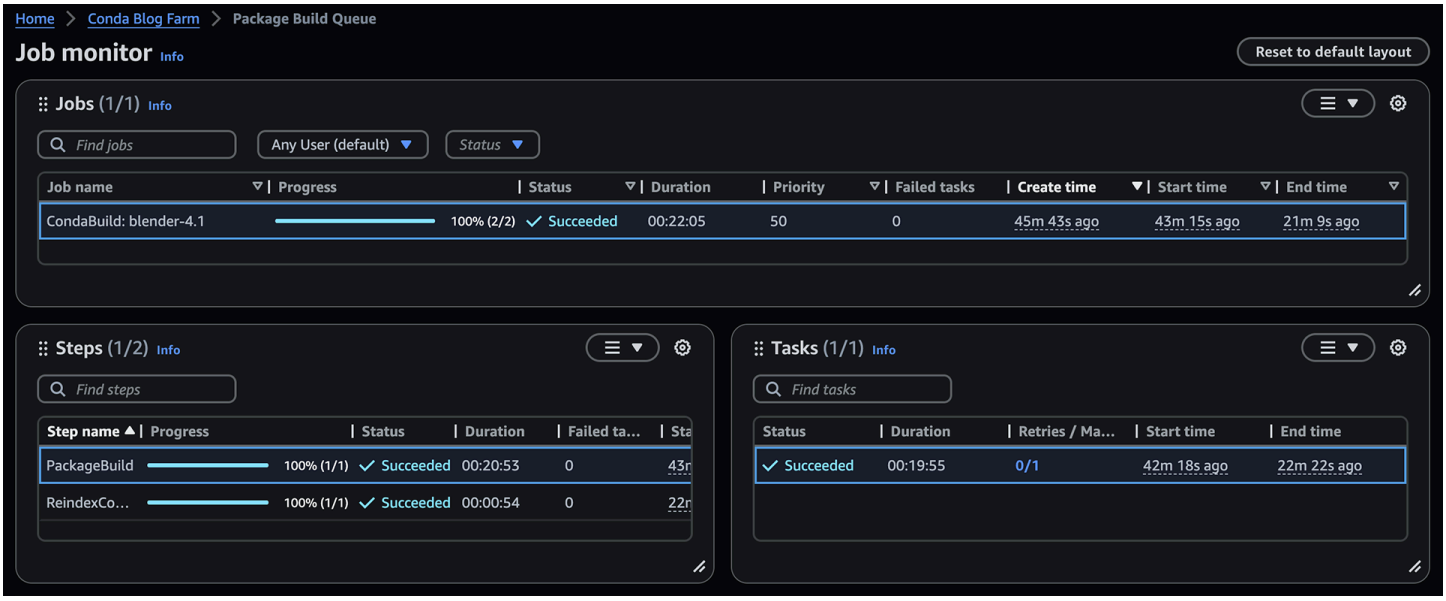
```
cd deadline-cloud-samples/conda_recipes
```

3. Jalankan `submit-package-job` skrip dengan direktori resep. Contoh berikut membangun resep Blender 4.5.

```
./submit-package-job blender-4.5/
```

   Jika resep memerlukan arsip sumber yang belum Anda unduh, skrip menyediakan instruksi unduhan. Unduh arsip dan jalankan skrip lagi.

Setelah Anda mengirimkan pekerjaan, gunakan monitor Deadline Cloud untuk melihat kemajuan dan status pekerjaan.



Monitor menunjukkan dua langkah pekerjaan: membangun paket dan kemudian mengindeks ulang saluran conda. Ketika Anda mengklik kanan pada tugas untuk langkah pembuatan paket dan memilih Lihat log, monitor akan menampilkan tindakan sesi:

- Sinkronkan lampiran - Menyalin lampiran pekerjaan input atau memasang sistem file virtual.
- Luncurkan Conda - Tindakan lingkungan antrian. Pekerjaan build tidak menentukan paket conda, jadi tindakan ini selesai dengan cepat.

- Luncurkan CondaBuild Env - Menciptakan lingkungan virtual conda dengan perangkat lunak yang diperlukan untuk membangun paket conda dan mengindeks ulang saluran.
- Jalankan tugas - Membangun paket dan mengunggah hasilnya ke Amazon S3.

Saat tindakan berjalan, mereka mengirim log ke Amazon CloudWatch (CloudWatch). Saat pekerjaan selesai, pilih Lihat log untuk semua tugas guna melihat log tambahan tentang penyiapan dan pembongkaran lingkungan.

Jalankan skrip konfigurasi host dengan hak administrator

Skrip konfigurasi host memungkinkan Anda untuk melakukan tugas-tugas administratif, seperti instalasi perangkat lunak, pada pekerja armada yang dikelola layanan Anda. Skrip ini berjalan dengan hak istimewa yang tinggi (`sudo`aktifLinux, Administrator aktifWindows), memberi Anda fleksibilitas untuk mengonfigurasi pekerja Anda untuk sistem Anda.

Deadline Cloud menjalankan skrip setelah pekerja memasuki `STARTING` status dan sebelum menjalankan tugas apa pun.

> **Important**
>
> Skrip berjalan dengan izin yang ditinggikan. Adalah tanggung jawab Anda untuk memastikan bahwa skrip tidak menimbulkan masalah keamanan apa pun.
> Saat Anda menggunakan skrip konfigurasi host, Anda bertanggung jawab untuk memantau kesehatan armada Anda.

Penggunaan umum untuk skrip konfigurasi host meliputi:

- Menginstal perangkat lunak yang membutuhkan akses administrator
- Memasang Docker kontainer
- Menginstal solusi penyimpanan cloud pihak ketiga sepertiLucidLink. Untuk penelusurannya, lihat Mengatur LucidLink dengan skrip armada terkelola layanan untuk Deadline Cloud di Blog for M&E. AWS

Anda dapat membuat dan memperbarui skrip konfigurasi host menggunakan konsol atau menggunakan file AWS CLI.

Console

1. Pada halaman Detail armada, pilih tab Konfigurasi.
2. Di bidang Script, masukkan skrip untuk dijalankan dengan izin yang ditinggikan. Anda dapat memilih Impor untuk memuat skrip dari workstation Anda.
3. Tetapkan periode batas waktu dalam hitungan detik untuk menjalankan skrip. Default-nya adalah 300 detik (5 menit).
4. Pilih Simpan perubahan untuk menyimpan skrip.

Create with CLI

Gunakan AWS CLI perintah berikut untuk membuat armada dengan skrip konfigurasi host. Ganti *placeholder* teks dengan informasi Anda.

```
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

Gunakan AWS CLI perintah berikut untuk memperbarui skrip konfigurasi host armada. Ganti *placeholder* teks dengan informasi Anda.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
```

```
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Skrip berikut menunjukkan:

- Variabel lingkungan yang tersedia untuk skrip
- AWS Kredensi itu berfungsi di shell
- Bahwa skrip berjalan di shell yang ditinggikan

Linux

Gunakan skrip berikut untuk menunjukkan bahwa skrip berjalan dengan `root` hak istimewa:

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Gunakan PowerShell skrip berikut untuk menunjukkan bahwa skrip berjalan dengan hak istimewa Administrator:

```
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
```

Memecahkan masalah skrip konfigurasi host

Saat Anda menjalankan skrip konfigurasi host:

- Tentang kesuksesan: Pekerja menjalankan pekerjaan
- Pada kegagalan (kode keluar bukan nol atau crash):
  - Pekerja menutup

  Armada secara otomatis meluncurkan pekerja baru menggunakan skrip konfigurasi host terbaru

Untuk memantau skrip:

1. Buka halaman armada di konsol Deadline Cloud.
2. Pilih Lihat pekerja untuk membuka monitor Deadline Cloud.
3. Lihat status pekerja di halaman monitor.

> **Tip**
>
> Saat menguji skrip konfigurasi host, atur jumlah pekerja maksimum armada ke 1 untuk menghindari memulai beberapa pekerja saat melakukan iterasi pada skrip.

Catatan penting:

- Pekerja yang ditutup karena kesalahan tidak tersedia dalam daftar pekerja di monitor. Gunakan CloudWatch Log untuk melihat log pekerja di grup log berikut:

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  Dalam grup log itu, cari aliran bernama`worker-ZZZZZ`.
- CloudWatch Log menyimpan log pekerja sesuai dengan periode retensi yang dikonfigurasi.

Pantau eksekusi skrip konfigurasi host

Dengan skrip konfigurasi host, Anda dapat mengambil kendali penuh dari pekerja Deadline Cloud. Anda dapat menginstal paket perangkat lunak apa pun, mengkonfigurasi ulang parameter sistem operasi, atau memasang sistem file bersama. Dengan fitur canggih ini dan kemampuan Deadline Cloud untuk menskalakan ke ribuan pekerja, Anda dapat memantau kapan skrip konfigurasi berhasil dijalankan atau gagal.

Kami merekomendasikan solusi berikut untuk memantau eksekusi skrip konfigurasi host.

CloudWatch Pemantauan log

Semua log konfigurasi host armada dialirkan ke grup CloudWatch log armada, dan secara khusus ke aliran CloudWatch log pekerja. Misalnya, `/aws/deadline/farm-123456789012/fleet-777788889999` adalah grup log untuk pertanian`123456789012`, armada`777788889999`.

Setiap pekerja menyediakan aliran log khusus, misalnya`worker-123456789012`. Log konfigurasi host termasuk spanduk log seperti Running Host Configuration Script dan Selesai menjalankan Host Configuration Script, kode keluar: 0. Kode keluar skrip disertakan dalam spanduk yang sudah jadi dan dapat ditanyakan menggunakan CloudWatch alat.

CloudWatch Wawasan Log

CloudWatch Logs Insights menawarkan kemampuan canggih untuk menganalisis informasi log. Misalnya, kueri Log Insights berikut mengurai kode keluar konfigurasi host, diurutkan berdasarkan waktu:

```
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
```

Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat Menganalisis data CloudWatch log dengan Wawasan Log di Panduan Pengguna CloudWatch Log Amazon.

Pencatatan terstruktur agen pekerja

Agen pekerja Deadline Cloud menerbitkan log JSON terstruktur ke. CloudWatch Agen pekerja menawarkan beragam log terstruktur untuk menganalisis kesehatan pekerja. Untuk informasi selengkapnya, lihat Deadline Login agen pekerja Cloud. GitHub

Atribut log terstruktur dibongkar ke bidang di Wawasan Log. Anda dapat menggunakan CloudWatch kemampuan ini untuk menghitung dan menganalisis kegagalan startup konfigurasi host. Misalnya, kueri count dan bin dapat digunakan untuk menentukan seberapa sering kegagalan terjadi:

```
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 filter metrik untuk metrik dan mengkhawatirkan

Anda dapat mengatur filter CloudWatch metrik untuk menghasilkan CloudWatch metrik dari log. Filter metrik memungkinkan Anda membuat alarm dan dasbor untuk memantau eksekusi skrip konfigurasi host.

Untuk membuat filter metrik

1. Buka CloudWatch konsol.
2. Di panel navigasi, pilih Log, lalu Log grup.
3. Pilih grup log armada Anda.
4. Pilih Create metric filter (Buat filter metrik).
5. Tentukan pola filter Anda menggunakan salah satu dari berikut ini:
   - Untuk metrik sukses:

     ```
     {$.message = "*Worker Agent host configuration succeeded.*"}
     ```

   - Untuk metrik kegagalan:

     ```
     {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
      exit code*"}
     ```

6. Pilih Berikutnya untuk membuat metrik dengan nilai berikut:
   - Namespace metrik: Namespace metrik Anda (misalnya,) **`MyDeadlineFarm`**
   - Nama metrik: Nama metrik yang Anda minta (misalnya,**`host_config_failure`**)
   - Nilai metrik: **`1`** (setiap contoh adalah hitungan 1)
   - Nilai default: Biarkan kosong
   - Satuan: **`Count`**

Setelah membuat filter metrik, Anda dapat mengonfigurasi CloudWatch alarm standar untuk mengambil tindakan pada tingkat kegagalan konfigurasi host yang meningkat, atau menambahkan metrik ke CloudWatch dasbor untuk day-to-day pengoperasian dan pemantauan.

Untuk detail selengkapnya, lihat Filter dan sintaks pola di Panduan Pengguna Amazon CloudWatch Logs.

Menggunakan lisensi perangkat lunak dengan Deadline Cloud

Deadline Cloud menyediakan dua metode untuk menyediakan lisensi perangkat lunak untuk pekerjaan Anda:

- Lisensi berbasis penggunaan (UBL) — melacak dan menagih berdasarkan jumlah jam yang digunakan armada Anda untuk memproses pekerjaan. Tidak ada jumlah lisensi yang ditetapkan sehingga armada Anda dapat menskalakan sesuai kebutuhan. UBL adalah standar untuk armada yang dikelola layanan. Untuk armada yang dikelola pelanggan, Anda dapat menghubungkan titik akhir lisensi Deadline Cloud untuk UBL. UBL menyediakan lisensi untuk pekerja Deadline Cloud Anda untuk dirender, itu tidak memberikan lisensi untuk aplikasi DCC Anda.
- Bawa lisensi Anda sendiri (BYOL) — memungkinkan Anda untuk menggunakan lisensi perangkat lunak yang ada dengan armada yang dikelola layanan atau pelanggan Anda. Anda dapat menggunakan BYOL untuk terhubung ke server lisensi untuk perangkat lunak yang tidak didukung oleh lisensi berbasis penggunaan Deadline Cloud. Anda dapat menggunakan BYOL dengan armada yang dikelola layanan dengan menghubungkan ke server lisensi khusus.

Topik

Menggabungkan BYOL dan UBL

Anda dapat menggabungkan BYOL dan UBL sehingga pekerja Anda menggunakan lisensi yang ada terlebih dahulu dan secara otomatis kembali ke lisensi berbasis penggunaan Deadline Cloud ketika lisensi BYOL Anda habis. Pendekatan ini berguna ketika Anda memiliki sejumlah lisensi yang ada tetapi perlu skala di luar kapasitas itu selama beban kerja puncak.

Cara kerja lisensi gabungan

Saat Anda mengonfigurasi lisensi gabungan, lingkungan antrian mengatur variabel lingkungan lisensi sehingga server lisensi BYOL terdaftar sebelum titik akhir lisensi UBL. Sebagian besar aplikasi pihak

ketiga memeriksa server lisensi dalam urutan mereka muncul dalam variabel lingkungan. Ketika seorang pekerja meminta lisensi, aplikasi pertama-tama menghubungi server lisensi BYOL Anda. Jika tidak ada lisensi BYOL yang tersedia, aplikasi kembali ke titik akhir lisensi UBL.

Template lingkungan antrian BYOL yang disediakan dalam Connect armada yang dikelola layanan ke server lisensi kustom mengonfigurasi perilaku fallback ini secara otomatis. Skrip Python di lingkungan antrian menambahkan alamat server lisensi BYOL Anda ke variabel lingkungan lisensi UBL yang ada. Untuk menggunakan lisensi gabungan, simpan bagian UBL dalam skrip lingkungan antrian untuk produk yang Anda inginkan perilaku fallback.

Untuk hanya menggunakan BYOL tanpa fallback UBL untuk produk tertentu, hapus bagian UBL untuk produk tersebut dari skrip dan tambahkan variabel lingkungan lisensi langsung ke `variables` bagian lingkungan antrian. Misalnya, untuk hanya menggunakan BYOL untuk Cinema 4D, hapus bagian Cinema 4D dari skrip dan tambahkan `g_licenseServerRLM: 127.0.0.1:7057` ke bagian. `variables`

Contoh: Menggunakan lisensi BYOL Cinema 4D dengan fallback UBL

Pertimbangkan studio yang memiliki lisensi Cinema 4D yang ada di server lisensi di jaringan lokal mereka. Studio ingin menggunakan lisensi tersebut untuk rendering Deadline Cloud, tetapi juga ingin skala melampaui jumlah lisensi mereka dengan kembali ke UBL ketika semua lisensi BYOL digunakan.

Untuk mengonfigurasi pengaturan ini, ikuti langkah-langkah Connect armada yang dikelola layanan ke server lisensi kustom dan buat perubahan berikut pada templat lingkungan antrian:

Untuk mengonfigurasi lisensi BYOL Cinema 4D dengan fallback UBL

1. Setel `LicenseInstanceId` parameter ke ID instans Amazon Elastic Compute Cloud (Amazon EC2) dari server lisensi atau proxy yang memiliki akses ke server lisensi Cinema 4D.
2. Atur `LicensePorts` parameter untuk menyertakan port `7057` (port lisensi Cinema 4D RLM).
3. Dalam skrip Python, simpan bagian Cinema 4D yang menambahkan server BYOL ke konfigurasi 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']}")
```

Konfigurasi ini disetel `g_licenseServerRLM` ke`127.0.0.1:7057;UBL_endpoint:7057`. Cinema 4D memeriksa server BYOL pada awalnya`127.0.0.1:7057`. Jika tidak ada lisensi yang tersedia, Cinema 4D kembali ke titik akhir UBL.

4. Hapus bagian untuk produk yang tidak Anda gunakan (misalnya, Arnold, Nuke, atau SideFX) untuk menjaga konfigurasi tetap bersih.

Jika Anda juga memiliki produk lain yang hanya menggunakan BYOL tanpa fallback UBL, tambahkan variabel lingkungan lisensi tersebut langsung ke `variables` bagian lingkungan antrian dan hapus bagian yang sesuai dari skrip Python.

Pertimbangan untuk lisensi gabungan

Ingatlah pertimbangan berikut saat Anda menggunakan lisensi gabungan:

- Beberapa aplikasi tidak mendukung beberapa server lisensi dalam satu variabel lingkungan. Misalnya, V-Ray menggunakan file konfigurasi XMLsebagai gantinya. Template lingkungan antrian menangani konfigurasi V-Ray secara terpisah. Untuk informasi selengkapnya, lihat bagian V-Ray di templat lingkungan antrian di. Connect armada yang dikelola layanan ke server lisensi kustom
- Urutan server lisensi dalam variabel lingkungan menentukan prioritas. Buat daftar server BYOL terlebih dahulu sehingga lisensi Anda yang ada digunakan sebelum lisensi UBL.
- Pada Windows pekerja, pisahkan entri server lisensi dalam variabel lingkungan dengan titik koma (), bukan titik dua (`;`). `:` Untuk informasi selengkapnya tentang mengonfigurasi variabel lingkungan lisensi, lihatConnect armada yang dikelola pelanggan ke titik akhir lisensi.
- Untuk menggunakan fallback UBL dengan armada yang dikelola pelanggan, siapkan titik akhir lisensi. Lihat informasi yang lebih lengkap di Connect armada yang dikelola pelanggan ke titik akhir lisensi.

Connect armada yang dikelola layanan ke server lisensi kustom

Anda dapat membawa server lisensi Anda sendiri untuk digunakan dengan armada yang dikelola layanan Deadline Cloud. Untuk membawa lisensi Anda sendiri, Anda dapat mengonfigurasi server lisensi menggunakan lingkungan antrian di peternakan Anda. Untuk mengonfigurasi server lisensi Anda, Anda harus sudah menyiapkan pertanian dan antrian.

Bagaimana Anda terhubung ke server lisensi perangkat lunak tergantung pada konfigurasi armada Anda dan persyaratan vendor perangkat lunak. Biasanya, Anda mengakses server dengan salah satu dari dua cara:

- Langsung ke server lisensi. Pekerja Anda mendapatkan lisensi dari server lisensi vendor perangkat lunak menggunakan Internet. Semua pekerja Anda harus dapat terhubung ke server.
- Melalui proxy lisensi. Pekerja Anda terhubung ke server proxy di jaringan lokal Anda. Hanya server proxy yang diizinkan untuk terhubung ke server lisensi vendor melalui Internet.

Dengan petunjuk di bawah ini, Anda menggunakan Amazon EC2 Systems Manager (SSM) untuk meneruskan port dari instans pekerja ke server lisensi atau instans proxy Anda. Dalam contoh di bawah ini jika server lisensi Anda tidak dapat memberikan lisensi, lisensi berbasis penggunaan Deadline Cloud akan digunakan. Hapus bagian yang tidak berlaku untuk pipeline atau produk yang tidak ingin Anda gunakan lisensi berbasis penggunaan setelah lisensi Anda habis.

Topik

Langkah 1: Konfigurasikan lingkungan antrian

Anda dapat mengonfigurasi lingkungan antrian dalam antrian untuk mengakses server lisensi Anda. Pertama, pastikan bahwa Anda memiliki AWS instance yang dikonfigurasi dengan akses server lisensi menggunakan salah satu metode berikut:

- Server lisensi — Instance meng-host server lisensi secara langsung.
- Proxy lisensi — Instance memiliki akses jaringan ke server lisensi, dan meneruskan port server lisensi ke server lisensi. Untuk detail tentang cara mengonfigurasi instance proxy lisensi, lihatLangkah 2: (Opsional) Pengaturan instance proxy lisensi.

Untuk informasi tentang mengonfigurasi variabel lingkungan lisensi, lihatLangkah 3: Hubungkan aplikasi rendering ke titik akhir. Untuk penyiapan server lisensi khusus, alamat server lisensi tetap localhost, bukan titik akhir Amazon VPC.

Untuk menambahkan izin yang diperlukan ke peran antrian

1. Dari konsol Cloud Deadline, pilih Buka Dasbor.
2. Dari dasbor, pilih pertanian, lalu antrian yang ingin Anda konfigurasikan.
3. Dari detail antrian > peran layanan, pilih peran.
4. Pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.
5. Pilih editor kebijakan JSON, lalu salin dan tempel teks berikut ke 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. Sebelum menyimpan kebijakan baru, ganti nilai berikut dalam teks kebijakan:
   - Ganti `region` dengan AWS Wilayah tempat peternakan Anda berada
   - Ganti `instance_id` dengan ID instance untuk server lisensi atau instance proxy yang Anda gunakan
   - Ganti `account_id` dengan nomor AWS rekening yang berisi peternakan Anda
7. Pilih Berikutnya.
8. Untuk nama Policy, masukkan**`LicenseForwarding`**.
9. Pilih Buat kebijakan untuk menyimpan perubahan dan membuat kebijakan dengan izin yang diperlukan.

Untuk menambahkan lingkungan antrian baru ke antrian

1. Dari konsol Deadline Cloud, pilih Go to Dashboard jika Anda belum melakukannya.
2. Dari dasbor, pilih pertanian, lalu antrian yang ingin Anda konfigurasikan.
3. Pilih Lingkungan Antrian > Tindakan > Buat baru dengan YAMG.
4. Salin dan tempel teks berikut ke editor skrip YAMG.

   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. Sebelum menyimpan lingkungan antrian, buat perubahan berikut pada teks lingkungan sesuai kebutuhan:

- Perbarui nilai default untuk parameter berikut untuk mencerminkan lingkungan Anda:
  - LicenseInstanceID - ID instans Amazon EC2 dari server lisensi atau instans proxy Anda
  - LicenseInstanceRegion— AWS Wilayah yang berisi peternakan Anda
  - LicensePorts— Daftar port yang dipisahkan koma untuk diteruskan ke server lisensi atau instance proxy (misalnya 2700,2701)
- Jika Anda ingin menggunakan lisensi berbasis penggunaan (UBL) setelah Bring your own license (BYOL) habis pastikan port sudah benar untuk server lisensi Anda. Jika Anda tidak ingin menggunakan UBL setelah kehabisan BYOL, tambahkan variabel lingkungan lisensi yang diperlukan ke bagian variabel.

  Variabel-variabel ini harus mengarahkan DCCs ke localhost pada port server lisensi. Misalnya, jika server lisensi Foundry Anda mendengarkan pada port 6101, Anda akan menambahkan variabel sebagai. **`foundry_LICENSE: 6101@localhost`**

6. (Opsional) Anda dapat membiarkan Prioritas disetel ke 0, atau Anda dapat mengubahnya untuk mengurutkan prioritas secara berbeda di antara beberapa lingkungan antrian.
7. Pilih Buat lingkungan antrian untuk menyelamatkan lingkungan baru.

   Dengan pengaturan lingkungan antrian, pekerjaan yang dikirimkan ke antrian ini akan mengambil lisensi dari server lisensi yang dikonfigurasi.

Langkah 2: (Opsional) Pengaturan instance proxy lisensi

Sebagai alternatif untuk menggunakan server lisensi, Anda dapat menggunakan proxy lisensi. Untuk membuat proxy lisensi, buat instans Amazon Linux 2023 baru yang memiliki akses jaringan ke server lisensi. Jika perlu, Anda dapat mengonfigurasi akses ini menggunakan koneksi VPN. Untuk informasi selengkapnya, lihat Koneksi VPN di Panduan Pengguna Amazon VPC.

Untuk menyiapkan instance proxy lisensi untuk Deadline Cloud, ikuti langkah-langkah dalam prosedur ini. Lakukan langkah-langkah konfigurasi berikut pada instance baru ini untuk mengaktifkan penerusan lalu lintas lisensi ke server lisensi Anda

1. Untuk menginstal HAProxy paket, masukkan

```
sudo yum install haproxy
```

2. Perbarui bagian server lisensi dengarkan dari file konfigurasi/etc/haproxy/haproxy.cfg dengan yang berikut ini:

   a. Ganti LicensePort1 dan LicensePort2 dengan nomor port yang akan diteruskan ke server lisensi. Tambahkan atau hapus nilai yang dipisahkan koma untuk mengakomodasi jumlah port yang diperlukan.

   b. Ganti LicenseServerHostdengan nama host atau alamat IP server lisensi.

```
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. Untuk mengaktifkan dan memulai HAProxy layanan, jalankan perintah berikut:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Setelah menyelesaikan langkah-langkah, permintaan lisensi yang dikirim ke localhost dari lingkungan antrian penerusan harus diteruskan ke server lisensi yang ditentukan.

Langkah 3: pengaturan CloudFormation template

Anda dapat menggunakan CloudFormation template untuk mengonfigurasi seluruh peternakan untuk menggunakan lisensi Anda sendiri.

1. Ubah template yang disediakan di langkah berikutnya untuk menambahkan variabel lingkungan lisensi yang diperlukan ke bagian variabel di bawah BYOLQueueLingkungan.
2. Gunakan CloudFormation template berikut.

```
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. Saat menerapkan CloudFormation template, berikan parameter berikut:

   - Memperbarui LicenseInstanceID dengan ID Instans Amazon EC2 dari server lisensi atau instans proxy Anda
   - Perbarui LicensePortsdengan daftar port yang dipisahkan koma untuk diteruskan ke server lisensi atau instance proxy (misalnya 2700,2701)
   - Tambahkan variabel lingkungan lisensi dengan mengganti **`example_LICENSE: 2700@localhost`** dalam template
4. Terapkan template untuk menyiapkan peternakan Anda dengan membawa kemampuan lisensi Anda sendiri.

Connect armada yang dikelola pelanggan ke titik akhir lisensi

Server lisensi berbasis penggunaan AWS Deadline Cloud menyediakan lisensi sesuai permintaan untuk produk pihak ketiga tertentu. Dengan lisensi berbasis penggunaan, Anda dapat membayar sesuai keinginan Anda. Anda hanya dikenakan biaya untuk waktu yang Anda gunakan. Lisensi berbasis penggunaan menyediakan lisensi untuk dirender oleh pekerja Cloud Deadline Anda, tidak memberikan lisensi untuk aplikasi DCC Anda.

Server lisensi berbasis penggunaan Deadline Cloud dapat digunakan dengan jenis armada apa pun selama pekerja Deadline Cloud dapat berkomunikasi dengan server lisensi. Server lisensi secara otomatis diatur dalam armada yang dikelola layanan. Pengaturan berikut hanya diperlukan untuk armada yang dikelola pelanggan.

Untuk membuat server lisensi, Anda memerlukan grup keamanan untuk VPC pertanian Anda yang memungkinkan lalu lintas untuk lisensi pihak ketiga.

Topik

Langkah 1: Buat grup keamanan

Gunakan Konsol VPC Amazon untuk membuat grup keamanan untuk VPC pertanian Anda. Konfigurasikan grup keamanan untuk mengizinkan aturan masuk berikut:

- Autodesk Maya dan Arnold - 2701 - 2702, TCP,, IPv4 IPv6
- Bioskop 4D — 7057, TCP,, IPv4 IPv6
- Pengecoran Nuke — 6101, TCP,, IPv4 IPv6
- Raksasa Merah — 7055, TCP, IPV4
- Pergeseran Merah — 7054, TCP,, IPv4 IPv6
- SidFX Houdini, Mantra, dan Karma — 1715 - 1717, TCP,, IPv4 IPv6
- VRay — 30304, TCP, IPV4

Sumber untuk setiap aturan masuk adalah kelompok keamanan pekerja armada.

Untuk informasi selengkapnya tentang membuat grup keamanan, lihat Membuat grup keamanan di panduan pengguna Amazon Virtual Private Cloud.

Langkah 2: Siapkan titik akhir lisensi

Titik akhir lisensi menyediakan akses ke server lisensi untuk produk pihak ketiga. Permintaan lisensi dikirim ke titik akhir lisensi. Titik akhir merutekan mereka ke server lisensi yang sesuai. Server lisensi

melacak batas penggunaan dan hak. Membuat titik akhir lisensi di Deadline Cloud menyediakan titik akhir AWS PrivateLink antarmuka di VPC Anda. Titik akhir ini ditagih sesuai dengan harga standar AWS PrivateLink . Untuk informasi selengkapnya, lihat harga AWS PrivateLink.

Dengan izin yang sesuai, Anda dapat membuat titik akhir lisensi Anda. Untuk kebijakan yang diperlukan untuk membuat titik akhir lisensi, lihat Kebijakan untuk mengizinkan pembuatan titik akhir lisensi.

Anda dapat membuat titik akhir lisensi dari dasbor di konsol Deadline Cloud.

1. Dari panel navigasi kiri, pilih titik akhir Lisensi, lalu pilih Buat titik akhir lisensi.
2. Dari halaman Create license endpoint, lengkapi yang berikut ini:
   - Pilih VPC.
   - Pilih subnet yang berisi pekerja Deadline Cloud Anda. Anda dapat memilih hingga 10 subnet.
   - Pilih grup keamanan yang Anda buat di langkah 1. Anda dapat memilih hingga 10 grup keamanan untuk skenario yang lebih rumit.
   - (Opsional) Pilih Tambahkan tag baru dan tambahkan satu atau beberapa tag. Anda dapat menambahkan hingga 50 tag.
3. Pilih Buat titik akhir lisensi. Ketika titik akhir lisensi dibuat, itu akan ditampilkan pada halaman titik akhir lisensi.
4. Dari bagian produk terukur, pilih Tambahkan produk, lalu pilih produk yang ingin Anda tambahkan ke titik akhir lisensi Anda. Pilih Tambahkan.

Untuk menghapus produk dari titik akhir lisensi, di bagian produk terukur, pilih produk dan kemudian pilih Hapus. Dalam konfirmasi, pilih Hapus lagi.

Langkah 3: Hubungkan aplikasi rendering ke titik akhir

Setelah titik akhir lisensi diatur, aplikasi menggunakannya sama seperti mereka menggunakan server lisensi pihak ketiga. Anda biasanya mengonfigurasi server lisensi untuk aplikasi dengan menetapkan variabel lingkungan atau pengaturan sistem lainnya, seperti kunci registri Microsoft Windows, ke port dan alamat server lisensi.

Untuk mendapatkan nama DNS titik akhir lisensi, pilih titik akhir lisensi di konsol dan kemudian pilih ikon salin di bagian Nama DNS.

Contoh konfigurasi

Example Autodesk Maya dan Arnold

> **Note**
>
> Anda dapat menggunakan Autodesk Maya dan Arnold bersama-sama atau secara terpisah. Gunakan port 2702 untuk Autodesk Maya dan port 2701 untuk Arnold.

Untuk Autodesk Maya, atur variabel lingkungan `ADSKFLEX_LICENSE_FILE` ke:

```
2702@VPC_Endpoint_DNS_Name
```

Untuk Arnold, atur variabel lingkungan `ADSKFLEX_LICENSE_FILE` ke:

```
2701@VPC_Endpoint_DNS_Name
```

Untuk Autodesk Maya dan Arnold, atur variabel `ADSKFLEX_LICENSE_FILE` lingkungan ke:

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Untuk Windows pekerja, gunakan titik koma (;) alih-alih titik dua (:) untuk memisahkan titik akhir.

Example— Bioskop 4D

Atur variabel lingkungan `g_licenseServerRLM` ke:

```
VPC_Endpoint_DNS_Name:7057
```

Setelah Anda membuat variabel lingkungan, Anda harus dapat merender gambar menggunakan baris perintah yang mirip dengan yang ini:

```
"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— Pengecoran Nuke

Atur variabel lingkungan `foundry_LICENSE` ke:

```
6101@VPC_Endpoint_DNS_Name
```

Untuk menguji bahwa lisensi berfungsi dengan baik, Anda dapat menjalankan Nuke di terminal:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Raksasa Merah

Atur variabel lingkungan `redshift_LICENSE` ke:

```
7055@VPC_Endpoint_DNS_Name
```

Perhatikan bahwa Raksasa Merah dan Pergeseran Merah memiliki variabel `redshift_LICENSE` lingkungan yang sama. Jika Anda ingin menggunakan kedua aplikasi, Anda dapat mengatur variabel lingkungan ke:

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Untuk Windows pekerja, gunakan titik koma (;) alih-alih titik dua (:) untuk memisahkan titik akhir.

Untuk menguji bahwa lisensi berfungsi dengan baik, pastikan Anda menginstal After Effects dan Red Giant. Kemudian, Anda dapat merender proyek menggunakan perintah yang mirip dengan yang ini:

```
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— Pergeseran Merah

Atur variabel lingkungan `redshift_LICENSE` ke:

```
7054@VPC_Endpoint_DNS_Name
```

Setelah Anda membuat variabel lingkungan, Anda harus dapat merender gambar menggunakan baris perintah yang mirip dengan yang ini:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SidFX Houdini, Mantra, dan Karma

Jalankan perintah berikut:

```
/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;"
```

Untuk menguji bahwa lisensi berfungsi dengan baik, Anda dapat merender adegan Houdini melalui perintah ini:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Atur variabel lingkungan `VRAY_AUTH_CLIENT_SETTINGS` ke:

```
licset://VPC_Endpoint_DNS_Name:30304
```

Atur variabel lingkungan `VRAY_AUTH_CLIENT_FILE_PATH` ke:

```
/null
```

Untuk menguji bahwa lisensi berfungsi dengan benar, Anda dapat merender gambar VRay menggunakan perintah yang mirip dengan yang ini:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Langkah 4: Hapus titik akhir lisensi

Saat menghapus armada yang dikelola pelanggan, ingatlah untuk menghapus titik akhir lisensi Anda. Jika Anda tidak menghapus titik akhir lisensi, Anda akan terus dikenakan biaya AWS PrivateLink tetap

Anda dapat menghapus titik akhir lisensi dari dasbor di konsol Deadline Cloud.

1. Dari panel navigasi kiri, pilih titik akhir Lisensi.
2. Pilih titik akhir yang ingin Anda hapus dan pilih hapus, lalu pilih hapus lagi untuk mengonfirmasi.

Menggunakan agen AI dengan Deadline Cloud

Gunakan agen AI untuk menulis paket pekerjaan, mengembangkan paket conda, dan memecahkan masalah pekerjaan di Deadline Cloud. Topik ini menjelaskan apa itu agen AI, poin-poin penting untuk bekerja dengan mereka secara efektif, dan sumber daya untuk membantu agen memahami Deadline Cloud.

Agen AI adalah alat perangkat lunak yang menggunakan model bahasa besar (LLM) untuk melakukan tugas secara mandiri. Agen AI dapat membaca dan menulis file, menjalankan perintah, dan mengulangi solusi berdasarkan umpan balik. Contohnya termasuk alat baris perintah seperti Kiro dan asisten terintegrasi IDE.

Poin penting untuk bekerja dengan agen AI

Poin-poin penting berikut membantu Anda mendapatkan hasil yang lebih baik ketika Anda menggunakan agen AI dengan Deadline Cloud.

- Menyediakan landasan — agen AI berkinerja terbaik ketika mereka memiliki akses ke dokumentasi, spesifikasi, dan contoh yang relevan. Anda dapat memberikan landasan dengan mengarahkan agen ke halaman dokumentasi tertentu, membagikan kode contoh yang ada sebagai referensi, mengkloning repositori open source yang relevan ke ruang kerja lokal, dan menyediakan dokumentasi untuk aplikasi pihak ketiga.
- Tentukan kriteria keberhasilan — Tentukan hasil yang diharapkan dan persyaratan teknis untuk agen. Misalnya, ketika Anda meminta agen untuk mengembangkan paket pekerjaan, tentukan input pekerjaan, parameter, dan output yang diharapkan. Jika Anda tidak yakin tentang spesifikasinya, mintalah agen untuk mengusulkan opsi terlebih dahulu, lalu perbaiki persyaratan bersama.
- Aktifkan loop umpan balik — agen AI melakukan iterasi lebih efektif ketika mereka dapat menguji solusi mereka dan menerima umpan balik. Alih-alih mengharapkan solusi yang berfungsi pada upaya pertama, berikan agen kemampuan untuk menjalankan solusinya dan meninjau hasilnya. Pendekatan ini bekerja dengan baik ketika agen dapat mengakses pembaruan status, log, dan kesalahan validasi. Misalnya, ketika Anda mengembangkan paket pekerjaan, izinkan agen untuk mengirimkan pekerjaan dan meninjau log.
- Berharap untuk mengulangi — Bahkan dengan konteks yang baik, agen dapat keluar jalur atau membuat asumsi yang tidak sesuai dengan lingkungan Anda. Amati bagaimana agen mendekati tugas dan memberikan bimbingan di sepanjang jalan. Tambahkan konteks yang hilang jika agen berjuang, membantu menemukan kesalahan dengan menunjuk ke file log tertentu, memperbaiki

persyaratan saat Anda menemukannya, dan menambahkan persyaratan negatif untuk secara eksplisit menyatakan apa yang harus dihindari agen.

Sumber daya untuk konteks agen

Sumber daya berikut membantu agen AI memahami konsep Deadline Cloud dan menghasilkan output yang akurat.

- Server Deadline Cloud Model Context Protocol (MCP) — Untuk agen yang mendukung Model Context Protocol, repositori deadline-cloud berisi klien Deadline Cloud yang mencakup server MCP untuk berinteraksi dengan pekerjaan.
- AWSDokumentasi MCP server — Untuk agen yang mendukung MCP, konfigurasikan server MCP AWS Dokumentasi untuk memberikan agen akses langsung ke AWS dokumentasi, termasuk Panduan Pengguna Cloud Tenggat Waktu dan Panduan Pengembang.
- Spesifikasi Open Job Description — Spesifikasi Open Job Description pada GitHub mendefinisikan skema untuk template pekerjaan. Referensikan repositori ini ketika agen perlu memahami struktur dan sintaks template pekerjaan.
- deadline-cloud-samples— deadline-cloud-samplesRepositori berisi contoh bundel pekerjaan, resep conda, dan CloudFormation template untuk aplikasi umum dan kasus penggunaan.
- aws-deadline GitHub organization — Organisasi aws-deadline GitHub berisi plugin referensi untuk banyak aplikasi pihak ketiga yang dapat Anda gunakan sebagai contoh untuk integrasi lainnya.

Contoh prompt: Menulis bundel pekerjaan

Contoh prompt berikut menunjukkan cara menggunakan agen AI untuk membuat bundel pekerjaan yang melatih adaptor LoRa (Adaptasi Peringkat Rendah) untuk menghasilkan gambar AI. Prompt mengilustrasikan poin-poin penting yang dibahas sebelumnya: ini memberikan landasan dengan menunjuk ke repositori yang relevan, mendefinisikan kriteria keberhasilan untuk output bundel pekerjaan, dan menguraikan loop umpan balik untuk pengembangan berulang.

```
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.
```

AWSBatas Waktu Pemantauan Cloud

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja AWS Deadline Cloud (Deadline Cloud) dan solusi AndaAWS. Kumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Sebelum Anda mulai memantau Deadline Cloud, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapakah yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

AWSdan Deadline Cloud menyediakan alat yang dapat Anda gunakan untuk memantau sumber daya Anda dan menanggapi potensi insiden. Beberapa alat ini melakukan pemantauan untuk Anda, beberapa alat memerlukan intervensi manual. Anda harus mengotomatiskan tugas pemantauan sebanyak mungkin.

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

  Deadline Cloud memiliki tiga CloudWatch metrik.

- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari EC2 instans Amazon CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.

- Amazon EventBridge dapat digunakan untuk mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem, seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis mana yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.
- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggilAWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

Topik

Logging panggilan Deadline Cloud API menggunakan AWS CloudTrail

Deadline Cloud terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk Deadline Cloud sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari Deadline Cloud konsol dan panggilan kode ke operasi Deadline Cloud API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat Deadline Cloud, alamat IP dari mana permintaan dibuat, kapan dibuat, dan detail tambahan.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Anda Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. Wilayah AWS Untuk informasi selengkapnya, lihat Bekerja dengan riwayat CloudTrail Acara di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara CloudTrailDanau.

CloudTrail jalan setapak

Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan Konsol Manajemen AWS Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan. AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. Wilayah AWS Untuk informasi selengkapnya tentang jejak, lihat Membuat jejak untuk Anda Akun AWS dan Membuat jejak untuk organisasi di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga. Untuk informasi tentang harga Amazon S3, lihat Harga Amazon S3.

CloudTrail Menyimpan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail Lake mengonversi peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC. ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara tingkat lanjut. Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat Bekerja dengan AWS CloudTrail Danau di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih opsi harga yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa,

dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Deadline Cloud peristiwa data di CloudTrail

Peristiwa data memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya (misalnya, membaca atau menulis ke objek Amazon S3). Ini juga dikenal sebagai operasi bidang data. Peristiwa data sering kali merupakan aktivitas bervolume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data. Riwayat CloudTrail peristiwa tidak merekam peristiwa data.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Anda dapat mencatat peristiwa data untuk jenis Deadline Cloud sumber daya menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Untuk informasi selengkapnya tentang cara mencatat peristiwa data, lihat Mencatat peristiwa data dengan Konsol Manajemen AWS dan Logging peristiwa data dengan AWS Command Line Interface di Panduan AWS CloudTrail Pengguna.

Tabel berikut mencantumkan jenis Deadline Cloud sumber daya yang dapat Anda log peristiwa data. Kolom tipe peristiwa data (konsol) menunjukkan nilai yang akan dipilih dari daftar tipe peristiwa Data di CloudTrail konsol. Kolom nilai resources.type menunjukkan **`resources.type`** nilai, yang akan Anda tentukan saat mengonfigurasi penyeleksi acara lanjutan menggunakan or. AWS CLI CloudTrail APIs CloudTrailKolom Data yang APIs dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk ke CloudTrail
Armada Batas Waktu	`AWS::Deadline::Fleet`	• SearchWorkers
Antrian Batas Waktu	`AWS::Deadline::Fleet`	• SearchJobs
Batas waktu Pekerja	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk ke CloudTrail
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

Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada `eventName`,`readOnly`, dan `resources.ARN` bidang untuk mencatat hanya peristiwa yang penting bagi Anda. Untuk informasi selengkapnya tentang bidang ini, lihat AdvancedFieldSelectordi Referensi AWS CloudTrail API.

Deadline Cloud acara manajemen di CloudTrail

Acara manajemen memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di Anda Akun AWS. Ini juga dikenal sebagai operasi bidang kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

AWS Deadline Cloud mencatat operasi bidang Deadline Cloud kontrol berikut ke CloudTrail sebagai peristiwa manajemen.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-baca
- assume-fleet-role-for-pekerja
- assume-queue-role-for-baca
- assume-queue-role-for-pengguna
- assume-queue-role-for-pekerja
- buat-anggaran
- buat-pertanian
- membuat-armada
- create-license-endpoint
- buat-batas
- buat-monitor
- buat-antrian
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- menciptakan-pekerja
- hapus-anggaran
- hapus-pertanian

- hapus-armada
- delete-license-endpoint
- hapus-batas
- delete-metered-product
- hapus-monitor
- hapus-antrian
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- hapus-pekerja
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- dapatkan-anggaran
- dapatkan-pertanian
- get-feature-map
- dapatkan-armada
- get-license-endpoint
- dapatkan-batas
- dapatkan-monitor
- antrian
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-antrian

- list-available-metered-products
- daftar-anggaran
- list-farm-members
- daftar-peternakan
- list-fleet-members
- daftar-armada
- list-job-members
- list-license-endpoints
- daftar-batas
- list-metered-products
- daftar-monitor
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- daftar-antrian
- list-storage-profiles
- list-storage-profiles-for-antrian
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-sumber daya
- untag-sumber daya
- pembaruan-anggaran
- perbaruan-pertanian
- perbaruan-armada
- batas pembaruan-
- pembaruan-monitor
- antrian pembaruan
- update-queue-environment

- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- pembaruan-pekerja

Deadline Cloud contoh acara

Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang operasi API yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga peristiwa tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail peristiwa yang menunjukkan `CreateFarm` operasi.

```
{
    "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",
}
```


Contoh JSON menunjukkan Wilayah AWS, alamat IP, dan "`requestParameters`" lainnya seperti "" dan `displayName` "`kmsKeyArn`" yang dapat membantu Anda mengidentifikasi acara.

Untuk informasi tentang konten CloudTrail rekaman, lihat konten CloudTrail rekaman di Panduan AWS CloudTrail Pengguna.

Pemantauan CloudWatch dengan

Amazon CloudWatch (CloudWatch) mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. Anda dapat membuka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/untuk melihat dan memfilter metrik Deadline Cloud.

Statistik ini disimpan selama 15 bulan sehingga Anda dapat mengakses informasi historis untuk mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau

mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Deadline Cloud memiliki dua jenis log — log tugas dan log pekerja. Log tugas adalah saat Anda menjalankan log eksekusi sebagai skrip atau saat DCC berjalan. Log tugas mungkin menampilkan peristiwa seperti pemuatan aset, rendering ubin, atau tekstur yang tidak ditemukan.

Log pekerja menunjukkan proses agen pekerja. Ini mungkin termasuk hal-hal seperti ketika agen pekerja memulai, mendaftarkan dirinya sendiri, melaporkan kemajuan, memuat konfigurasi, atau menyelesaikan tugas.

Namespace untuk log ini adalah. `/aws/deadline/*`

Untuk Deadline Cloud, pekerja mengunggah log ini ke CloudWatch Log. Secara default, log tidak pernah kedaluwarsa. Jika suatu pekerjaan menghasilkan volume data yang tinggi, Anda dapat dikenakan biaya tambahan. Untuk informasi selengkapnya, lihat CloudWatch harga Amazon.

Anda dapat menyesuaikan kebijakan penyimpanan untuk setiap grup log. Retensi yang lebih pendek menghilangkan log lama dan dapat membantu mengurangi biaya penyimpanan. Untuk menyimpan log, Anda dapat mengarsipkannya ke Amazon Simple Storage Service sebelum menghapus log. Untuk informasi selengkapnya, lihat Mengekspor data log ke Amazon S3 menggunakan konsol di panduan CloudWatch pengguna Amazon.

> **Note**
>
> CloudWatch pembacaan log dibatasi olehAWS. Jika Anda berencana untuk bergabung dengan banyak artis, kami sarankan Anda menghubungi dukungan AWS pelanggan dan meminta kenaikan `GetLogEvents` kuota. CloudWatch Selain itu, kami sarankan Anda menutup portal tailing log saat Anda tidak men-debug.

Untuk informasi selengkapnya, lihat Kuota CloudWatch log di panduan CloudWatch pengguna Amazon.

CloudWatch metrik

Deadline Cloud mengirimkan metrik ke Amazon. CloudWatch Anda dapat menggunakan, API Konsol Manajemen AWSAWS CLI, atau API untuk membuat daftar metrik yang dikirimkan oleh Deadline Cloud. CloudWatch Secara default, setiap titik data mencakup 1 menit yang mengikuti

waktu mulai aktivitas. Untuk informasi tentang cara melihat metrik yang tersedia menggunakan Konsol Manajemen AWS atau metrikAWS CLI, lihat Melihat metrik yang tersedia di CloudWatch Panduan Pengguna Amazon.

Metrik armada yang dikelola pelanggan

`AWS/DeadlineCloud`Namespace berisi metrik berikut untuk armada yang dikelola pelanggan Anda:

Metrik	Deskripsi	Unit
`RecommendedFleetSize`	Jumlah pekerja yang direkomendasikan Deadline Cloud yang Anda gunakan untuk memproses pekerjaan . Anda dapat menggunakan metrik ini untuk memperlua s atau mengontrak jumlah pekerja dari armada Anda.	Hitungan
`UnhealthyWorkerCount`	Jumlah pekerja yang ditugaskan untuk memproses pekerjaan yang tidak sehat.	Hitungan

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik armada yang dikelola pelanggan:

Dimensi	Deskripsi
FarmId	Dimensi ini menyaring data yang Anda minta ke peternakan yang ditentukan.
FleetId	Dimensi ini menyaring data yang Anda minta ke armada pekerja yang ditentukan.

Metrik perizinan

`AWS/DeadlineCloud`Namespace berisi metrik berikut untuk lisensi:

Metrik	Deskripsi	Unit
`LicensesInUse`	Jumlah sesi lisensi yang digunakan.	Hitungan

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik lisensi:

Dimensi	Deskripsi
FleetId	Gunakan dimensi ini untuk memfilter data ke armada yang dikelola layanan yang ditentuka n. Untuk armada yang dikelola pelanggan, gunakan dimensi sebagai LicenseEndpointId gantinya.
LicenseEndpointId	Gunakan dimensi ini untuk memfilter data ke titik akhir lisensi yang ditentukan.
Produk	Gunakan dimensi ini untuk memfilter data ke produk terukur yang ditentukan.

Metrik batas sumber daya

`AWS/DeadlineCloud`Namespace berisi metrik berikut untuk batas sumber daya:

Metrik	Deskripsi	Unit
`CurrentCount`	Jumlah sumber daya yang dimodelkan oleh batas ini digunakan.	Hitungan
`MaxCount`	Jumlah maksimum sumber daya yang dimodelkan oleh batas ini. Jika Anda menyetel `maxCount` nilai ke -1 menggunakan API, Deadline	Hitungan

Metrik	Deskripsi	Unit
	Cloud tidak memancarkan metrik. `MaxCount`	

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik batas bersamaan:

Dimensi	Deskripsi
FarmId	Dimensi ini menyaring data yang Anda minta ke peternakan yang ditentukan.
LimitId	Dimensi ini menyaring data yang Anda minta ke batas yang ditentukan.

Alarm-alarm yang direkomendasikan

Dengan CloudWatch, Anda dapat membuat alarm yang menonton metrik dan mengirimi Anda pemberitahuan atau melakukan tindakan lain saat ambang batas dilanggar. Untuk informasi selengkapnya tentang mengonfigurasi CloudWatch alarm, lihat CloudWatch Panduan Pengguna Amazon.

Kami menyarankan Anda menyetel alarm untuk metrik Deadline Cloud berikut:

LicensesInUse

Dimensi: FleetId, LicenseEndpointId

Deskripsi alarm: Alarm ini mendeteksi saat sesi lisensi aktif untuk armada yang dikelola layanan atau titik akhir lisensi mendekati kuota akun Anda. Jika ini terjadi, Anda dapat menaikkan kuota akun untuk sesi lisensi. Lihat kuota Anda saat ini dan peningkatan permintaan menggunakan Service Quotas. Untuk mempelajari selengkapnya, lihat Panduan Pengguna Service Quotas.

Maksud: Mencegah kegagalan checkout lisensi dengan memantau penggunaan sebelum mencapai batas kuota.

Statistik: Maksimum

Ambang batas yang disarankan: 90% dari kuota sesi lisensi Anda

Pembenaran ambang batas: Tetapkan ambang batas ke persentase kuota Anda, sehingga Anda dapat mengambil tindakan sebelum mencapai batas.

Periode: 1 menit

Titik data untuk alarm: 1

Periode evaluasi: 1

Operator Perbandingan: `GREATER_THAN_THRESHOLD`

Sumber daya tambahan

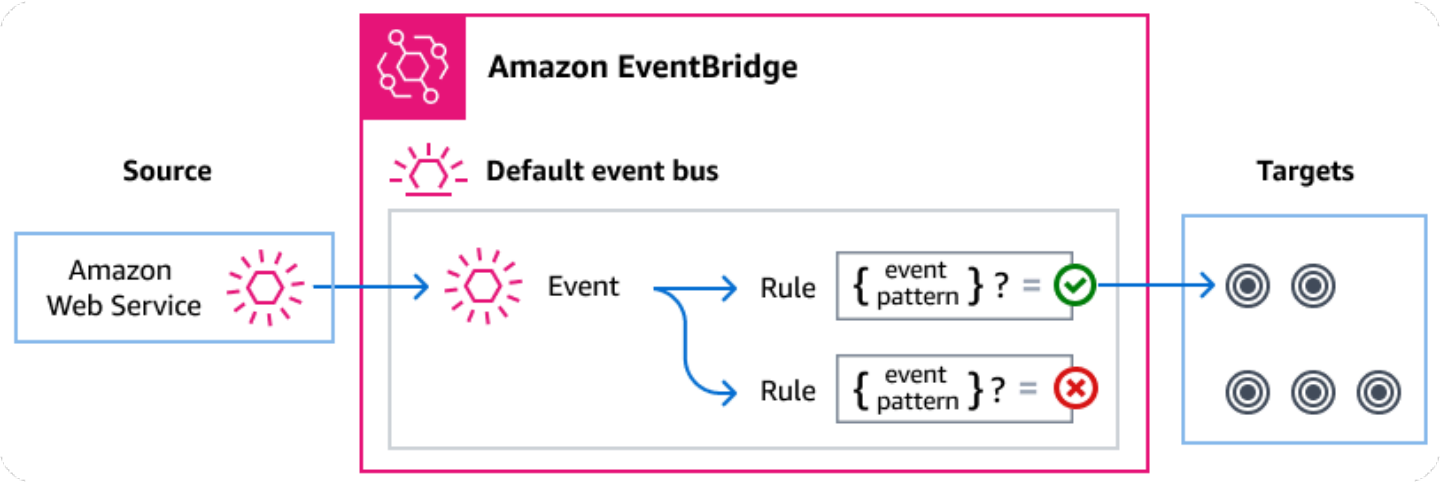
- Panduan CloudWatch Pengguna Amazon
- Panduan Pengguna Service Quotas

Mengelola peristiwa Deadline Cloud menggunakan Amazon EventBridge

Amazon EventBridge adalah layanan tanpa server yang menggunakan peristiwa untuk menghubungkan komponen aplikasi bersama-sama, sehingga memudahkan Anda untuk membangun aplikasi berbasis peristiwa yang dapat diskalakan. Arsitektur berbasis peristiwa adalah gaya membangun sistem perangkat lunak yang digabungkan secara longgar yang bekerja sama dengan memancarkan dan menanggapi peristiwa. Peristiwa mewakili perubahan dalam sumber daya atau lingkungan.

Begini cara kerjanya:

Seperti banyak AWS layanan, Deadline Cloud menghasilkan dan mengirim acara ke bus acara EventBridge default. (Bus acara default secara otomatis disediakan di setiap AWS akun.) Bus acara adalah router yang menerima acara dan mengirimkannya ke nol atau lebih tujuan, atau target. Aturan yang Anda tentukan untuk bus acara mengevaluasi peristiwa saat mereka tiba. Setiap aturan memeriksa apakah suatu peristiwa cocok dengan pola acara aturan. Jika acara tidak cocok, bus acara mengirimkan acara ke target yang ditentukan.



Topik

- Acara Batas Waktu Cloud
- Menyampaikan event Deadline Cloud menggunakan aturan EventBridge
- Referensi detail acara Batas waktu Cloud

Acara Batas Waktu Cloud

Deadline Cloud mengirimkan peristiwa berikut ke bus EventBridge acara default secara otomatis. Peristiwa yang cocok dengan pola acara aturan dikirim ke target yang ditentukan berdasarkan upaya terbaik. Acara mungkin dikirim keluar dari pesanan.

Untuk informasi selengkapnya, lihat EventBridge peristiwa di Panduan Amazon EventBridge Pengguna.

Jenis detail acara	Deskripsi
Ambang Batas Anggaran Tercapai	Dikirim ketika antrian mencapai persentase dari anggaran yang ditetapkan.
Perubahan Status Siklus Hidup Job	Dikirim ketika ada perubahan pada status siklus hidup suatu pekerjaan.
Perubahan Status Job Run	Dikirim ketika status keseluruhan tugas dalam pekerjaan berubah.

Jenis detail acara	Deskripsi
Langkah Perubahan Status Siklus Hidup	Dikirim ketika ada perubahan status siklus hidup dari langkah dalam pekerjaan.
Langkah Jalankan Perubahan Status	Dikirim ketika status keseluruhan tugas dalam satu langkah berubah.
Perubahan Status Jalankan Tugas	Dikirim saat status tugas berubah.

Menyampaikan event Deadline Cloud menggunakan aturan EventBridge

Agar bus acara EventBridge default mengirim peristiwa Deadline Cloud ke target, Anda harus membuat aturan. Setiap aturan berisi pola acara, yang EventBridge cocok dengan setiap acara yang diterima di bus acara. Jika data peristiwa cocok dengan pola peristiwa yang ditentukan, EventBridge mengirimkan peristiwa itu ke target aturan.

Untuk petunjuk komprehensif tentang cara membuat aturan bus acara, lihat Membuat aturan yang bereaksi terhadap peristiwa di Panduan EventBridge Pengguna.

Membuat pola acara yang cocok dengan peristiwa Deadline Cloud

Setiap pola acara adalah objek JSON yang berisi:

- `source`Atribut yang mengidentifikasi layanan yang mengirim acara. Untuk acara Deadline Cloud, sumbernya adalah`aws.deadline`.
- (Opsional): `detail-type` Atribut yang berisi array jenis acara yang cocok.
- (Opsional): `detail` Atribut yang berisi data acara lain yang cocok.

Misalnya, pola peristiwa berikut cocok dengan semua peristiwa Perubahan Rekomendasi Ukuran Armada untuk yang ditentukan `farmId` untuk Deadline Cloud:

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
```

```
  }
}
```

Untuk informasi selengkapnya tentang penulisan pola acara, lihat Pola acara di Panduan EventBridge Pengguna.

Referensi detail acara Batas waktu Cloud

Semua peristiwa dari AWS layanan memiliki seperangkat bidang umum yang berisi metadata tentang acara tersebut, seperti AWS layanan yang merupakan sumber acara, waktu acara dibuat, akun dan wilayah tempat acara berlangsung, dan lainnya. Untuk definisi bidang umum ini, lihat Referensi struktur acara di Panduan Amazon EventBridge Pengguna.

Selain itu, setiap acara memiliki `detail` bidang yang berisi data khusus untuk peristiwa tertentu. Referensi di bawah ini mendefinisikan bidang detail untuk berbagai peristiwa Deadline Cloud.

Saat menggunakan EventBridge untuk memilih dan mengelola peristiwa Deadline Cloud, penting untuk mengingat hal berikut:

- `source`Bidang untuk semua acara dari Deadline Cloud diatur ke`aws.deadline`.
- `detail-type`Bidang menentukan jenis acara.

  Misalnya, `Fleet Size Recommendation Change`.
- `detail`Bidang berisi data yang spesifik untuk peristiwa tertentu.

Untuk informasi tentang membuat pola peristiwa yang memungkinkan aturan agar sesuai dengan peristiwa Deadline Cloud, lihat Pola acara di Amazon EventBridge Panduan Pengguna.

Untuk informasi selengkapnya tentang peristiwa dan cara EventBridge memprosesnya, lihat Amazon EventBridge peristiwa di Panduan Amazon EventBridge Pengguna.

Topik

- Ambang Batas Anggaran Mencapai acara
- Acara Perubahan Rekomendasi Ukuran Armada
- Acara Perubahan Status Siklus Hidup Job
- Acara Perubahan Status Job Run
- Langkah Siklus Hidup Status Ubah acara

- Langkah Jalankan Status Ubah acara
- Acara Ubah Status Jalankan Tugas

Ambang Batas Anggaran Mencapai acara

Anda dapat menggunakan acara Budget Threshold Received untuk memantau persentase anggaran yang telah digunakan. Deadline Cloud mengirimkan peristiwa ketika persentase yang digunakan melewati ambang batas berikut:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

Frekuensi Deadline Cloud mengirimkan peristiwa Budget Threshold Received meningkat saat anggaran mendekati batasnya. Frekuensi ini memungkinkan Anda untuk memantau anggaran dengan cermat saat mendekati batasnya dan mengambil tindakan untuk mencegah pengeluaran berlebihan. Anda juga dapat menetapkan ambang anggaran Anda sendiri. Deadline Cloud mengirimkan peristiwa saat penggunaan melewati ambang batas kustom Anda.

Jika Anda mengubah jumlah anggaran, waktu berikutnya Deadline Cloud mengirimkan acara Ambang Batas Anggaran Tercapai, hal itu didasarkan pada persentase anggaran saat ini yang telah digunakan. Misalnya, jika Anda menambahkan $50 ke anggaran $100 yang telah mencapai batasnya, acara Ambang Batas Anggaran berikutnya menunjukkan bahwa anggaran berada pada 75 persen.

Di bawah ini adalah bidang detail untuk `Budget Threshold Reached` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Budget Threshold Reached`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`budgetId`

Pengidentifikasi anggaran yang telah mencapai ambang batas.

`thresholdInPercent`

Persentase anggaran yang telah digunakan.

Acara Perubahan Rekomendasi Ukuran Armada

Saat mengonfigurasi armada untuk menggunakan penskalaan otomatis berbasis peristiwa, Deadline Cloud mengirimkan peristiwa yang dapat Anda gunakan untuk mengelola armada Anda. Masing-masing acara ini berisi informasi tentang ukuran saat ini dan ukuran armada yang diminta. Untuk contoh penggunaan EventBridge peristiwa dan contoh fungsi Lambda untuk menangani acara, lihat Skala otomatis armada Amazon EC2 Anda dengan fitur rekomendasi skala Deadline Cloud.

Acara perubahan rekomendasi ukuran armada dikirim ketika hal berikut terjadi:

- Ketika ukuran armada yang direkomendasikan berubah dan oldFleetSize berbeda dari newFleetSize.
- Ketika layanan mendeteksi bahwa ukuran armada sebenarnya tidak sesuai dengan ukuran armada yang direkomendasikan. Anda bisa mendapatkan ukuran armada sebenarnya dari WorkerCount dalam respons GetFleet operasi. Hal ini dapat terjadi ketika instans Amazon EC2 aktif gagal mendaftar sebagai pekerja Deadline Cloud.

Di bawah ini adalah bidang detail untuk `Fleet Size Recommendation Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Fleet Size Recommendation Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`fleetId`

Pengidentifikasi armada yang membutuhkan perubahan ukuran.

`oldFleetSize`

Ukuran armada saat ini.

`newFleetSize`

Ukuran baru yang direkomendasikan untuk armada.

Acara Perubahan Status Siklus Hidup Job

Saat Anda membuat atau memperbarui pekerjaan, Deadline Cloud menetapkan status siklus hidup untuk menampilkan status tindakan yang dimulai pengguna terbaru.

Peristiwa perubahan status siklus hidup pekerjaan dikirim untuk setiap perubahan status siklus hidup, termasuk saat pekerjaan dibuat.

Di bawah ini adalah bidang detail untuk `Job Lifecycle Status Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Job Lifecycle Status Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`previousLifecycleStatus`

Siklus hidup menyatakan bahwa pekerjaan akan pergi. Bidang ini tidak termasuk saat Anda pertama kali mengirimkan pekerjaan.

`lifecycleStatus`

Siklus hidup menyatakan bahwa pekerjaan sedang masuk.

Acara Perubahan Status Job Run

Pekerjaan terdiri dari banyak tugas. Setiap tugas memiliki status. Status semua tugas digabungkan untuk memberikan status keseluruhan untuk suatu pekerjaan. Untuk informasi selengkapnya, lihat Status pekerjaan di Deadline Cloud di Panduan Pengguna Cloud AWS Deadline.

Peristiwa perubahan status job run dikirim saat:

- `taskRunStatus`Bidang gabungan berubah.
- Pekerjaan tersebut diminta kembali, kecuali pekerjaan tersebut dalam keadaan READY.

Peristiwa perubahan status job run TIDAK dikirim saat:

- Pekerjaan pertama kali dibuat. Untuk memantau pembuatan lowongan kerja, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.
- `taskRunStatusCounts`Bidang pekerjaan berubah tetapi status tugas tugas kerja gabungan tidak berubah.

Di bawah ini adalah bidang detail untuk `Job Run Status Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Job Run Status Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`previousTaskRunStatus`

Task run menyatakan bahwa pekerjaan akan pergi.

`taskRunStatus`

Task run menyatakan bahwa pekerjaan sedang masuk.

`taskRunStatusCounts`

Jumlah tugas pekerjaan di setiap negara bagian.

Langkah Siklus Hidup Status Ubah acara

Saat Anda membuat atau memperbarui peristiwa, Deadline Cloud menetapkan status siklus hidup pekerjaan untuk menjelaskan status tindakan yang dimulai pengguna terbaru.

Peristiwa perubahan status siklus hidup langkah dikirim saat:

- Pembaruan langkah dimulai (UPDATE_IN_PROGRESS).
- Pembaruan langkah berhasil diselesaikan (UPDATE_SUCCEEDED).
- Pembaruan langkah gagal (UPDATE_FAILED).

Peristiwa tidak dikirim saat langkah pertama kali dibuat. Untuk memantau pembuatan langkah, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.

Di bawah ini adalah bidang detail untuk `Step Lifecycle Status Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Step Lifecycle Status Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`stepId`

Pengidentifikasi langkah pekerjaan saat ini.

`previousLifecycleStatus`

Siklus hidup menyatakan bahwa langkah akan pergi.

`lifecycleStatus`

Siklus hidup menyatakan bahwa langkah tersebut masuk.

Langkah Jalankan Status Ubah acara

Setiap langkah dalam suatu pekerjaan terdiri dari banyak tugas. Setiap tugas memiliki status. Status tugas digabungkan untuk memberikan status keseluruhan untuk langkah dan pekerjaan.

Peristiwa perubahan status step run dikirim saat:

- `taskRunStatus`Perubahan gabungan.
- Langkah tersebut diminta ulang, kecuali langkah itu dalam keadaan READY.

Sebuah acara tidak dikirim ketika:

- Langkah ini pertama kali dibuat. Untuk memantau pembuatan langkah, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.
- Langkah `taskRunStatusCounts` berubah tetapi status run tugas langkah gabungan tidak berubah.

Di bawah ini adalah bidang detail untuk `Step Run Status Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Step Run Status Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`stepId`

Pengidentifikasi langkah pekerjaan saat ini.

`previousTaskRunStatus`

Lari menyatakan bahwa langkahnya akan pergi.

`taskRunStatus`

Jalankan menyatakan bahwa langkah sedang masuk.

`taskRunStatusCounts`

Jumlah tugas langkah di setiap negara bagian.

Acara Ubah Status Jalankan Tugas

`runStatus`Bidang diperbarui saat tugas berjalan. Sebuah acara dikirim ketika:

- Status run tugas berubah.
- Tugas tersebut diminta kembali, kecuali tugas dalam keadaan READY.

Sebuah acara tidak dikirim ketika:

- Tugas pertama kali dibuat. Untuk memantau pembuatan tugas, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.

Di bawah ini adalah bidang detail untuk `Task Run Status Change` acara tersebut.

`detail-type`Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat Referensi struktur acara di Amazon EventBridge Panduan Pengguna.

```
{
    "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`

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah`Fleet Size Recommendation Change`.

`source`

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah`aws.deadline`.

`detail`

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi peternakan yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

stepId

Pengidentifikasi langkah pekerjaan saat ini.

`taskId`

Pengidentifikasi tugas yang sedang berjalan.

`previousRunStatus`

Jalankan menyatakan bahwa tugas akan pergi.

`runStatus`

Status run yang dimasukkan tugas.

Kueri statistik sesi data agregat menggunakan AWS CLI

Untuk melacak biaya, menganalisis penggunaan sumber daya, atau mengidentifikasi pengguna mana yang paling banyak menggunakan sumber daya, Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk menanyakan statistik sesi agregat untuk farm AWS Deadline Cloud (Deadline Cloud) Anda. API statistik sesi menyediakan data tentang biaya, runtime, dan penggunaan yang dapat Anda kelompokkan berdasarkan berbagai dimensi seperti antrian, armada, jenis instans, atau pengguna.

Kueri statistik sesi adalah proses asinkron. Pertama, Anda memulai permintaan agregasi, lalu Anda mengambil hasilnya menggunakan ID agregasi.

Memulai permintaan agregasi

Untuk memulai permintaan agregasi, jalankan `start-sessions-statistics-aggregation` perintah. Contoh berikut mengelompokkan statistik berdasarkan ID pengguna untuk antrian tertentu. Ganti *`placeholder`* teks dengan informasi Anda.

```
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
```

Anda dapat mengelompokkan statistik berdasarkan dimensi lain sepertiQUEUE_ID,FLEET_ID,JOB_ID,INSTANCE_TYPE, atauLICENSE_PRODUCT. Untuk informasi selengkapnya tentang semua parameter yang tersedia, lihat start-sessions-statistics-aggregationdi Referensi AWS CLI Perintah.

Respons berisi ID agregasi:

```
{
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Mengambil hasil

Jalankan `get-sessions-statistics-aggregation` perintah dengan ID agregasi untuk mengambil hasilnya. Ganti *`placeholder`* teks dengan informasi Anda.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

Contoh berikut menunjukkan respons saat Anda mengelompokkan statistik berdasarkan ID pengguna. `userId`Bidang berisi UUID yang harus Anda petakan ke nama pengguna untuk mengidentifikasi pengguna:

```
{
    "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"
}
```

Untuk menemukan nama pengguna yang terkait dengan a`userId`, lihatthe section called “Mengambil metadata pengguna menggunakan userID”.

Untuk informasi selengkapnya tentang API, lihat Referensi API Cloud Deadline.

Topik

- the section called “Mengambil metadata pengguna menggunakan userID”

Mengambil metadata dan atribut pengguna menggunakan userId di toko identitas

> **Note**
>
> Prosedur ini juga berlaku untuk `createdBy` bidang yang dikembalikan oleh SearchJobsAPI, yang menggunakan format ID pengguna yang sama.

`userId`Bidang dalam statistik sesi berisi salah satu nilai berikut:

- Peran AWS Identity and Access Management (IAM) ARN, misalnya:. `arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`
- ID pengguna IAM Identity Center (UUID), misalnya:. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Untuk peran IAM ARNs, nama pengguna terlihat di ARN itu sendiri. Untuk pengguna IAM Identity Center IDs, Anda dapat mencari nama pengguna menggunakan IAM Identity Center Identity Store API.

Untuk mengidentifikasi nama pengguna yang terkait dengan ID pengguna IAM Identity Center, gunakan prosedur berikut. Sebelum memulai, dapatkan ID Toko Identitas dari pengaturan Pusat Identitas IAM Anda. Untuk informasi selengkapnya, lihat the section called “Menemukan ID Toko Identitas Anda”.

Untuk memetakan ID pengguna

1. Jalankan perintah berikut, ganti *`IdentityStoreId`* dengan ID Toko Identitas Anda dan *`userUUID`* dengan respons statistik `userId` dari sesi:

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Tinjau tanggapan, yang mencakup nama pengguna:

```
{
    "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"
}
```

Menemukan ID Toko Identitas Anda

Untuk memetakan pengguna IDs ke nama pengguna, Anda memerlukan ID Toko Identitas. Anda dapat menemukan ID Toko Identitas menggunakan konsol IAM Identity Center atau. AWS CLI

Konsol

Untuk menemukan ID Toko Identitas Anda menggunakan konsol, gunakan prosedur berikut.

1. Masuk ke Konsol AWS Manajemen dan buka konsol Pusat Identitas IAM.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Salin nilai ID Toko Identitas Pusat Identitas IAM. Formatnya adalah `d-xxxxxxxxxx`.

AWS CLI

Jalankan perintah berikut, ganti *`region-name`* dengan Region tempat instans IAM Identity Center Anda dikonfigurasi:

```
aws sso-admin list-instances --region region-name
```

Tanggapan tersebut meliputi`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"
        }
    ]
}
```

Memverifikasi pemetaan pengguna

Setelah memetakan ID pengguna ke nama pengguna, Anda dapat memverifikasi di konsol Pusat Identitas IAM bahwa ID pengguna cocok dengan pengguna yang diharapkan. Untuk memverifikasi pemetaan pengguna, gunakan prosedur berikut.

1. Masuk ke Konsol AWS Manajemen dan buka konsol Pusat Identitas IAM.
2. Di panel navigasi, pilih Users (Pengguna).
3. Pilih nama pengguna dari AWS CLI respons.
4. Di bagian Informasi umum, verifikasi bahwa ID Pengguna cocok dengan statistik sesi `userId` dari Anda.

Sumber daya tambahan

- Panduan Pengguna Pusat Identitas IAM
- Referensi API Toko Identitas Pusat Identitas IAM

Keamanan di Deadline Cloud

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Third-party auditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program Kepatuhan Program AWS Kepatuhan Program AWS . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Deadline Cloud, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Deadline Cloud. Topik berikut menunjukkan cara mengonfigurasi Deadline Cloud untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan Deadline Cloud sumber daya Anda.

Topik

- Praktik terbaik keamanan untuk Deadline Cloud

Perlindungan data di Deadline Cloud

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS Deadline Cloud. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data AWS . Untuk informasi tentang perlindungan data di Eropa, lihat Pusat Peraturan Perlindungan Data Umum (GDPR).

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Deadline Cloud atau lainnya Layanan AWS

menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Data yang dimasukkan ke dalam bidang nama dalam templat Deadline Cloud pekerjaan juga dapat dimasukkan dalam log penagihan atau diagnostik dan tidak boleh berisi informasi rahasia atau sensitif.

Topik

Enkripsi saat diam

AWS Deadline Cloud melindungi data sensitif dengan mengenkripsinya saat istirahat menggunakan kunci enkripsi yang disimpan di AWS Key Management Service ()AWS KMS. Enkripsi saat istirahat tersedia di semua Wilayah AWS tempat Deadline Cloud yang tersedia.

Mengenkripsi data berarti data sensitif yang disimpan pada disk tidak dapat dibaca oleh pengguna atau aplikasi tanpa kunci yang valid. Hanya pihak dengan kunci terkelola yang valid yang dapat mendekripsi data.

Deadline Cloud menghapus volume Amazon Elastic Block Store saat instance pekerja armada yang dikelola layanan dihentikan.

Untuk informasi tentang cara Deadline Cloud penggunaan AWS KMS untuk mengenkripsi data saat istirahat, lihat. Manajemen kunci

Enkripsi saat bergerak

Untuk data dalam perjalanan, AWS Deadline Cloud gunakan Transport Layer Security (TLS) 1.2 atau 1.3 untuk mengenkripsi data yang dikirim antara layanan dan pekerja. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3. Selain itu, jika Anda menggunakan virtual private cloud (VPC), Anda

dapat menggunakannya AWS PrivateLink untuk membuat koneksi pribadi antara VPC dan VPC Anda. Deadline Cloud

Manajemen kunci

Saat membuat peternakan baru, Anda dapat memilih salah satu kunci berikut untuk mengenkripsi data pertanian Anda:

- AWS kunci KMS yang dimiliki — Jenis enkripsi default jika Anda tidak menentukan kunci saat membuat peternakan. Kunci KMS dimiliki oleh AWS Deadline Cloud. Anda tidak dapat melihat, mengelola, atau menggunakan kunci AWS yang dimiliki. Namun, Anda tidak perlu mengambil tindakan apa pun untuk melindungi kunci yang mengenkripsi data Anda. Untuk informasi selengkapnya, lihat kunci yang AWS dimiliki di panduan AWS Key Management Service pengembang.
- Kunci KMS yang dikelola pelanggan — Anda menentukan kunci yang dikelola pelanggan saat membuat peternakan. Semua konten di dalam peternakan dienkripsi dengan kunci KMS. Kunci disimpan di akun Anda dan dibuat, dimiliki, dan dikelola oleh Anda dan AWS KMS dikenakan biaya. Anda memiliki kontrol penuh atas tombol KMS. Anda dapat melakukan tugas-tugas seperti:
  - Menetapkan dan memelihara kebijakan utama
  - Menetapkan dan memelihara kebijakan dan hibah IAM
  - Mengaktifkan dan menonaktifkan kebijakan utama
  - Menambahkan tanda
  - Membuat alias kunci

  Anda tidak dapat memutar kunci milik pelanggan secara manual yang digunakan dengan Deadline Cloud peternakan. Rotasi otomatis tombol didukung.

  Untuk informasi selengkapnya, lihat Kunci milik pelanggan di Panduan AWS Key Management Service Pengembang.

  Untuk membuat kunci terkelola pelanggan, ikuti langkah-langkah untuk Membuat kunci terkelola pelanggan simetris di Panduan AWS Key Management Service Pengembang.

Bagaimana Deadline Cloud menggunakan AWS KMS hadiah

Deadline Cloud membutuhkan hibah untuk menggunakan kunci yang dikelola pelanggan Anda. Saat Anda membuat peternakan yang dienkripsi dengan kunci yang dikelola pelanggan, Deadline Cloud

buat hibah atas nama Anda dengan mengirimkan `CreateGrant` permintaan untuk mendapatkan akses AWS KMS ke kunci KMS yang Anda tentukan.

Deadline Cloud menggunakan beberapa hibah. Setiap hibah digunakan oleh bagian yang berbeda Deadline Cloud yang perlu mengenkripsi atau mendekripsi data Anda. Deadline Cloud juga menggunakan hibah untuk memungkinkan akses ke AWS layanan lain yang digunakan untuk menyimpan data atas nama Anda, seperti Amazon Simple Storage Service, Amazon Elastic Block Store, atau OpenSearch.

Hibah yang memungkinkan Deadline Cloud untuk mengelola mesin dalam armada yang dikelola layanan mencakup nomor Deadline Cloud akun dan peran dalam `GranteePrincipal` alih-alih prinsip layanan. Meskipun tidak khas, ini diperlukan untuk mengenkripsi volume Amazon EBS untuk pekerja dalam armada yang dikelola layanan menggunakan kunci KMS terkelola pelanggan yang ditentukan untuk pertanian.

Kebijakan kunci yang dikelola pelanggan

Kebijakan utama mengontrol akses ke kunci yang dikelola pelanggan Anda. Setiap kunci harus memiliki persis satu kebijakan kunci yang berisi pernyataan yang menentukan siapa yang dapat menggunakan kunci dan bagaimana mereka dapat menggunakannya. Saat membuat kunci terkelola pelanggan, Anda dapat menentukan kebijakan kunci. Untuk informasi selengkapnya, lihat Mengelola akses ke kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

Kebijakan IAM minimal untuk CreateFarm

Untuk menggunakan kunci terkelola pelanggan Anda untuk membuat farm menggunakan konsol atau operasi `CreateFarm` API, operasi AWS KMS API berikut harus diizinkan:

- `kms:CreateGrant`— Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses konsol ke AWS KMS kunci tertentu. Untuk informasi selengkapnya, lihat Menggunakan hibah di panduan AWS Key Management Service pengembang.
- `kms:Decrypt`— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- `kms:DescribeKey`— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.
- `kms:GenerateDataKey`— Memungkinkan Deadline Cloud untuk mengenkripsi data menggunakan kunci data yang unik.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi. `CreateFarm`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Kebijakan IAM minimal untuk operasi hanya-baca

Untuk menggunakan kunci yang dikelola pelanggan Anda untuk Deadline Cloud operasi hanya-baca, seperti mendapatkan informasi tentang peternakan, antrian, dan armada. Operasi AWS KMS API berikut harus diizinkan:

- `kms:Decrypt`— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- `kms:DescribeKey`— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi hanya-baca.

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"
                }
            }
        }
    ]
}
```

Kebijakan IAM minimal untuk operasi baca-tulis

Untuk menggunakan kunci terkelola pelanggan Anda untuk Deadline Cloud operasi baca-tulis, seperti membuat dan memperbarui peternakan, antrian, dan armada. Operasi AWS KMS API berikut harus diizinkan:

- `kms:Decrypt`— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- `kms:DescribeKey`— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.
- `kms:GenerateDataKey`— Memungkinkan Deadline Cloud untuk mengenkripsi data menggunakan kunci data yang unik.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi. `CreateFarm`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Memantau kunci enkripsi Anda

Saat Anda menggunakan kunci terkelola AWS KMS pelanggan dengan Deadline Cloud peternakan, Anda dapat menggunakan AWS CloudTrailatau Amazon CloudWatch Logs untuk melacak permintaan yang Deadline Cloud dikirim AWS KMS.

CloudTrail acara untuk hibah

Contoh CloudTrail peristiwa berikut terjadi ketika hibah dibuat, biasanya ketika Anda memanggil`CreateFarm`,`CreateMonitor`, atau `CreateFleet` operasi.

```
{
    "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 acara untuk dekripsi

Contoh CloudTrail peristiwa berikut terjadi ketika mendekripsi nilai menggunakan kunci KMS yang dikelola pelanggan.

```
{
    "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 acara untuk enkripsi

Contoh CloudTrail peristiwa berikut terjadi ketika mengenkripsi nilai menggunakan kunci KMS yang dikelola pelanggan.


```
{
    "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"
}
```

Menghapus kunci KMS yang dikelola pelanggan

Menghapus kunci KMS yang dikelola pelanggan di AWS Key Management Service (AWS KMS) bersifat merusak dan berpotensi berbahaya. Ini secara permanen menghapus materi kunci dan semua metadata yang terkait dengan kunci. Setelah kunci KMS yang dikelola pelanggan dihapus, Anda tidak dapat lagi mendekripsi data yang dienkripsi oleh kunci itu. Menghapus kunci berarti bahwa data menjadi tidak dapat dipulihkan.

Inilah sebabnya mengapa AWS KMS memberi pelanggan masa tunggu hingga 30 hari sebelum menghapus kunci KMS. Masa tunggu default adalah 30 hari.

Tentang masa tunggu

Karena menghapus kunci KMS yang dikelola pelanggan merusak dan berpotensi berbahaya, kami mengharuskan Anda menetapkan masa tunggu 7-30 hari. Masa tunggu default adalah 30 hari.

Namun, masa tunggu sebenarnya mungkin hingga 24 jam lebih lama dari periode yang Anda jadwalkan. Untuk mendapatkan tanggal dan waktu aktual ketika kunci akan dihapus, gunakan DescribeKeyoperasi. Anda juga dapat melihat tanggal penghapusan kunci yang dijadwalkan di AWS KMS konsol pada halaman detail kunci, di bagian Konfigurasi umum. Perhatikan zona waktu.

Selama masa tunggu, status dan status kunci yang dikelola pelanggan adalah Penghapusan tertunda.

- Kunci KMS yang dikelola pelanggan yang tertunda penghapusan tidak dapat digunakan dalam operasi kriptografi apa pun.
- AWS KMS tidak memutar kunci dukungan kunci KMS yang dikelola pelanggan yang sedang menunggu penghapusan.

Untuk informasi selengkapnya tentang menghapus kunci KMS yang dikelola pelanggan, lihat Menghapus kunci master pelanggan di Panduan Pengembang AWS Key Management Service .

Inter-network privasi lalu lintas

AWS Deadline Cloud mendukung Amazon Virtual Private Cloud (Amazon VPC) untuk mengamankan koneksi. Amazon VPC menyediakan fitur yang dapat Anda gunakan untuk meningkatkan dan memantau keamanan virtual private cloud (VPC) Anda.

Anda dapat menyiapkan armada yang dikelola pelanggan (CMF) dengan instans Amazon Elastic Compute Cloud (Amazon EC2) yang berjalan di dalam VPC. Dengan menerapkan titik akhir VPC Amazon untuk AWS PrivateLink digunakan, lalu lintas antar pekerja di CMF Anda dan Deadline Cloud titik akhir tetap berada dalam VPC Anda. Selanjutnya, Anda dapat mengonfigurasi VPC Anda untuk membatasi akses internet ke instans Anda.

Dalam armada yang dikelola layanan, pekerja tidak dapat dijangkau dari internet, tetapi mereka memiliki akses internet dan terhubung ke layanan melalui internet. Deadline Cloud Setiap armada yang dikelola layanan berjalan dalam jaringan terisolasinya sendiri, dan instance pekerja tetap didedikasikan untuk pelanggan individu.

Memilih keluar

AWS Deadline Cloud mengumpulkan informasi operasional tertentu untuk membantu kami mengembangkan dan meningkatkan Deadline Cloud. Data yang dikumpulkan mencakup hal-hal seperti ID AWS akun dan ID pengguna Anda, sehingga kami dapat mengidentifikasi Anda dengan benar jika Anda memiliki masalah dengan Deadline Cloud. Kami juga mengumpulkan informasi Deadline Cloud spesifik, seperti ID Sumber Daya (FarmId atau QueueID bila berlaku), nama produk (misalnya,, JobAttachments WorkerAgent, dan lainnya) dan versi produk.

Anda dapat memilih untuk memilih keluar dari pengumpulan data ini menggunakan konfigurasi aplikasi. Setiap komputer yang berinteraksi dengan Deadline Cloud, baik workstation klien dan pekerja armada, perlu memilih keluar secara terpisah.

Deadline Cloud monitor - desktop

Deadline Cloud monitor - desktop mengumpulkan informasi operasional, seperti ketika crash terjadi dan ketika aplikasi dibuka, untuk membantu kami mengetahui kapan Anda mengalami masalah dengan aplikasi. Untuk memilih keluar dari pengumpulan informasi operasional ini, buka halaman pengaturan dan hapus Aktifkan pengumpulan data untuk mengukur kinerja Deadline Cloud Monitor.

Setelah Anda memilih keluar, monitor desktop tidak lagi mengirimkan data operasional. Setiap data yang dikumpulkan sebelumnya disimpan dan masih dapat digunakan untuk meningkatkan layanan. Untuk informasi selengkapnya, lihat FAQ Privasi Data.

AWS Deadline Cloud CLI dan Alat

AWS Deadline Cloud CLI, pengirim, dan agen pekerja semuanya mengumpulkan informasi operasional seperti kapan crash terjadi dan kapan pekerjaan dikirimkan untuk membantu kami mengetahui kapan Anda mengalami masalah dengan aplikasi ini. Untuk memilih keluar dari pengumpulan informasi operasional ini, gunakan salah satu metode berikut:

- Di terminal, masukkan**`deadline config set telemetry.opt_out true`**.

  Ini akan memilih keluar dari CLI, pengirim, dan agen pekerja saat berjalan sebagai pengguna saat ini.
- Saat menginstal agen Deadline Cloud pekerja, tambahkan argumen baris **`--telemetry-opt-out`** perintah. Misalnya, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.
- Sebelum menjalankan agen pekerja, CLI, atau submitter, tetapkan variabel lingkungan: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Setelah Anda memilih keluar, Deadline Cloud alat tidak lagi mengirim data operasional. Setiap data yang dikumpulkan sebelumnya disimpan dan masih dapat digunakan untuk meningkatkan layanan. Untuk informasi selengkapnya, lihat FAQ Privasi Data.

Identity and Access Management di Deadline Cloud

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya Deadline Cloud. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Pemecahan masalah AWS Batas waktu Identitas dan akses Cloud)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Deadline Cloud bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Identity-based contoh kebijakan untuk Deadline Cloud)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensional dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional.

Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensi dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Identity-based kebijakan

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Identity-based kebijakan dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada beberapa identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Resource-based kebijakan

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Resource-based kebijakan adalah kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – Menentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCP) – Menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCP) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Deadline Cloud bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Deadline Cloud, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Deadline Cloud.

Fitur IAM yang dapat Anda gunakan AWS Batas Waktu Cloud

Fitur IAM	Dukungan Batas Waktu Cloud
Identity-based kebijakan	Ya
Resource-based kebijakan	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Ya
Service-linked peran	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS kerja Deadline Cloud dan lainnya dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Identity-based kebijakan untuk Deadline Cloud

Mendukung kebijakan berbasis identitas: Ya

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Identity-based contoh kebijakan untuk Deadline Cloud

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. Identity-based contoh kebijakan untuk Deadline Cloud

Resource-based kebijakan dalam Deadline Cloud

Mendukung kebijakan berbasis sumber daya: Tidak

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh principal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai principal dalam kebijakan berbasis sumber daya. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Tindakan kebijakan untuk Deadline Cloud

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan Cloud Deadline, lihat Tindakan yang ditentukan oleh AWS Deadline Cloud di Referensi Otorisasi Layanan.

Tindakan kebijakan di Deadline Cloud menggunakan awalan berikut sebelum tindakan:

```
deadline
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. Identity-based contoh kebijakan untuk Deadline Cloud

Sumber daya kebijakan untuk Deadline Cloud

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis resource Deadline Cloud dan ARNnya, lihat Sumber Daya yang ditentukan oleh AWS Deadline Cloud di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang

dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh AWS Deadline Cloud.

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. Identity-based contoh kebijakan untuk Deadline Cloud

Kunci kondisi kebijakan untuk Deadline Cloud

Mendukung kunci kondisi kebijakan khusus layanan: Yes

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Deadline Cloud, lihat Kunci kondisi untuk AWS Deadline Cloud di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS Deadline Cloud.

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. Identity-based contoh kebijakan untuk Deadline Cloud

ACL di Deadline Cloud

Mendukung ACL: Tidak

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Deadline Cloud

Mendukung ABAC (tanda dalam kebijakan): Ya

Attribute-based Access Control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut yang disebut tag. Anda dapat melampirkan tag ke entitas dan AWS sumber daya

IAM, lalu merancang kebijakan ABAC untuk mengizinkan operasi saat tag prinsipal cocok dengan tag pada sumber daya.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensil sementara dengan Deadline Cloud

Mendukung kredensial sementara: Ya

Kredensi sementara menyediakan akses jangka pendek ke AWS sumber daya dan secara otomatis dibuat saat Anda menggunakan federasi atau beralih peran. AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM dan Layanan AWS yang berfungsi dengan IAM dalam Panduan Pengguna IAM.

Teruskan sesi akses untuk Deadline Cloud

Mendukung sesi akses terusan (FAS): Ya

Sesi akses teruskan (FAS) menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses terusan.

Peran layanan untuk Deadline Cloud

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Deadline Cloud. Edit peran layanan hanya jika Deadline Cloud memberikan panduan untuk melakukannya.

Service-linked peran untuk Deadline Cloud

Mendukung peran terkait layanan: Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat mengambil peran untuk melakukan tindakan atas nama Anda. Service-linked peran muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau manajemen peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan layanan dalam tabel yang `Yes` menyertakan kolom Service-linked peran. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Identity-based contoh kebijakan untuk Deadline Cloud

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Deadline Cloud. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Deadline Cloud, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk AWS Deadline Cloud di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Cloud Deadline
- Kebijakan untuk mengakses konsol
- Kebijakan untuk mengirimkan pekerjaan ke antrian
- Kebijakan untuk mengizinkan pembuatan titik akhir lisensi

- Kebijakan untuk memungkinkan pemantauan antrian pertanian tertentu

Praktik terbaik kebijakan

Identity-based kebijakan menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Deadline Cloud di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Cloud Deadline

Untuk mengakses konsol AWS Deadline Cloud, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Cloud Deadline di Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Deadline Cloud, lampirkan juga Deadline Cloud *ConsoleAccess* atau kebijakan *ReadOnly* AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambah izin untuk pengguna dalam Panduan Pengguna IAM.

Kebijakan untuk mengakses konsol

Untuk memberikan akses ke semua fungsionalitas di konsol Deadline Cloud, lampirkan kebijakan identitas ini ke pengguna atau peran yang ingin Anda akses penuh.

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": ["*"]
      }]
}
```

Kebijakan untuk mengirimkan pekerjaan ke antrian

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin untuk mengirimkan pekerjaan ke antrian tertentu di peternakan tertentu.

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/*"
        }
    ]
}
```

Kebijakan untuk mengizinkan pembuatan titik akhir lisensi

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin yang diperlukan untuk membuat dan mengelola titik akhir lisensi. Gunakan kebijakan ini untuk membuat titik akhir lisensi untuk VPC yang terkait dengan farm Anda.

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/*"
        ]
    }]
}
```

Kebijakan untuk memungkinkan pemantauan antrian pertanian tertentu

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin untuk memantau pekerjaan dalam antrian tertentu untuk peternakan tertentu.

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 kebijakan terkelola untuk Deadline Cloud

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AWSDeadlineCloud-FleetWorker

Anda dapat melampirkan `AWSDeadlineCloud-FleetWorker` kebijakan ke identitas AWS Identity and Access Management (IAM) Anda.

Kebijakan ini memberi pekerja di armada ini izin yang diperlukan untuk terhubung dan menerima tugas dari layanan.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan kepala sekolah untuk mengelola pekerja dalam armada.

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-FleetWorkerdi panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-WorkerHost

Anda dapat melampirkan kebijakan `AWSDeadlineCloud-WorkerHost` ke identitas IAM Anda.

Kebijakan ini memberikan izin yang diperlukan untuk awalnya terhubung ke layanan. Ini dapat digunakan sebagai profil instans Amazon Elastic Compute Cloud (Amazon EC2).

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk membuat pekerja, mengambil peran armada untuk pekerja, dan menerapkan tag untuk pekerja

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-WorkerHostdi panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessFarms

Anda dapat melampirkan kebijakan `AWSDeadlineCloud-UserAccessFarms` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data pertanian berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis instans Amazon EC2.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.
- `kms`— Memungkinkan pengguna untuk mengonfigurasi AWS Key Management Service (AWS KMS) kunci yang dikelola pelanggan untuk instance AWS IAM Identity Center (IAM Identity Center) mereka.

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-UserAccessFarmsdi panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessFleets

Anda dapat melampirkan kebijakan `AWSDeadlineCloud-UserAccessFleets` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data armada berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis instans Amazon EC2.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-UserAccessFleetsdi panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessJobs

Anda dapat melampirkan kebijakan `AWSDeadlineCloud-UserAccessJobs` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data pekerjaan berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis instans Amazon EC2.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-UserAccessJobsdi panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessQueues

Anda dapat melampirkan kebijakan `AWSDeadlineCloud-UserAccessQueues` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data antrian berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis instans Amazon EC2.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat AWSDeadlineCloud-UserAccessQueuesdi panduan referensi Kebijakan Terkelola AWS.

Tenggat waktu pembaruan Cloud ke AWS kebijakan terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Deadline Cloud sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen Cloud Batas Waktu.

Ubah	Deskripsi	Date
AWSDeadlineCloud-UserAccessFarms— Ubah	Deadline Cloud menambahkan tindakan baru `kms:Decrypt` sehingga Anda dapat menggunakan kunci yang AWS KMS dikelola pelanggan dengan instans Pusat Identitas IAM Anda.	Desember 22, 2025
AWSDeadlineCloud-WorkerHost— Ubah	Deadline Cloud menambahkan tindakan baru `deadline:TagResource` dan `deadline:ListTagsForResource` memungkinkan Anda menambahkan dan melihat tag yang terkait dengan pekerja di armada Anda.	30 Mei 2025
AWSDeadlineCloud-UserAccessFarms— Ubah AWSDeadlineCloud-UserAccessJobs— Ubah AWSDeadlineCloud-UserAccessQueues— Ubah	Deadline Cloud menambahkan tindakan baru `deadline:GetJobTemplate` dan `deadline:ListJobParameterDefinitions` memungkinkan Anda mengirimkan kembali pekerjaan.	Oktober 7, 2024
Deadline Cloud mulai melacak perubahan	Deadline Cloud mulai melacak perubahan pada kebijakan AWS terkelolanya.	April 2, 2024

Peran layanan

Bagaimana Deadline Cloud menggunakan peran layanan IAM

Deadline Cloud secara otomatis mengasumsikan peran IAM dan memberikan kredensyal sementara kepada pekerja, pekerjaan, dan monitor Deadline Cloud. Pendekatan ini menghilangkan manajemen kredensi manual sambil menjaga keamanan melalui kontrol akses berbasis peran.

Saat membuat monitor, armada, dan antrian, Anda menentukan peran IAM yang diasumsikan Deadline Cloud atas nama Anda. Pekerja dan monitor Deadline Cloud kemudian menerima kredensi sementara dari peran ini untuk diakses. Layanan AWS

Peran armada

Konfigurasikan peran armada untuk memberi pekerja Deadline Cloud izin yang mereka perlukan untuk menerima pekerjaan dan melaporkan kemajuan pekerjaan itu.

Anda biasanya tidak perlu mengonfigurasi peran ini sendiri. Peran ini dapat dibuat untuk Anda di konsol Deadline Cloud untuk menyertakan izin yang diperlukan. Gunakan panduan berikut untuk memahami secara spesifik peran ini untuk pemecahan masalah.

Saat membuat atau memperbarui armada secara terprogram, tentukan peran armada ARN menggunakan operasi atau API. `CreateFleet` `UpdateFleet`

Apa peran armada

Peran armada memberi pekerja izin untuk:

- Menerima pekerjaan baru dan melaporkan kemajuan pekerjaan yang sedang berlangsung ke layanan Deadline Cloud
- Mengelola siklus hidup dan status pekerja
- Rekam peristiwa log ke Amazon CloudWatch Logs untuk log pekerja

Menyiapkan kebijakan kepercayaan peran armada

Peran armada Anda harus mempercayai layanan Deadline Cloud dan dicakup ke peternakan spesifik Anda.

Sebagai praktik terbaik, kebijakan kepercayaan harus mencakup kondisi keamanan untuk perlindungan Deputi Bingung. Untuk mempelajari lebih lanjut tentang perlindungan Deputi Bingung, lihat Deputi Bingung di Panduan Pengguna Cloud Tenggat Waktu.

- `aws:SourceAccount`memastikan hanya sumber daya dari yang sama yang Akun AWS dapat mengambil peran ini.
- `aws:SourceArn`membatasi asumsi peran ke pertanian Cloud Deadline tertentu.

```
{
  "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"
        }
      }
    }
  ]
}
```

Lampirkan izin peran Armada

Lampirkan kebijakan AWS terkelola berikut ke peran armada Anda:

AWSDeadlineCloud-FleetWorker

Kebijakan terkelola ini memberikan izin untuk:

- `deadline:AssumeFleetRoleForWorker`- Memungkinkan pekerja untuk menyegarkan kredensialnya.
- `deadline:UpdateWorker`- Memungkinkan pekerja untuk memperbarui status mereka (misalnya, untuk BERHENTI saat keluar).
- `deadline:UpdateWorkerSchedule`- Untuk mendapatkan pekerjaan dan melaporkan kemajuan.
- `deadline:BatchGetJobEntity`- Untuk mengambil informasi pekerjaan.

- `deadline:AssumeQueueRoleForWorker`- Untuk mengakses kredensyal peran antrian selama eksekusi pekerjaan.

Tambahkan izin KMS untuk peternakan terenkripsi

Jika peternakan Anda dibuat menggunakan kunci KMS, tambahkan izin ini ke peran armada Anda untuk memastikan pekerja dapat mengakses data terenkripsi di peternakan.

Izin KMS hanya diperlukan jika peternakan Anda memiliki kunci KMS terkait. `kms:ViaService`Kondisi harus menggunakan format`deadline.{region}.amazonaws.com`.

Saat membuat armada, grup CloudWatch log Log dibuat untuk armada tersebut. Izin pekerja digunakan oleh layanan Deadline Cloud untuk membuat aliran log khusus untuk pekerja tertentu. Setelah pekerja diatur dan dijalankan, pekerja akan menggunakan izin ini untuk mengirim peristiwa log langsung ke CloudWatch Log.

```
{
  "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"
        }
      }
    }
  ]
}
```

Memodifikasi peran armada

Izin untuk peran armada tidak dapat disesuaikan. Izin yang dijelaskan selalu diperlukan dan menambahkan izin tambahan tidak berpengaruh.

Customer-managed peran tuan rumah armada

Siapkan WorkerHost peran jika Anda menggunakan armada yang dikelola pelanggan di instans Amazon EC2 atau host lokal.

Apa WorkerHost perannya

WorkerHost Peran bootstrap pekerja pada host armada yang dikelola pelanggan. Ini memberikan izin minimal yang diperlukan untuk host untuk:

- Buat pekerja di Deadline Cloud
- Asumsikan peran armada untuk mengambil kredensi operasional
- Beri tag pekerja dengan tag armada (jika propagasi tag diaktifkan)

Siapkan izin WorkerHost peran

Lampirkan kebijakan AWS terkelola berikut ke WorkerHost peran Anda:

AWSDeadlineCloud-WorkerHost

Kebijakan terkelola ini memberikan izin untuk:

- `deadline:CreateWorker`- Memungkinkan tuan rumah untuk mendaftarkan pekerja baru.
- `deadline:AssumeFleetRoleForWorker`- Memungkinkan tuan rumah untuk mengambil peran armada.
- `deadline:TagResource`- Memungkinkan menandai pekerja selama pembuatan (jika diaktifkan).
- `deadline:ListTagsForResource`- Memungkinkan membaca tag armada untuk propagasi.

Memahami proses bootstrap

WorkerHost Peran ini hanya digunakan selama startup pekerja awal:

1. Agen pekerja memulai pada host menggunakan WorkerHost kredensil.
2. Ini memanggil `deadline:CreateWorker` untuk mendaftar dengan Deadline Cloud.
3. Kemudian memanggil `deadline:AssumeFleetRoleForWorker` untuk mengambil kredensi peran armada.
4. Mulai saat ini, pekerja hanya menggunakan kredensi peran armada untuk semua operasi.

WorkerHost Peran tidak digunakan setelah pekerja mulai berjalan. Kebijakan ini tidak diperlukan untuk Service-managed armada. Dalam Service-managed armada, bootstrap dilakukan secara otomatis.

Peran antrian

Peran antrian diasumsikan oleh pekerja saat memproses tugas. Peran ini memberikan izin yang diperlukan untuk menyelesaikan tugas.

Saat membuat atau memperbarui antrian secara terprogram, tentukan peran antrian ARN menggunakan operasi atau API. `CreateQueue` `UpdateQueue`

Menyiapkan kebijakan kepercayaan peran antrian

Peran antrian Anda harus mempercayai layanan Deadline Cloud.

Sebagai praktik terbaik, kebijakan kepercayaan harus mencakup kondisi keamanan untuk perlindungan Deputi Bingung. Untuk mempelajari lebih lanjut tentang perlindungan Deputi Bingung, lihat Deputi Bingung di Panduan Pengguna Cloud Tenggat Waktu.

- `aws:SourceAccount`memastikan hanya sumber daya dari yang sama yang Akun AWS dapat mengambil peran ini.
- `aws:SourceArn`membatasi asumsi peran ke pertanian Cloud Deadline tertentu.

```
{
  "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}"
        }
      }
    }
  ]
}
```

Memahami izin peran antrian

Peran antrian tidak menggunakan satu kebijakan terkelola. Sebagai gantinya, saat Anda mengonfigurasi antrian di konsol, Deadline Cloud membuat kebijakan khusus untuk antrian berdasarkan konfigurasi Anda.

Kebijakan yang dibuat secara otomatis ini menyediakan akses ke:

Lampiran Job

Akses baca dan tulis ke bucket Amazon S3 yang Anda tentukan untuk file input dan output pekerjaan:

```
{
  "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"
    }
  }
}
```

Log Job

Baca akses ke CloudWatch Log untuk pekerjaan dalam antrian ini. Setiap antrian memiliki grup lognya sendiri dan setiap sesi memiliki aliran lognya sendiri:

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Third-party perangkat lunak

Akses untuk mengunduh perangkat lunak pihak ketiga yang didukung oleh Deadline Cloud (seperti Maya, Blender, dan lainnya):

```
{
```

```
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Tambahkan izin untuk pekerjaan Anda

Tambahkan izin ke peran antrian Anda untuk pekerjaan Layanan AWS yang perlu diakses. Saat menulis skrip OpenJobDescription langkah, SDK AWS CLI dan akan secara otomatis menggunakan kredensil dari peran antrian Anda. Gunakan ini untuk mengakses layanan tambahan yang diperlukan untuk menyelesaikan pekerjaan Anda.

Contoh kasus penggunaan meliputi:

- untuk mengambil data kustom
- Izin SSM untuk terowongan ke server lisensi khusus
- CloudWatch untuk memancarkan metrik khusus
- Batas waktu izin Cloud untuk membuat pekerjaan baru untuk alur kerja dinamis

Bagaimana kredensi peran antrian digunakan

Deadline Cloud menyediakan kredensi peran antrian untuk:

- Pekerja selama eksekusi pekerjaan
- Pengguna melalui Deadline Cloud CLI dan monitor saat berinteraksi dengan lampiran pekerjaan dan log

Deadline Cloud membuat grup CloudWatch log Log terpisah untuk setiap antrian.
Pekerjaan menggunakan kredensyal peran antrian untuk menulis log ke grup log

antrean mereka. CLI dan monitor Deadline Cloud menggunakan peran antrian (`deadline:AssumeQueueRoleForRead`melalui) untuk membaca log pekerjaan dari grup log antrian. CLI dan monitor Deadline Cloud menggunakan peran antrian (`deadline:AssumeQueueRoleForUser`melalui) untuk mengunggah atau mengunduh data lampiran pekerjaan.

Memantau peran

Konfigurasikan peran monitor untuk memberi Deadline Cloud monitor web dan aplikasi desktop akses ke sumber daya Deadline Cloud Anda.

Saat membuat atau memperbarui monitor secara terprogram, tentukan peran monitor ARN menggunakan operasi atau API. `CreateMonitor` `UpdateMonitor`

Apa yang dilakukan peran monitor

Peran monitor memungkinkan monitor Deadline Cloud untuk memberi pengguna akhir akses ke:

- Fungsionalitas dasar yang diperlukan untuk Deadline Cloud Integrated Submitters, CLI dan monitor
- Fungsionalitas khusus untuk pengguna akhir

Siapkan kebijakan kepercayaan peran monitor

Peran monitor Anda harus mempercayai layanan Deadline Cloud.

Sebagai praktik terbaik, kebijakan kepercayaan harus mencakup kondisi keamanan untuk perlindungan Deputi Bingung. Untuk mempelajari lebih lanjut tentang perlindungan Deputi Bingung, lihat Deputi Bingung di Panduan Pengguna Cloud Tenggat Waktu.

`aws:SourceAccount`memastikan hanya sumber daya dari yang sama yang Akun AWS dapat mengambil peran ini.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

Lampirkan izin peran monitor

Lampirkan semua kebijakan AWS terkelola berikut ke peran monitor Anda untuk operasi dasar:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

Cara kerja peran monitor

Saat menggunakan monitor Deadline Cloud, pengguna layanan masuk menggunakan AWS IAM Identity Center (IAM Identity Center), dan peran monitor diasumsikan. Kredensi peran yang diasumsikan digunakan oleh aplikasi monitor untuk menampilkan UI monitor, termasuk daftar peternakan, armada, antrian, dan informasi lainnya.

Saat menggunakan aplikasi desktop monitor Deadline Cloud, kredensil ini juga tersedia di workstation menggunakan profil AWS kredensi bernama yang sesuai dengan nama profil yang diberikan oleh pengguna akhir. Pelajari lebih lanjut tentang profil bernama dalam panduan referensi AWS SDK dan Alat.

Profil bernama ini adalah bagaimana Deadline CLI dan pengirim mengakses sumber daya Deadline Cloud.

Menyesuaikan peran monitor untuk kasus penggunaan lanjutan

Anda dapat menyesuaikan peran monitor untuk mengubah apa yang dapat dilakukan pengguna di setiap tingkat akses (Penampil, Kontributor, Manajer, Pemilik) atau menambahkan izin untuk alur kerja lanjutan.

Menyesuaikan izin tingkat akses

Keempat kebijakan AWS terkelola yang dilampirkan pada peran monitor mengontrol apa yang dapat dilakukan setiap tingkat akses. Anda dapat menambahkan kebijakan khusus ke peran monitor untuk memberikan atau membatasi izin untuk tingkat akses tertentu menggunakan kunci `deadline:MembershipLevel` kondisi.

Misalnya, untuk mengizinkan Kontributor memperbarui dan membatalkan pekerjaan (yang biasanya dibatasi untuk Manajer dan Pemilik), tambahkan kebijakan seperti berikut ini:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Dengan kebijakan ini, Kontributor dapat memperbarui dan membatalkan pekerjaan selain mengirimkannya.

Menambahkan izin untuk alur kerja lanjutan

Anda dapat menambahkan kebijakan IAM khusus ke peran monitor untuk memberikan izin tambahan kepada semua pengguna monitor. Ini berguna untuk alur kerja scripting lanjutan di mana pengguna memerlukan akses ke Layanan AWS luar fungsionalitas Deadline Cloud standar.

Ikuti panduan ini saat memodifikasi peran monitor Anda:

- Jangan hapus salah satu kebijakan terkelola. Menghapus kebijakan ini merusak fungsionalitas monitor.

Bagaimana monitor Deadline Cloud menggunakan kredensi peran monitor

Deadline Cloud monitor secara otomatis memperoleh kredensi peran monitor saat Anda mengautentikasi. Kemampuan ini memungkinkan aplikasi desktop untuk menyediakan kemampuan pemantauan yang ditingkatkan di luar apa yang tersedia di browser web standar.

Saat Anda masuk dengan monitor Deadline Cloud, secara otomatis membuat profil yang dapat Anda gunakan dengan AWS CLI atau AWS alat lainnya. Profil ini menggunakan kredensi peran monitor, memberi Anda akses terprogram Layanan AWS berdasarkan izin dalam peran monitor Anda.

Deadline Pengirim Cloud bekerja dengan cara yang sama - mereka menggunakan profil yang dibuat oleh monitor Deadline Cloud untuk mengakses Layanan AWS dengan izin peran yang sesuai.

Kustomisasi lanjutan dari peran Deadline Cloud

Anda dapat memperpanjang peran Deadline Cloud dengan izin tambahan untuk mengaktifkan kasus penggunaan lanjutan di luar alur kerja rendering dasar. Pendekatan ini memanfaatkan sistem manajemen akses Deadline Cloud untuk mengontrol akses tambahan Layanan AWS berdasarkan keanggotaan antrian.

Kolaborasi tim dengan AWS CodeCommit

Tambahkan AWS CodeCommit izin ke peran Antrian Anda untuk mengaktifkan kolaborasi tim di repositori proyek. Pendekatan ini menggunakan sistem manajemen akses Deadline Cloud untuk kasus penggunaan tambahan - hanya pengguna dengan akses ke antrian tertentu yang akan menerima AWS CodeCommit izin ini, memungkinkan Anda mengelola akses repositori per proyek melalui keanggotaan antrian Deadline Cloud.

Ini berguna untuk skenario di mana artis perlu mengakses aset, skrip, atau file konfigurasi khusus proyek yang disimpan dalam AWS CodeCommit repositori sebagai bagian dari alur kerja rendering mereka.

Tambahkan AWS CodeCommit izin untuk peran antrian

Tambahkan izin berikut ke peran antrian Anda untuk mengaktifkan AWS CodeCommit akses:

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
```

```
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Menyiapkan penyedia kredensi di workstation artis

Konfigurasikan setiap workstation artis untuk menggunakan kredenal antrian Deadline Cloud untuk akses. AWS CodeCommit Pengaturan ini dilakukan sekali per workstation.

Untuk mengkonfigurasi penyedia kredensi

1. Tambahkan profil penyedia kredensyal ke file AWS konfigurasi () `~/.aws/config` Anda:

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Konfigurasikan Git untuk menggunakan profil ini untuk AWS CodeCommit repositori:

   ```
   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
   ```

Ganti *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* dan *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* dengan ID pertanian dan antrian Anda yang sebenarnya. Ganti *REGION* dengan AWS wilayah Anda (misalnya,`us-west-2`).

Penggunaan AWS CodeCommit dengan kredensyal antrian

Setelah dikonfigurasi, operasi Git akan secara otomatis menggunakan kredensyal peran antrian saat mengakses repositori. AWS CodeCommit `deadline queue export-credentials`Perintah mengembalikan kredensyal sementara yang terlihat seperti ini:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
```

```
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Kredensi ini secara otomatis disegarkan sesuai kebutuhan, dan operasi Git akan bekerja dengan lancar:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Artis sekarang dapat mengakses repositori proyek menggunakan izin antrian mereka tanpa memerlukan kredensyal terpisah. AWS CodeCommit Hanya pengguna dengan akses ke antrian tertentu yang dapat mengakses repositori terkait, memungkinkan kontrol akses berbutir halus melalui sistem keanggotaan antrian Deadline Cloud.

Pemecahan masalah AWS Batas waktu Identitas dan akses Cloud

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Deadline Cloud dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Deadline Cloud
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Deadline Cloud saya

Saya tidak berwenang untuk melakukan tindakan di Deadline Cloud

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `deadline:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `deadline:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Deadline Cloud.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Deadline Cloud. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Deadline Cloud saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Deadline Cloud mendukung fitur-fitur ini, lihatBagaimana Deadline Cloud bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Validasi kepatuhan untuk Deadline Cloud

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Ketahanan di Deadline Cloud

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

AWS Deadline Cloud tidak mencadangkan data yang disimpan di bucket S3 lampiran pekerjaan Anda. Anda dapat mengaktifkan pencadangan data lampiran pekerjaan Anda menggunakan mekanisme pencadangan Amazon S3 standar apa pun, seperti Pembuatan Versi S3 atau. AWS Backup

Keamanan infrastruktur di Deadline Cloud

Sebagai layanan terkelola, AWS Deadline Cloud dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Deadline Cloud melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Deadline Cloud tidak mendukung penggunaan kebijakan titik akhir AWS PrivateLink virtual private cloud (VPC). Ini menggunakan kebijakan AWS PrivateLink default, yang memberikan akses penuh ke titik akhir. Untuk informasi selengkapnya, lihat Kebijakan titik akhir default di panduan AWS PrivateLink pengguna.

Analisis konfigurasi dan kerentanan di Deadline Cloud

AWS menangani tugas-tugas keamanan dasar seperti sistem operasi tamu (OS) dan patching database, konfigurasi firewall, dan pemulihan bencana. Prosedur ini telah ditinjau dan disertifikasi oleh pihak ketiga yang sesuai. Untuk detail selengkapnya, lihat sumber daya berikut:

- Model Tanggung Jawab Bersama
- Amazon Web Services: Gambaran Umum Proses Keamanan (whitepaper)

AWS Deadline Cloud mengelola tugas pada armada yang dikelola layanan atau yang dikelola pelanggan:

- Untuk armada yang dikelola layanan, Deadline Cloud mengelola sistem operasi tamu.
- Untuk armada yang dikelola pelanggan, Anda bertanggung jawab untuk mengelola sistem operasi.

Untuk informasi tambahan tentang konfigurasi dan analisis kerentanan untuk AWS Deadline Cloud, lihat

- Praktik terbaik keamanan untuk Deadline Cloud

Cross-service pencegahan wakil bingung

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Cross-service peniruan identitas dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data untuk semua layanan dengan principal layanan yang telah diberi akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi aws:SourceAccountglobal aws:SourceArndan dalam kebijakan sumber daya untuk membatasi izin yang AWS Deadline Cloud memberikan layanan lain ke sumber daya. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan Nama Sumber Daya Amazon (ARN) lengkap dari sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks global `aws:SourceArn` dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:deadline:*:123456789012:*`.

Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan Deadline Cloud untuk mencegah masalah wakil yang membingungkan.

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"
      }
    }
  }
}
```

Akses AWS Deadline Cloud menggunakan titik akhir antarmuka (AWS PrivateLink)

Anda dapat menggunakan AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan. AWS Deadline Cloud Anda dapat mengakses Deadline Cloud seolah-olah itu ada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi. Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses. Deadline Cloud

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka, yang didukung oleh AWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk

titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditakdirkan. Deadline Cloud

Deadline Cloud juga memiliki titik akhir dual-stack yang tersedia. Dual-stack endpoint mendukung permintaan melalui IPv6 dan IPv4.

Untuk informasi selengkapnya, lihat Mengakses Layanan AWS melalui AWS PrivateLink di Panduan AWS PrivateLink .

Pertimbangan untuk Deadline Cloud

Sebelum menyiapkan titik akhir antarmuka Deadline Cloud, lihat Mengakses layanan AWS menggunakan titik akhir VPC antarmuka di Panduan.AWS PrivateLink

Deadline Cloud mendukung panggilan ke semua tindakan API-nya melalui titik akhir antarmuka.

Secara default, akses penuh ke Deadline Cloud diizinkan melalui titik akhir antarmuka. Atau, Anda dapat mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengontrol lalu lintas Deadline Cloud melalui titik akhir antarmuka.

Deadline Cloud juga mendukung kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat Mengontrol akses ke titik akhir VPC menggunakan kebijakan titik akhir di Panduan.AWS PrivateLink

Deadline Cloud titik akhir

Deadline Cloud menggunakan empat titik akhir untuk akses ke layanan menggunakan AWS PrivateLink - dua untuk IPv4 dan dua untuk IPv6.

Pekerja menggunakan `scheduling.deadline.region.amazonaws.com` endpoint untuk mendapatkan tugas dari antrian, melaporkan kemajuan ke Deadline Cloud, dan mengirim output tugas kembali. Jika Anda menggunakan armada yang dikelola pelanggan, titik akhir penjadwalan adalah satu-satunya titik akhir yang perlu Anda buat kecuali Anda menggunakan operasi manajemen. Misalnya, jika pekerjaan menciptakan lebih banyak pekerjaan, Anda perlu mengaktifkan titik akhir manajemen untuk memanggil `CreateJob` operasi.

Deadline Cloud Monitor menggunakan `management.deadline.region.amazonaws.com` untuk mengelola sumber daya di peternakan Anda, seperti membuat dan memodifikasi antrian dan armada atau mendapatkan daftar pekerjaan, langkah, dan tugas.

AWS SDK dan CLI secara otomatis menambahkan awalan `scheduling` dan `management` ke titik akhir. Jika Anda ingin menonaktifkan perilaku ini, lihat bagian injeksi awalan host di AWS SDK dan Panduan Referensi Alat.

Deadline Cloud juga membutuhkan titik akhir untuk titik akhir AWS layanan berikut:

- Jika Anda menyiapkan armada yang dikelola pelanggan di subnet tanpa koneksi internet, Anda harus membuat titik akhir VPC untuk CloudWatch Amazon Logs agar pekerja dapat menulis log. Untuk informasi selengkapnya, lihat Memantau dengan CloudWatch.
- Jika Anda menggunakan lampiran pekerjaan, Anda harus membuat titik akhir VPC untuk Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) sehingga pekerja dapat mengakses lampiran. Untuk informasi selengkapnya, lihat Lampiran Job di Deadline Cloud.

Buat titik akhir untuk Deadline Cloud

Anda dapat membuat titik akhir antarmuka untuk Deadline Cloud menggunakan konsol VPC Amazon atau () AWS Command Line Interface .AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLink Panduan.

Buat endpoint manajemen dan penjadwalan untuk Deadline Cloud menggunakan nama layanan berikut. Ganti *`region`* dengan Wilayah AWS tempat yang Anda gunakan. Deadline Cloud

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud mendukung titik akhir dual-stack.

Jika Anda mengaktifkan DNS pribadi untuk titik akhir antarmuka, Anda dapat membuat permintaan API untuk Deadline Cloud menggunakan nama DNS Regional default. Misalnya, `scheduling.deadline.us-east-1.amazonaws.com` untuk operasi pekerja, atau `management.deadline.us-east-1.amazonaws.com` untuk semua operasi lainnya.

Jika armada yang dikelola pelanggan berada di subnet tanpa koneksi internet, Anda harus membuat titik akhir CloudWatch Log menggunakan nama layanan berikut:

```
com.amazonaws.region.logs
```

Jika Anda menggunakan lampiran pekerjaan untuk mentransfer file, Anda harus membuat titik akhir Amazon S3 menggunakan nama layanan berikut:

```
com.amazonaws.region.s3
```

Lingkungan jaringan terbatas

Deadline Cloud menyediakan alat yang digunakan oleh artis atau pengguna lain di workstation lokal mereka. Alat-alat ini memerlukan akses ke AWS API dan titik akhir web untuk menjalankan fungsinya. Jika Anda memfilter akses ke AWS domain atau titik akhir URL tertentu dengan menggunakan solusi pemfilteran konten web seperti firewall generasi berikutnya (NGFW) atau Secure Web Gateways (SWG), Anda harus menambahkan domain atau titik akhir URL berikut ke daftar izin solusi pemfilteran konten web Anda.

AWS Titik akhir API untuk daftar yang diizinkan

Tenggat waktu alat klien Cloud, seperti Konsol Manajemen AWS, monitor, CLI, dan pengirim terintegrasi, memerlukan akses AWS ke API selain Deadline Cloud. Titik akhir ini hanya mendukung 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`

Domain web untuk daftar yang diizinkan

Monitor Deadline Cloud memerlukan akses ke domain berikut untuk beroperasi.

Untuk informasi tambahan tentang domain daftar yang diizinkan AWS Sign-In, lihat Domain untuk ditambahkan ke daftar izin Anda di Panduan Pengguna.AWS Sign-In

- `downloads.deadlinecloud.amazonaws.com`

- d2ev1rdnjzhmnr.cloudfront.net
- prod.log.shortbread.aws.dev
- prod.tools.shortbread.aws.dev
- prod.log.shortbread.analytics.console.aws.a2z.com
- prod.tools.shortbread.analytics.console.aws.a2z.com
- global.help-panel.docs.aws.a2z.com
- *[Region]*.signin.aws
- *[Region]*.signin.aws.amazon.com
- sso.*[Region]*.amazonaws.com
- portal.sso.*[Region]*.amazonaws.com
- oidc.*[Region]*.amazonaws.com
- assets.sso-portal.*[Region]*.amazonaws.com

Environment-specific titik akhir untuk daftar yang diizinkan

Domain ini bervariasi tergantung pada konfigurasi spesifik Deadline Cloud. Jika monitor atau antrian Deadline Cloud tambahan dibuat, domain tambahan harus diijinkan.

- *[Directory ID or alias]*.awsapps.com

  Domain ini terkait dengan penyiapan IAM Identity Center dan harus sama untuk semua pengaturan dalam hal ini menggunakan instance IAM Identity Center yang sama. Nilai yang tepat dapat ditemukan oleh admin perusahaan di konsol Pusat Identitas IAM di bawah Pengaturan → Portal akses AWS URL.
- *[Monitor alias]*.*[Region]*.deadlinecloud.amazonaws.com

  Domain ini untuk pengaturan Monitor di Deadline Cloud. Artis memasukkan tautan ini ke browser atau aplikasi monitor Deadline Cloud mereka. Jika Deadline Cloud diatur di akun atau wilayah tambahan di masa mendatang, domain ini akan berubah. Anda dapat menemukan nilai ini di konsol Deadline Cloud di Dasbor → Monitor ikhtisar → Monitor detail → URL.
- *[Bucket name]*.*[Region]*.s3.amazonaws.com

  Ini adalah domain untuk bucket lampiran pekerjaan yang digunakan oleh antrian Deadline Cloud. Setiap antrian dapat memiliki bucket lampiran pekerjaannya sendiri yang dikonfigurasi. Nama bucket yang tepat dapat ditemukan di konsol Deadline Cloud di bawah Antrian → Detail antrian

→ Lampiran Job. Untuk informasi selengkapnya tentang lampiran pekerjaan, lihat dokumentasi antrian.

Praktik terbaik keamanan untuk Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

> **Note**
>
> Untuk informasi selengkapnya tentang pentingnya banyak topik keamanan, lihat Model Tanggung Jawab Bersama.

Perlindungan data

Untuk tujuan perlindungan data, kami menyarankan Anda untuk melindungi Akun AWS kredensil dan menyiapkan akun individual dengan AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon Simple Storage Service (Amazon S3).
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi sensitif apapun, seperti nomor rekening pelanggan Anda, ke dalam kolom isian teks bebas seperti kolom Nama. Rekomendasi ini mencakup saat Anda bekerja dengan AWS Deadline Cloud atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke Deadline Cloud atau layanan lain mungkin diambil untuk dimasukkan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensil di URL untuk memvalidasi permintaan Anda ke server tersebut.

AWS Identity and Access Management izin

Kelola akses ke AWS sumber daya menggunakan pengguna, peran AWS Identity and Access Management (IAM), dan dengan memberikan hak istimewa paling sedikit kepada pengguna. Menetapkan kebijakan dan prosedur manajemen kredensi untuk membuat, mendistribusikan, memutar, dan mencabut kredensi AWS akses. Untuk informasi selengkapnya, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Jalankan pekerjaan sebagai pengguna dan grup

Saat menggunakan fungsionalitas antrian di Deadline Cloud, ini adalah praktik terbaik untuk menentukan pengguna sistem operasi (OS) dan grup utamanya sehingga pengguna OS memiliki izin hak istimewa paling sedikit untuk pekerjaan antrian.

Saat Anda menentukan “Jalankan sebagai pengguna” (dan grup), proses apa pun untuk pekerjaan yang dikirimkan ke antrian akan dijalankan menggunakan pengguna OS tersebut dan akan mewarisi izin OS terkait pengguna tersebut.

Konfigurasi armada dan antrian bergabung untuk membangun postur keamanan. Di sisi antrian, peran “Job run as user” dan IAM dapat ditentukan untuk menggunakan OS dan AWS izin untuk pekerjaan antrian. Armada mendefinisikan infrastruktur (host pekerja, jaringan, penyimpanan bersama yang dipasang) yang, ketika dikaitkan dengan antrian tertentu, menjalankan pekerjaan dalam antrian. Data yang tersedia pada host pekerja perlu diakses oleh pekerjaan dari satu atau lebih antrian terkait. Menentukan pengguna atau grup membantu melindungi data dalam pekerjaan dari antrian lain, perangkat lunak lain yang diinstal, atau pengguna lain dengan akses ke host pekerja. Ketika antrian tanpa pengguna, itu berjalan sebagai pengguna agen yang dapat meniru (`sudo`) setiap pengguna antrian. Dengan cara ini, antrian tanpa pengguna dapat meningkatkan hak istimewa ke antrian lain.

Jaringan

Untuk mencegah lalu lintas dicegat atau dialihkan, penting untuk mengamankan bagaimana dan di mana lalu lintas jaringan Anda diarahkan.

Kami menyarankan Anda mengamankan lingkungan jaringan Anda dengan cara berikut:

- Amankan tabel rute subnet Amazon Virtual Private Cloud (Amazon VPC) untuk mengontrol bagaimana lalu lintas lapisan IP dirutekan.
- Jika Anda menggunakan Amazon Route 53 (Route 53) sebagai penyedia DNS di penyiapan farm atau workstation Anda, amankan akses ke API Route 53.
- Jika Anda terhubung ke Deadline Cloud di luar AWS seperti menggunakan workstation lokal atau pusat data lainnya, amankan infrastruktur jaringan lokal. Ini termasuk server DNS dan tabel rute pada router, switch, dan perangkat jaringan lainnya.

Pekerjaan dan data pekerjaan

Tenggat waktu pekerjaan Cloud berjalan dalam sesi pada host pekerja. Setiap sesi menjalankan satu atau lebih proses pada host pekerja, yang umumnya mengharuskan Anda memasukkan data untuk menghasilkan output.

Untuk mengamankan data ini, Anda dapat mengonfigurasi pengguna sistem operasi dengan antrian. Agen pekerja menggunakan pengguna OS antrian untuk menjalankan sub-proses sesi. Sub-proses ini mewarisi izin pengguna OS antrian.

Kami menyarankan Anda mengikuti praktik terbaik untuk mengamankan akses ke data akses sub-proses ini. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

Struktur pertanian

Anda dapat mengatur armada Deadline Cloud dan antrian banyak cara. Namun, ada implikasi keamanan dengan pengaturan tertentu.

Sebuah peternakan memiliki salah satu batas paling aman karena tidak dapat berbagi sumber daya Deadline Cloud dengan peternakan lain, termasuk armada, antrian, dan profil penyimpanan. Namun, Anda dapat berbagi AWS sumber daya eksternal di dalam peternakan, yang membahayakan batas keamanan.

Anda juga dapat menetapkan batas keamanan antara antrian dalam peternakan yang sama menggunakan konfigurasi yang sesuai.

Ikuti praktik terbaik ini untuk membuat antrian aman di peternakan yang sama:

- Kaitkan armada hanya dengan antrian dalam batas keamanan yang sama. Perhatikan hal-hal berikut:
  - Setelah pekerjaan berjalan di host pekerja, data mungkin tetap tertinggal, seperti di direktori sementara atau direktori home pengguna antrian.
  - Pengguna OS yang sama menjalankan semua pekerjaan pada host pekerja armada milik layanan, terlepas dari antrian mana Anda mengirimkan pekerjaan.
  - Pekerjaan mungkin membiarkan proses berjalan pada host pekerja, sehingga memungkinkan pekerjaan dari antrian lain untuk mengamati proses berjalan lainnya.
- Pastikan hanya antrian dalam batas keamanan yang sama yang berbagi bucket Amazon S3 untuk lampiran pekerjaan.
- Pastikan bahwa hanya antrian dalam batas keamanan yang sama berbagi pengguna OS.
- Amankan AWS sumber daya lain yang terintegrasi ke dalam pertanian hingga batas.

Antrian lampiran pekerjaan

Lampiran Job dikaitkan dengan antrian, yang menggunakan bucket Amazon S3 Anda.

- Lampiran Job menulis dan membaca dari awalan root di bucket Amazon S3. Anda menentukan awalan root ini dalam panggilan `CreateQueue` API.
- Bucket memiliki kode yang sesuai`Queue Role`, yang menentukan peran yang memberi pengguna antrian akses ke awalan bucket dan root. Saat membuat antrian, Anda menentukan Nama Sumber Daya `Queue Role` Amazon (ARN) di samping bucket lampiran pekerjaan dan awalan root.
- Panggilan resmi ke`AssumeQueueRoleForRead`,`AssumeQueueRoleForUser`, dan operasi `AssumeQueueRoleForWorker` API mengembalikan satu set kredensil keamanan sementara untuk. `Queue Role`

Jika Anda membuat antrian dan menggunakan kembali bucket Amazon S3 dan awalan root, ada risiko informasi diungkapkan kepada pihak yang tidak berwenang. Misalnya, QueueA dan QueueB berbagi bucket dan awalan root yang sama. Dalam alur kerja yang aman, ArtisTa memiliki akses ke QueueA tetapi tidak QueueB. Namun, ketika beberapa antrian berbagi bucket, ArtisTA dapat

mengakses data dalam data QueueB karena menggunakan bucket dan awalan root yang sama dengan QueueA.

Konsol mengatur antrian yang aman secara default. Pastikan antrian memiliki kombinasi yang berbeda antara bucket Amazon S3 dan awalan root kecuali mereka merupakan bagian dari batas keamanan umum.

Untuk mengisolasi antrian Anda, Anda harus mengonfigurasi `Queue Role` untuk hanya mengizinkan akses antrian ke bucket dan awalan root. Dalam contoh berikut, ganti masing-masing *`placeholder`* dengan informasi spesifik sumber daya Anda.

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/*"
```

```
        }
    ]
}
```

Anda juga harus menetapkan kebijakan kepercayaan tentang peran tersebut. Dalam contoh berikut, ganti *placeholder* teks dengan informasi spesifik sumber daya Anda.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
```

```
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

Bucket Amazon S3 perangkat lunak khusus

Anda dapat menambahkan pernyataan berikut ke perangkat lunak khusus `Queue Role` untuk mengakses perangkat lunak khusus di bucket Amazon S3 Anda. Dalam contoh berikut, ganti *SOFTWARE_BUCKET_NAME* dengan nama bucket S3 Anda dan *BUCKET_ACCOUNT_OWNER* dengan Akun AWS ID yang memiliki 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"
         }
      }
    }
]
```

Untuk informasi selengkapnya tentang praktik terbaik keamanan Amazon S3, lihat Praktik terbaik keamanan untuk Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Tuan rumah pekerja

Host pekerja aman untuk membantu memastikan bahwa setiap pengguna hanya dapat melakukan operasi untuk peran yang ditetapkan.

Kami merekomendasikan praktik terbaik berikut untuk mengamankan host pekerja:

- Menggunakan skrip konfigurasi host dapat mengubah keamanan dan operasi pekerja. Konfigurasi yang salah dapat menyebabkan pekerja menjadi tidak stabil atau berhenti bekerja. Adalah tanggung jawab Anda untuk men-debug kegagalan tersebut.
- Jangan gunakan `jobRunAsUser` nilai yang sama dengan beberapa antrian kecuali pekerjaan yang dikirimkan ke antrian tersebut berada dalam batas keamanan yang sama.
- Jangan atur antrian `jobRunAsUser` ke nama pengguna OS yang dijalankan oleh agen pekerja.
- Berikan izin OS dengan hak istimewa paling sedikit kepada pengguna antrian yang diperlukan untuk beban kerja antrian yang dimaksud. Pastikan bahwa mereka tidak memiliki izin menulis sistem file untuk bekerja file program agen atau perangkat lunak bersama lainnya.
- Pastikan hanya pengguna root yang aktif Linux dan `Administrator` memiliki akun sendiri Windows dan dapat memodifikasi file program agen pekerja.
- Pada host Linux pekerja, pertimbangkan untuk mengonfigurasi `umask` penggantian `/etc/sudoers` yang memungkinkan pengguna agen pekerja meluncurkan proses sebagai pengguna antrian. Konfigurasi ini membantu memastikan pengguna lain tidak dapat mengakses file yang ditulis ke antrian.
- Berikan individu tepercaya akses paling tidak memiliki hak istimewa ke host pekerja.
- Batasi izin untuk mengganti file konfigurasi DNS lokal (`/etc/hosts`aktif dan aktifWindows), Linux dan untuk merutekan tabel `C:\Windows\system32\etc\hosts` di workstation dan sistem operasi host pekerja.
- Batasi izin untuk konfigurasi DNS pada workstation dan sistem operasi host pekerja.
- Secara teratur menambal sistem operasi dan semua perangkat lunak yang diinstal. Pendekatan ini mencakup perangkat lunak yang khusus digunakan dengan Deadline Cloud seperti submitter, adaptor, agen pekerja, OpenJD paket, dan lain-lain.
- Gunakan kata sandi yang kuat untuk Windows antrian`jobRunAsUser`.
- Putar kata sandi untuk antrian `jobRunAsUser` Anda secara teratur.
- Pastikan akses hak istimewa paling sedikit ke rahasia Windows kata sandi dan hapus rahasia yang tidak digunakan.

- Jangan berikan `jobRunAsUser` izin antrian perintah jadwal untuk dijalankan di masa mendatang:
  - LinuxAktif, tolak akses akun ini ke `cron` dan`at`.
  - WindowsAktif, tolak akses akun ini ke penjadwal Windows tugas.

> **Note**
>
> Untuk informasi selengkapnya tentang pentingnya menambal sistem operasi dan perangkat lunak yang diinstal secara teratur, lihat Model Tanggung Jawab Bersama.

Skrip konfigurasi host

- Menggunakan skrip konfigurasi host dapat mengubah keamanan dan operasi pekerja. Konfigurasi yang salah dapat menyebabkan pekerja menjadi tidak stabil atau berhenti bekerja. Adalah tanggung jawab Anda untuk men-debug kegagalan tersebut.

Workstation

Sangat penting untuk mengamankan workstation dengan akses ke Deadline Cloud. Pendekatan ini membantu memastikan bahwa pekerjaan apa pun yang Anda kirimkan ke Deadline Cloud tidak dapat menjalankan beban kerja sewenang-wenang yang ditagih ke Anda. Akun AWS

Kami merekomendasikan praktik terbaik berikut untuk mengamankan workstation artis. Untuk informasi selengkapnya, lihat Model Tanggung Jawab Bersama.

- Amankan semua kredensil tetap yang menyediakan akses ke AWS, termasuk Deadline Cloud. Untuk informasi lebih lanjut, lihat Mengelola access key untuk pengguna IAM dalam Panduan Pengguna IAM.
- Hanya instal perangkat lunak tepercaya dan aman.
- Mengharuskan pengguna berfederasi dengan penyedia identitas untuk mengakses AWS dengan kredensi sementara.
- Gunakan izin aman pada file program submitter Deadline Cloud untuk mencegah gangguan.
- Berikan individu tepercaya akses paling tidak istimewa ke workstation artis.
- Hanya gunakan pengirim dan adaptor yang Anda dapatkan melalui Deadline Cloud Monitor.

- Batasi izin ke DNS lokal mengganti file konfigurasi (`/etc/hosts`on Linux danmacOS, dan `C:\Windows\system32\etc\hosts` onWindows), dan untuk merutekan tabel pada workstation dan sistem operasi host pekerja.
- Batasi izin `/etc/resolve.conf` pada workstation dan sistem operasi host pekerja.
- Secara teratur menambal sistem operasi dan semua perangkat lunak yang diinstal. Pendekatan ini mencakup perangkat lunak yang khusus digunakan dengan Deadline Cloud seperti submitter, adaptor, agen pekerja, OpenJD paket, dan lain-lain.

Verifikasi keaslian perangkat lunak yang diunduh

Verifikasi keaslian perangkat lunak Anda setelah mengunduh penginstal untuk melindungi dari gangguan file. Prosedur ini berfungsi untuk keduanya Windows dan Linux sistem.

Windows

Untuk memverifikasi keaslian file yang Anda unduh, selesaikan langkah-langkah berikut.

1. Dalam perintah berikut, ganti *`file`* dengan file yang ingin Anda verifikasi. Misalnya, ***`C:\PATH\TO\MY\`*`DeadlineCloudSubmitter-windows-x64-installer.exe`** . Juga, ganti *`signtool-sdk-version`* dengan versi SignTool SDK yang diinstal. Misalnya, **`10.0.22000.0`**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Misalnya, Anda dapat memverifikasi file installer submitter Deadline Cloud dengan menjalankan perintah berikut:

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-
   windows-x64-installer.exe
   ```

Linux

Untuk memverifikasi keaslian file yang Anda unduh, gunakan alat baris gpg perintah.

1. Impor OpenPGP kunci dengan menjalankan perintah berikut:

   ```
   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. Tentukan apakah akan mempercayai OpenPGP kuncinya. Beberapa faktor yang perlu dipertimbangkan ketika memutuskan apakah akan mempercayai kunci di atas termasuk yang berikut:

   - Koneksi internet yang Anda gunakan untuk mendapatkan kunci GPG dari situs web ini aman.
   - Perangkat tempat Anda mengakses situs web ini aman.
   - AWS telah mengambil langkah-langkah untuk mengamankan hosting kunci OpenPGP publik di situs web ini.

3. Jika Anda memutuskan untuk mempercayai OpenPGP kunci, edit kunci untuk dipercaya dengan gpg mirip dengan contoh berikut:

```
$ 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. Verifikasi penginstal pengirim Cloud Deadline

   Untuk memverifikasi installer submitter Deadline Cloud, selesaikan langkah-langkah berikut:

a. Unduh file tanda tangan untuk penginstal pengirim Deadline Cloud.

   Unduh file tanda tangan (.sig)

b. Verifikasi tanda tangan penginstal submitter Deadline Cloud dengan menjalankan:

```
gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
DeadlineCloudSubmitter-linux-x64-installer.run
```

5. Verifikasi monitor Cloud Deadline

> **Note**
>
> Anda dapat memverifikasi unduhan monitor Deadline Cloud menggunakan file tanda tangan atau metode khusus platform. Untuk metode khusus platform, lihat Linux (Debian) tab, tab Linux (RPM), atau Linux (AppImage) tab berdasarkan jenis file yang Anda unduh.

Untuk memverifikasi aplikasi desktop monitor Deadline Cloud dengan file tanda tangan, selesaikan langkah-langkah berikut:

a. Unduh file tanda tangan yang sesuai untuk penginstal monitor Deadline Cloud Anda:

   - Unduh file tanda tangan.deb
   - Unduh file tanda tangan.rpm
   - Unduh. AppImage berkas tanda tangan

b. Verifikasi tanda tangan:

   Untuk.deb:

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

   Untuk .rpm:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

Untuk. AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Konfirmasikan bahwa output terlihat mirip dengan yang berikut:

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Jika output berisi frasa`Good signature from "AWS Deadline Cloud"`, itu berarti tanda tangan telah berhasil diverifikasi dan Anda dapat menjalankan skrip instalasi monitor Deadline Cloud.

Kunci Sejarah

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
```

```
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

Untuk memverifikasi paket yang menggunakan fileLinux. AppImage biner, pertama selesaikan langkah 1-3 di Linux tab, lalu selesaikan langkah-langkah berikut.

1. Dari AppImageUpdate halaman di GitHub, unduh validate-x86_64. AppImageberkas.
2. Setelah mengunduh file, untuk menambahkan izin eksekusi, jalankan perintah berikut.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```
3. Untuk menambahkan izin eksekusi, jalankan perintah berikut.

   ```
   chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```
4. Untuk memverifikasi tanda tangan monitor Deadline Cloud, jalankan perintah berikut.

   ```
   ./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

   Jika output berisi frasa`Validation successful`, itu berarti tanda tangan telah berhasil diverifikasi dan Anda dapat menjalankan skrip instalasi monitor Deadline Cloud dengan aman.

Linux (Debian)

Untuk memverifikasi paket yang menggunakan Linux biner.deb, pertama-tama selesaikan langkah 1-3 di tab. Linux

dpkg adalah alat manajemen paket inti di sebagian besar distribusi debian berbasisLinux. Anda dapat memverifikasi file.deb dengan alat ini.

1. Unduh file monitor Deadline Cloud .deb:

   Unduh Deadline Cloud monitor (.deb)

2. Verifikasi file.deb:

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. Outputnya akan mirip dengan:

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. Untuk memverifikasi file.deb, konfirmasikan bahwa `GOODSIG` ada dalam output.

Linux (RPM)

Untuk memverifikasi paket yang menggunakan Linux biner.rpm, pertama-tama selesaikan langkah 1-3 di Linux tab.

1. Unduh file monitor Deadline Cloud .rpm:

   Unduh Monitor Deadline Cloud (.rpm)

2. Verifikasi file.rpm:

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. Outputnya akan mirip dengan:

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. Untuk memverifikasi file.rpm, konfirmasikan yang `digests signatures OK` ada di output.

Riwayat dokumen

Untuk informasi tentang pembaruan ke AWS Deadline Cloud, lihat catatan rilis Deadline Cloud.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.