aws

Panduan Developerr

# Amazon Simple Workflow Service

Versi API 2012-01-25

# Amazon Simple Workflow Service: Panduan Developerr

# Table of Contents

# Apa itu Amazon Simple Workflow Service?

Dengan Amazon Simple Workflow Service (Amazon Simple Workflow Service), Anda dapat membuat, menjalankan, dan menskalakan pekerjaan latar belakang yang memiliki langkah paralel atau berurutan. Anda dapat mengoordinasikan pekerjaan di seluruh komponen terdistribusi dan melacak status tugas.

Di Amazon SWF, tugas mewakili unit kerja logis yang dilakukan oleh komponen aplikasi Anda. Mengkoordinasikan tugas di seluruh termasuk mengelola dependensi antar-tugas, penjadwalan, dan konkurensi dalam alur aplikasi Anda. Dengan Amazon SWF, Anda dapat mengontrol dan mengoordinasikan tugas tanpa mengkhawatirkan kompleksitas yang mendasarinya, seperti melacak kemajuan dan mempertahankan status tugas.

Saat menggunakan Amazon SWF, Anda menerapkan pekerja untuk melakukan tugas. Pekerja dapat menjalankan infrastruktur cloud, seperti Amazon Elastic Compute Cloud (Amazon EC2), atau di tempat Anda sendiri. Anda dapat membuat tugas jangka panjang, atau yang mungkin gagal, waktu habis—atau memerlukan mulai ulang—atau yang mungkin lengkap dengan berbagai throughput dan latensi. Amazon SWF menyimpan tugas dan menugaskannya kepada pekerja saat mereka siap, melacak kemajuan, dan mempertahankan status, termasuk detail penyelesaian tugas.

Untuk mengoordinasikan tugas, Anda menulis program yang mendapatkan status tugas terbaru dari Amazon SWF dan menggunakan status tersebut untuk memulai tugas berikutnya. Amazon SWF mempertahankan status eksekusi aplikasi secara tahan lama, sehingga aplikasi Anda tahan terhadap kegagalan komponen individual. Dengan Amazon SWF, Anda dapat membuat, menyebarkan, menskalakan, dan memodifikasi komponen aplikasi secara independen.

> **Layanan AWS alur kerja lainnya**
>
> Untuk sebagian besar kasus penggunaan, kami sarankan AWS Step Functions untuk mempertimbangkan alur kerja dan kebutuhan orkestrasi Anda.
> Dengan Step Functions, Anda dapat membuat alur kerja, juga disebut mesin status, untuk membangun aplikasi terdistribusi, mengotomatiskan proses, mengatur layanan mikro, dan membuat pipeline data dan pembelajaran mesin. Di konsol atau AWS toolkit Step Functions di VS Code, Anda dapat menggunakan Workflow Studio grafis untuk memvisualisasikan, mengedit, menguji, dan men-debug alur kerja aplikasi Anda.
> Untuk informasi teknis selengkapnya, lihat Panduan AWS Step Functions Pengembang.

# Mengembangkan komponen alur kerja dengan Amazon SWF

Mengembangkan aplikasi terdistribusi membutuhkan koordinasi banyak komponen dan berurusan dengan latensi dan ketidakandalan yang melekat dalam komunikasi jarak jauh.

Dengan Amazon Simple Workflow Service (Amazon Simple Workflow Service), Anda dapat mengembangkan aplikasi asinkron dan terdistribusi dengan menyediakan model dan infrastruktur pemrograman untuk mengoordinasikan komponen terdistribusi dan mempertahankan status pelaksanaannya dengan cara yang andal. Dengan mengandalkan Amazon SWF, Anda dibebaskan untuk fokus pada pembangunan aspek aplikasi Anda yang membedakannya.

## Komponen alur kerja

Komponen alur kerjaKonsep dasar di Amazon SWF adalah alur kerja. Alur kerja adalah satu set aktivitas yang melaksanakan beberapa tujuan, bersama dengan logika yang mengkoordinasikan aktivitas. Misalnya, alur kerja dapat menerima pesanan pelanggan dan mengambil tindakan apa pun yang diperlukan untuk memenuhi pesanan.

Setiap alur kerja berjalan dalam sumber daya yang disebut domain, yang mengontrol cakupan alur kerja. Akun AWS dapat memiliki beberapa domain, masing-masing dapat berisi beberapa alur kerja, tetapi alur kerja di domain yang berbeda tidak dapat berinteraksi.

Saat mendesain alur kerja Amazon SWF, Anda menentukan setiap aktivitas yang diperlukan. Kemudian, Anda mendaftarkan setiap aktivitas dengan Amazon SWF sebagai jenis aktivitas. Anda akan memberikan nama, versi, dan nilai batas waktu. Misalnya, pelanggan mungkin memiliki harapan bahwa pesanan akan dikirim dalam waktu 24 jam.

Dalam proses melaksanakan alur kerja, beberapa aktivitas mungkin perlu dilakukan lebih dari sekali, mungkin dengan input yang bervariasi. Misalnya, dalam alur kerja pesanan pelanggan, Anda mungkin memiliki aktivitas yang menangani item yang dibeli. Jika pelanggan membeli beberapa item, maka aktivitas ini harus menjalankan beberapa kali. Amazon SWF memiliki konsep tugas aktivitas yang mewakili salah satu invocation dari suatu aktivitas. Dalam contoh kita, pemrosesan setiap item akan direpresentasikan oleh tugas aktivitas tunggal.

Pekerja aktivitas adalah program yang menerima tugas aktivitas, melaksanakannya, dan memberikan hasil. Tugas itu sebenarnya bisa dilakukan oleh seseorang. Misalnya, seorang analis statistik mungkin menerima set data, menganalisis data, dan kemudian mengirim kembali analisisnya.

Tugas aktivitas, dan pekerja aktivitas yang melakukannya, dapat berjalan secara sinkron atau asinkron. Pekerja dapat berjalan di satu lokasi atau didistribusikan di beberapa komputer, berpotensi di wilayah geografis yang berbeda. Pekerja aktivitas yang berbeda dapat ditulis dalam bahasa pemrograman yang berbeda dan berjalan pada sistem operasi yang berbeda. Misalnya, satu pekerja aktivitas mungkin berjalan di server di Asia, sementara yang lain mungkin berjalan di perangkat seluler di Amerika Utara.

Logika koordinasi dalam alur kerja terdapat dalam program perangkat lunak yang disebut decider. Penentu menjadwalkan tugas aktivitas, memberikan masukan kepada pekerja aktivitas, memproses peristiwa yang tiba saat alur kerja sedang berlangsung, dan mengakhiri (atau menutup) alur kerja setelah tujuan terpenuhi.

Peran layanan Amazon SWF adalah berfungsi sebagai hub pusat yang andal di mana data dipertukarkan antara decider, pekerja aktivitas, dan entitas lain yang relevan seperti seseorang yang mengelola alur kerja. Amazon SWF juga memelihara kejadian setiap eksekusi alur kerja, yang menghemat aplikasi Anda dari keharusan untuk menyimpan state dengan cara yang tahan lama.

Decider mengarahkan alur kerja dengan menerima tugas keputusan dari Amazon SWF dan merespons kembali ke Amazon SWF dengan keputusan. Keputusan mewakili tindakan atau serangkaian tindakan, yang merupakan langkah selanjutnya dalam alur kerja. Sebuah keputusan umum akan menjadwalkan tugas aktivitas. Keputusan juga dapat digunakan untuk menunda tugas dengan pengatur waktu, meminta pembatalan tugas yang sedang berlangsung, dan untuk menyelesaikan alur kerja.

Mekanisme dimana kedua pekerja aktivitas dan decider menerima tugas mereka (tugas aktivitas dan tugas keputusan masing-masing) adalah dengan polling layanan Amazon SWF.

Amazon SWF menginformasikan decider state alur kerja dengan memasukkan, dengan setiap tugas keputusan, salinan riwayat eksekusi alur kerja saat ini. Riwayat eksekusi alur kerja terdiri dari kejadian, di mana suatu kejadian merepresentasikan perubahan signifikan dalam state eksekusi alur kerja. Contoh peristiwa termasuk penyelesaian tugas, waktu habis tugas, atau kedaluwarsa timer. Riwayat adalah catatan progres alur kerja yang lengkap, konsisten, dan otoritatif.

Amazon SWF Access Control menggunakan AWS Identity and Access Management (IAM), sehingga Anda dapat mengontrol akses ke sumber daya. AWS Misalnya, Anda dapat mengizinkan pengguna mengakses akun Anda, namun hanya untuk menjalankan alur kerja tertentu di domain tertentu.

# Menjalankan alur kerja Anda

Berikut ini memberikan ikhtisar langkah-langkah yang diperlukan untuk mengembangkan dan menjalankan alur kerja di Amazon SWF:

1. Tulis pekerja aktivitas untuk melakukan langkah-langkah pemrosesan dalam alur kerja Anda.
2. Tulis penentu untuk menangani logika koordinasi alur kerja Anda.
3. Mendaftarkan kegiatan Anda dan alur kerja dengan Amazon SWF.

   Anda dapat melakukan langkah ini secara terprogram atau dengan menggunakan. Konsol Manajemen AWS
4. Memulai pekerja aktivitas Anda dan decider.

   Aktor ini dapat berjalan pada setiap perangkat komputasi yang dapat mengakses endpoint Amazon SWF. Misalnya, Anda dapat menggunakan instance komputasi di cloud, seperti Amazon Elastic Compute Cloud (Amazon EC2); server di pusat data Anda; atau bahkan perangkat seluler, untuk meng-host decider atau pekerja aktivitas. Setelah dimulai, decider dan aktivitas pekerja harus memulai polling Amazon SWF untuk tugas-tugas.
5. Memulai satu atau lebih eksekusi alur kerja Anda.

   Anda dapat memulai alur kerja secara terprogram atau melalui file. Konsol Manajemen AWS

   Setiap eksekusi berjalan secara independen dan Anda dapat menyediakan masing-masing dengan set input datanya sendiri. Ketika eksekusi dimulai, Amazon SWF menjadwal tugas keputusan awal. Sebagai tanggapan, penentu Anda mulai menghasilkan keputusan yang memulai tugas aktivitas. Eksekusi berlanjut sampai keputusanmu membuat keputusan untuk menutup eksekusi.
6. Lihat eksekusi alur kerja menggunakan file. Konsol Manajemen AWS

   Anda dapat memfilter dan melihat detail lengkap eksekusi yang sedang berjalan dan selesai. Misalnya, Anda dapat memilih eksekusi terbuka untuk melihat tugas mana yang telah diselesaikan dan apa hasilnya.

# Menyiapkan lingkungan pengembangan Anda

Anda memiliki opsi untuk mengembangkan Amazon SWF di salah satu bahasa pemrograman yang didukung oleh. AWS Untuk pengembang Java, AWS Flow Framework ini juga tersedia. Untuk

informasi selengkapnya, lihat AWS Flow Frameworksitus web, dan lihat AWS Flow Framework Panduan Pengembang Java.

Untuk mengurangi latensi dan menyimpan data di lokasi yang memenuhi persyaratan Anda, Amazon SWF menyediakan titik akhir di Wilayah yang berbeda.

Setiap titik akhir di Amazon SWF sepenuhnya independen. Setiap domain, alur kerja, dan aktivitas yang telah Anda daftarkan di satu Wilayah tidak akan berbagi data atau atribut dengan yang ada di Wilayah lain.

Saat Anda mendaftarkan domain, alur kerja, atau aktivitas Amazon SWF, domain tersebut hanya ada di Wilayah tempat Anda mendaftarkannya. Misalnya, Anda dapat mendaftarkan domain yang diberi nama `SWF-Flows-1` di dua Wilayah berbeda, tetapi mereka tidak akan berbagi data atau atribut satu sama lain — masing-masing bertindak sebagai domain yang sepenuhnya independen.

Untuk daftar endpoint Amazon SWF, lihat Wilayah dan Endpoint.

## Kembangkan dengan AWS SDKs

Amazon SWF didukung oleh AWS SDKs untuk Java, .NET, Node.js, PHP, Python, dan Ruby, menyediakan cara yang nyaman untuk menggunakan Amazon SWF HTTP API dalam bahasa pemrograman pilihan Anda.

Anda dapat mengembangkan decider, activity worker, atau starter alur kerja menggunakan API yang diekspos oleh library ini. Dan, Anda dapat menggunakan operasi visibilitas melalui pustaka ini sehingga Anda dapat mengembangkan alat pemantauan dan pelaporan Amazon SWF Anda sendiri.

Untuk mengunduh alat untuk mengembangkan dan mengelola aplikasi AWS, termasuk SDKs, buka Pusat Pengembang.

Untuk informasi terperinci tentang operasi Amazon SWF di setiap SDK, lihat dokumentasi referensi khusus bahasa untuk SDK.

## Pertimbangkan AWS Flow Framework

AWS Flow Framework Ini adalah SDK yang disempurnakan untuk menulis program asinkron terdistribusi yang berjalan sebagai alur kerja di Amazon SWF. Kerangka kerja ini tersedia untuk bahasa pemrograman Java dan menyediakan kelas untuk menulis program terdistribusi yang kompleks.

Dengan AWS Flow Framework, Anda menggunakan tipe yang telah dikonfigurasi untuk memetakan definisi alur kerja Anda langsung ke metode dalam program Anda. AWS Flow Framework Mendukung konsep berorientasi objek standar, seperti penanganan kesalahan berbasis pengecualian. Program yang ditulis dengan AWS Flow Framework dapat dibuat, dijalankan, dan di-debug sepenuhnya dalam editor atau IDE pilihan Anda. Untuk informasi selengkapnya, lihat AWS Flow Frameworksitus web, dan lihat AWS Flow Framework Panduan Pengembang Java.

# Memulai dengan Amazon SWF

Anda dapat memulai dengan aplikasi alur kerja Amazon Simple Workflow Service berikut yang terdiri dari satu set empat aktivitas yang beroperasi secara berurutan. Tutorial ini juga mencakup topik-topik berikut:

- Mengatur default dan execution-time (waktu eksekusi) dari opsi alur kerja dan aktivitas.
- Polling Amazon SWF untuk tugas keputusan dan aktivitas.
- Meneruskan data antara aktivitas dan alur kerja dengan Amazon SWF.
- Menunggu human tasks (tugas manusia) dan melaporkan heatbeats ke Amazon SWF dari tugas aktivitas.
- Menggunakan Amazon SNS untuk membuat topik, membuat pengguna berlangganan topik tersebut, dan memublikasikan pesan ke titik akhir langganan.

Anda dapat menggunakan Amazon SWF dan Amazon Simple Notification Service (Amazon SNS) bersama-sama untuk meniru alur kerja “tugas manusia” — alur kerja di mana pekerja manusia diminta untuk melakukan beberapa tindakan dan kemudian berkomunikasi dengan Amazon SWF untuk meluncurkan aktivitas berikutnya dalam alur kerja.

Karena Amazon SWF adalah layanan web berbasis cloud, komunikasi dengan Amazon SWF dapat dilakukan di mana saja selama koneksi ke Internet tersedia. Dalam hal ini, kita akan menggunakan Amazon SNS untuk berkomunikasi dengan pengguna baik melalui email, pesan teks SMS, atau keduanya.

Tutorial ini menggunakan AWS SDK for Rubyuntuk mengakses Amazon SWF dan Amazon SNS, tetapi ada banyak opsi pengembangan yang tersedia, termasuk AWS Flow Framework untuk Ruby, yang menyediakan koordinasi dan komunikasi yang lebih mudah dengan Amazon SWF.

> **Note**
>
> Tutorial ini menggunakan AWS SDK for Ruby, tetapi kami sarankan Anda menggunakan AWS Flow Framework untuk Java.

Topik

# Tentang Alur Kerja

Alur kerja yang akan kita kembangkan terdiri dari empat langkah utama:

1. Dapatkan alamat berlangganan (email atau SMS) dari pengguna.
2. Buat topik SNS dan berlangganan titik akhir yang disediakan untuk topik tersebut.
3. Tunggu pengguna mengonfirmasi langganan.
4. Jika pengguna mengonfirmasi, publikasikan pesan ucapan selamat ke topik tersebut.

Langkah-langkah ini mencakup aktivitas yang sepenuhnya dilakukan secara otomatis (langkah 2 dan 4), dan yang lainnya memerlukan alur kerja yang dilakukan manusia untuk menyediakan beberapa data bagi aktivitas sebelum alur kerja dapat melanjutkan (langkah 1 dan 3).

Setiap langkah bergantung pada data yang dihasilkan pada langkah sebelumnya (Anda harus memiliki titik akhir sebelum berlangganan topik, dan Anda harus memiliki langganan topik sebelum Anda menunggu konfirmasi, dll.) Tutorial ini juga akan membahas cara menyediakan hasil aktivitas setelah proses selesai, dan cara meneruskan input ke tugas yang sedang dijadwalkan. Amazon SWF menangani koordinasi dan penyampaian informasi antara aktivitas dan alur kerja, dan sebaliknya.

Kami juga menggunakan input keyboard dan Amazon SNS untuk menangani komunikasi antara Amazon SWF dan manusia yang menyediakan data ke alur kerja. Dalam prakteknya, Anda dapat menggunakan banyak teknik yang berbeda untuk berkomunikasi dengan pengguna manusia, tetapi Amazon SNS menyediakan cara yang sangat mudah dengan menggunakan email atau pesan teks untuk memberitahu pengguna tentang peristiwa dalam alur kerja.

# Prasyarat

Untuk mengikuti tutorial ini, Anda membutuhkan:

- Akun Amazon Web Services
- Penerjemah Ruby
- AWS SDK for Ruby

Jika Anda sudah menyiapkan hal-hal tersebut, Anda siap untuk melanjutkan. Jika Anda tidak ingin menjalankan contoh, Anda masih dapat mengikuti tutorial — sebagian besar konten dalam tutorial ini berlaku untuk menggunakan Amazon SWF dan Amazon SNS terlepas dari opsi pengembangan yang Anda pilih.

# Langkah-Langkah Tutorial

Tutorial ini dibagi menjadi langkah-langkah berikut:

1. Tutorial Alur Kerja Berlangganan Bagian 1: Menggunakan Amazon SWF dengan AWS SDK for Ruby
2. Tutorial Alur Kerja Langganan Bagian 2: Menerapkan Alur Kerja
3. Tutorial Langganan Alur Kerja Bagian 3: Menerapkan Aktivitas
4. Tutorial Alur Kerja Langganan Bagian 4: Menerapkan Poller Tugas Aktivitas
5. Tutorial Alur Kerja Beerlangganan: Menjalankan Alur Kerja

# Tutorial Alur Kerja Berlangganan Bagian 1: Menggunakan Amazon SWF dengan AWS SDK for Ruby

Topik

- Sertakan AWS SDK for Ruby
- Mengkonfigurasi Sesi AWS
- Mendaftarkan Domain Amazon SWF
- Langkah selanjutnya

## Sertakan AWS SDK for Ruby

Mulailah dengan membuat file bernama `utils.rb`. Kode dalam file ini akan memperoleh, atau membuat jika perlu, domain Amazon SWF yang digunakan oleh kode alur kerja dan aktivitas dan akan menyediakan tempat untuk meletakkan kode yang umum untuk semua kelas kita.

Pertama, kita perlu menyertakan pustaka `aws-sdk-v1` dalam kode kita, sehingga kita dapat menggunakan fitur yang disediakan oleh SDK for Ruby.

```
require 'aws-sdk-v1'
```

Ini memberi kami akses ke AWS namespace, yang menyediakan kemampuan untuk menetapkan nilai terkait sesi global, seperti AWS kredensi dan wilayah Anda, dan juga menyediakan akses ke layanan. AWS APIs

## Mengkonfigurasi Sesi AWS

Kami akan mengonfigurasi AWS Sesi dengan menetapkan AWS kredensional kami (yang diperlukan untuk mengakses AWS layanan) dan AWS wilayah yang akan digunakan.

Ada beberapa cara untuk mengatur AWS kredensional di AWS SDK for Ruby: dengan menyetelnya dalam variabel lingkungan AWS_ACCESS (_KEY_ID dan AWS_SECRET _ACCESS_KEY) atau dengan menyetelnya dengan. `AWS.config` Kami akan menggunakan metode terakhir, yaitu memuatnya dari file konfigurasi YAML, disebut `aws-config.txt`, yang terlihat seperti ini.

```
---
:access_key_id: REPLACE_WITH_ACCESS_KEY_ID
:secret_access_key: REPLACE_WITH_SECRET_ACCESS_KEY
```

Buat file ini sekarang, ganti string yang dimulai dengan REPLACE_WITH_ dengan ID kunci AWS akses dan kunci akses rahasia Anda. Untuk informasi tentang kunci AWS akses Anda, lihat Bagaimana Cara Mendapatkan Kredensi Keamanan? dalam Referensi Umum Amazon Web Services.

Kita juga perlu mengatur AWS wilayah yang akan digunakan. Karena kami akan menggunakan Short Message Service (SMS) (Layanan Pesan Singkat) untuk mengirim pesan teks ke ponsel pengguna dengan Amazon SNS, kami perlu memastikan bahwa kami menggunakan wilayah yang didukung oleh Amazon SNS. Lihat Wilayah dan Negara yang Didukung di Panduan Developer Amazon Simple Notification Service.

> **Note**
>
> Jika Anda tidak memiliki akses ke us-east-1, atau tidak tertarik untuk menjalankan demo dengan mengaktifkan olahpesan SMS, jangan ragu untuk menggunakan wilayah mana pun yang Anda inginkan. Anda dapat menghapus fungsionalitas SMS dari sampel dan menggunakan email sebagai satu-satunya endpoint untuk berlangganan topik Amazon SNS. Untuk informasi selengkapnya tentang mengirim pesan SMS, lihat Mengirim dan Menerima Notifikasi SMS Menggunakan Amazon SNS di Panduan Developer Amazon Simple Notification Service.

Sekarang kita akan menambahkan beberapa kode ke `utils.rb` untuk memuat file konfigurasi, mendapatkan kredensial pengguna, lalu memberikan kredensial dan wilayah ke `AWS.config`.

```
require 'yaml'

# Load the user's credentials from a file, if it exists.
begin
  config_file = File.open('aws-config.txt') { |f| f.read }
rescue
  puts "No config file! Hope you set your AWS credentials in the environment..."
end

if config_file.nil?
  options = { }
else
  options = YAML.load(config_file)
end

# SMS Messaging (which can be used by Amazon SNS) is available only in the
# `us-east-1` region.
$SMS_REGION = 'us-east-1'
options[:region] = $SMS_REGION

# Now, set the options
AWS.config = options
```

# Mendaftarkan Domain Amazon SWF

Untuk menggunakan Amazon SWF, Anda perlu mengatur domain: entitas bernama yang akan menampung alur kerja dan aktivitas Anda. Anda dapat memiliki banyak domain Amazon SWF yang terdaftar, tetapi semuanya harus memiliki nama unik dalam AWS akun Anda, dan alur kerja tidak dapat berinteraksi di seluruh domain: Semua alur kerja dan aktivitas untuk aplikasi Anda harus berada dalam domain yang sama untuk berinteraksi satu sama lain.

Karena kita akan menggunakan domain yang sama di seluruh aplikasi kita, kita akan membuat fungsi yang *`utils.rb`* dipanggil*`init_domain`*, yang akan mengambil domain Amazon SWF bernama DomainSWFSample.

Setelah Anda mendaftarkan domain, Anda dapat menggunakannya kembali untuk eksekusi alur kerja yang banyak. Namun, itu adalah kesalahan untuk mencoba mendaftarkan domain yang sudah ada, jadi kode kami akan memeriksa terlebih dahulu untuk melihat apakah domain itu ada, dan akan menggunakan domain yang ada jika dapat ditemukan. Jika domain tidak dapat ditemukan, kami akan membuatnya.

Untuk bekerja dengan domain Amazon SWF di SDK for Ruby, AWS::Simplegunakan Workflow.domains, yang menampilkan DomainCollectiondomain yang dapat digunakan untuk menghitung dan mendaftarkan domain:

- Untuk memeriksa apakah domain sudah terdaftar, Anda dapat melihat daftar yang disediakan AWS::Simpleworkflowoleh.domains.registered.
- Untuk mendaftarkan domain baru, gunakan AWS::Simpleworkflow.domains.register.

Berikut adalah kode untuk `init_domain` di `utils.rb`.

```
# Registers the domain that the workflow will run in.
def init_domain
  domain_name = 'SWFSampleDomain'
  domain = nil
  swf = AWS::SimpleWorkflow.new

  # First, check to see if the domain already exists and is registered.
  swf.domains.registered.each do | d |
    if(d.name == domain_name)
      domain = d
      break
    end
```

```
  end

  if domain.nil?
    # Register the domain for one day.
    domain = swf.domains.create(
      domain_name, 1, { :description => "#{domain_name} domain" })
  end

  return domain
end
```

## Langkah selanjutnya

Selanjutnya, Anda akan membuat alur kerja dan kode starter di Tutorial Alur Kerja Langganan Bagian 2: Menerapkan Alur Kerja.

# Tutorial Alur Kerja Langganan Bagian 2: Menerapkan Alur Kerja

Sampai sekarang, kode kami masih cukup generik. Ini adalah bagian di mana kita mulai benar-benar menentukan apa yang alur kerja kita lakukan, dan aktivitas apa yang kita perlukan untuk menerapkannya.

Topik



## Merancang Alur Kerja

Jika Anda mengingat kembali, gagasan awal untuk alur kerja ini terdiri dari langkah-langkah berikut:

1. Dapatkan alamat berlangganan (email atau SMS) dari pengguna.
2. Buat topik SNS dan berlangganan titik akhir yang disediakan untuk topik tersebut.
3. Tunggu pengguna mengonfirmasi langganan.

4. Jika pengguna mengonfirmasi, publikasikan pesan ucapan selamat ke topik tersebut.

Kita dapat menganggap setiap langkah dalam alur kerja kita sebagai aktivitas yang harus dijalankan oleh alur kerja. Alur Kerja kita bertanggung jawab untuk menjadwalkan setiap aktivitas pada waktu yang tepat, dan mengoordinasikan transfer data antar aktivitas.

Untuk alur kerja ini, kita akan membuat aktivitas terpisah untuk setiap langkah berikut, menamainya secara deskriptif:

1. get_contact_activity
2. subscribe_topic_activity
3. wait_for_confirmation_activity
4. send_result_activity

Aktivitas ini akan dijalankan secara berurutan, dan data dari setiap langkah akan digunakan pada langkah berikutnya.

Kita bisa merancang aplikasi kita sehingga semua kode berada dalam satu file sumber, tapi ini berjalan bertentangan dengan cara yang dirancang Amazon SWF. Cara ini dirancang untuk alur kerja yang dapat menjangkau seluruh Internet dalam ruang lingkup, jadi mari kita setidaknya membagi aplikasi menjadi dua executable terpisah:

- `swf_sns_workflow.rb` - Berisi alur kerja dan starter alur kerja.
- `swf_sns_activities.rb` - Berisi aktivitas dan starter aktivitas.

Penerapan alur kerja dan aktivitas dapat dijalankan di jendela terpisah, komputer terpisah, atau bahkan bagian dunia yang berbeda. Karena Amazon SWF melacak detail alur kerja dan aktivitas Anda, alur kerja Anda dapat mengkoordinasikan penjadwalan dan transfer data aktivitas Anda di mana pun mereka berjalan.

## Menyiapkan Kode Alur Kerja

Kita akan mulai dengan membuat sebuah file bernama `swf_sns_workflow.rb`. Dalam file ini, deklarasikan kelas yang disebut. SampleWorkflow Berikut adalah deklarasi kelas dan konstruktornya, metode `initialize`.

```
require_relative 'utils.rb'
```

```
# SampleWorkflow - the main workflow for the SWF/SNS Sample
#
# See the file called `README.md` for a description of what this file does.
class SampleWorkflow

  attr_accessor :name

  def initialize(workflowId)

    # the domain to look for decision tasks in.
    @domain = init_domain

    # the task list is used to poll for decision tasks.
    @workflowId = workflowId

    # The list of activities to run, in order. These name/version hashes can be
    # passed directly to AWS::SimpleWorkflow::DecisionTask#schedule_activity_task.
    @activity_list = [
      { :name => 'get_contact_activity', :version => 'v1' },
      { :name => 'subscribe_topic_activity', :version => 'v1' },
      { :name => 'wait_for_confirmation_activity', :version => 'v1' },
      { :name => 'send_result_activity', :version => 'v1' },
    ].reverse! # reverse the order... we're treating this like a stack.

    register_workflow
  end
```

Seperti yang Anda lihat, kita menyimpan data instans kelas berikut:

- `domain` - Nama domain yang diambil dari `init_domain` di `utils.rb`.
- `workflowId` - Daftar tugas diteruskan ke `initialize`.
- `activity_list` - Daftar aktivitas, yang memiliki nama dan versi dari aktivitas yang akan kita jalankan.

Nama domain, nama aktivitas, dan versi aktivitas sudah mencukupi bagi Amazon SWF untuk mengidentifikasi tipe aktivitas secara positif, jadi itu semua merupakan data tentang aktivitas kita yang perlu disimpan untuk menjadwalkan aktivitas.

Daftar tugas akan digunakan oleh kode decider alur kerja untuk melakukan polling tugas keputusan dan aktivitas jadwal.

Pada akhir fungsi ini, kita memanggil metode yang belum kita tentukan: `register_workflow`. Kita akan menentukan metode ini selanjutnya.

## Mendaftarkan Alur Kerja

Untuk menggunakan tipe alur kerja, pertama-tama kita harus mendaftarkannya. Seperti tipe aktivitas, tipe alur kerja dapat diidentifikasi berdasarkan domain, nama, dan versinya. Selain itu, seperti domain dan tipe aktivitas, Anda tidak dapat mendaftarkan ulang tipe alur kerja yang ada. Jika Anda perlu mengubah apa pun tentang tipe alur kerja, Anda harus menyediakannya dengan versi baru, yang pada dasarnya menciptakan tipe baru.

Berikut adalah kode untuk `register_workflow`, yang digunakan untuk mengambil tipe alur kerja yang ada yang telah kita daftarkan pada eksekusi sebelumnya atau untuk mendaftarkan alur kerja jika belum terdaftar.

```
  # Registers the workflow
  def register_workflow
    workflow_name = 'swf-sns-workflow'
    @workflow_type = nil

    # a default value...
    workflow_version = '1'

    # Check to see if this workflow type already exists. If so, use it.
    @domain.workflow_types.each do | a |
      if (a.name == workflow_name) && (a.version == workflow_version)
        @workflow_type = a
      end
    end

    if @workflow_type.nil?
      options =  {
        :default_child_policy => :terminate,
        :default_task_start_to_close_timeout => 3600,
        :default_execution_start_to_close_timeout => 24 * 3600 }

      puts "registering workflow: #{workflow_name}, #{workflow_version},
#{options.inspect}"
      @workflow_type = @domain.workflow_types.register(workflow_name, workflow_version,
options)
    end
```

```
    puts "** registered workflow: #{workflow_name}"
  end
```

Pertama, kita memeriksa untuk melihat apakah nama dan versi alur kerja sudah terdaftar dengan mengiterasi melalui koleksi workflow_types domain. Jika kita menemukan kecocokan, kita akan menggunakan tipe alur kerja yang sudah terdaftar.

Jika kami tidak menemukan kecocokan, maka jenis alur kerja baru terdaftar (dengan memanggil register pada `workflow_types` koleksi yang sama dengan tempat kami mencari alur kerja) dengan nama '', versi swf-sns-workflow '1', dan opsi berikut.

```
      options =  {
        :default_child_policy => :terminate,
        :default_task_start_to_close_timeout => 3600,
        :default_execution_start_to_close_timeout => 24 * 3600 }
```

Opsi yang diteruskan selama pendaftaran digunakan untuk mengatur perilaku default untuk tipe alur kerja kita, jadi kita tidak perlu mengatur nilai-nilai ini setiap kali kita mulai mengeksekusi alur kerja baru.

Di sini, kita hanya menetapkan beberapa nilai batas waktu: waktu maksimum yang dapat diambil dari saat tugas mulai hingga menutup (satu jam), dan waktu maksimum yang dapat digunakan untuk menyelesaikan eksekusi alur kerja (24 jam). Jika salah satu dari waktu tersebut terlampaui, tugas atau alur kerja akan mencapai batas waktu.

Untuk informasi lebih lanjut tentang nilai-nilai batas waktu tersebut, lihat Tipe Batas Waktu Amazon SWF.

## Polling untuk Keputusan

Pada inti dari setiap eksekusi alur kerja terdapat decider. Tanggung jawab decider adalah untuk mengelola eksekusi alur kerja itu sendiri. Decider menerima tugas keputusan dan meresponsnya, baik dengan menjadwalkan aktivitas baru, membatalkan dan memulai ulang aktivitas, atau dengan menetapkan status eksekusi alur kerja sebagai selesai, dibatalkan, atau gagal.

Decider menggunakan nama daftar tugas eksekusi alur kerja untuk menerima tugas keputusan untuk direspons. Untuk melakukan polling untuk tugas keputusan, panggil poll pada koleksi decision_tasks domain untuk melakukan loop pada tugas keputusan yang tersedia. Anda kemudian dapat memeriksa kejadian baru dalam tugas keputusan dengan mengiterasi pada koleksi new_events.

Peristiwa yang dikembalikan adalah AWS::SimpleWorkflow::HistoryEventobjek, dan Anda bisa mendapatkan jenis acara dengan menggunakan anggota event_type acara yang dikembalikan. Untuk daftar dan deskripsi jenis peristiwa riwayat, lihat HistoryEventdi Referensi API Layanan Alur Kerja Sederhana Amazon.

Berikut adalah awal dari logika poller tugas keputusan ini. Sebuah metode baru di kelas alur kerja kita disebut sebagai `poll_for_decisions`.

```
def poll_for_decisions
  # first, poll for decision tasks...
  @domain.decision_tasks.poll(@workflowId) do | task |
    task.new_events.each do | event |
      case event.event_type
```

Sekarang kita akan membagi eksekusi decider kita berdasarkan `event_type` yang diterima. Yang pertama yang mungkin kita terima adalah WorkflowExecutionStarted. Ketika kejadian ini diterima, artinya Amazon SWF memberi sinyal untuk decider Anda bahwa decider harus memulai eksekusi alur kerja. Kita akan mulai dengan menjadwalkan aktivitas pertama dengan memanggil schedule_activity_task pada tugas yang kita terima saat melakukan polling.

Kita akan meneruskan aktivitas pertama yang kita nyatakan dalam daftar aktivitas kita, yang, karena kita membalik daftar sehingga kita dapat menggunakannya seperti tumpukan, menempati posisi `last` pada daftar. “Aktivitas” yang kita tentukan hanyalah berupa peta yang terdiri dari nama dan nomor versi, tapi inilah yang dibutuhkan Amazon SWF untuk mengidentifikasi aktivitas penjadwalan, dengan asumsi bahwa aktivitas tersebut telah terdaftar.

```
        when 'WorkflowExecutionStarted'
          # schedule the last activity on the (reversed, remember?) list to
          # begin the workflow.
          puts "** scheduling activity task: #{@activity_list.last[:name]}"

          task.schedule_activity_task( @activity_list.last,
            { :workflowId => "#{@workflowId}-activities" } )
```

Ketika kita menjadwalkan suatu aktivitas, Amazon SWF mengirimkan tugas aktivitas ke daftar tugas aktivitas yang kita teruskan saat menjadwalkannya, memberi sinyal pada tugas untuk mulai. Kita akan menangani tugas aktivitas di Tutorial Langganan Alur Kerja Bagian 3: Menerapkan Aktivitas, tetapi perlu dicatat bahwa kita tidak mengeksekusi tugas di sini. Kita hanya memberitahu Amazon SWF bahwa tugas perlu dijadwalkan.

Aktivitas berikutnya yang perlu kita atasi adalah ActivityTaskCompletedacara, yang terjadi ketika Amazon SWF telah menerima respons aktivitas yang diselesaikan dari tugas aktivitas.

```
          when 'ActivityTaskCompleted'
            # we are running the activities in strict sequential order, and
            # using the results of the previous activity as input for the next
            # activity.
            last_activity = @activity_list.pop

            if(@activity_list.empty?)
              puts "!! All activities complete! Sending complete_workflow_execution..."
              task.complete_workflow_execution
              return true;
            else
              # schedule the next activity, passing any results from the
              # previous activity. Results will be received in the activity
              # task.
              puts "** scheduling activity task: #{@activity_list.last[:name]}"
              if event.attributes.has_key?('result')
                task.schedule_activity_task(
                  @activity_list.last,
                  { :input => event.attributes[:result],
                    :workflowId => "#{@workflowId}-activities" } )
              else
                task.schedule_activity_task(
                  @activity_list.last, { :workflowId => "#{@workflowId}-activities" } )
              end
            end
```

Karena kita menjalankan tugas kita secara linier, dan hanya satu aktivitas yang dijalankan sekaligus, kita akan mengambil kesempatan ini untuk mengeluarkan tugas yang sudah selesai dari tumpukan. `activity_list` Jika hal ini menghasilkan daftar kosong, maka kita tahu bahwa alur kerja kita sudah selesai. Dalam hal ini, kita mengirim sinyal kepada Amazon SWF bahwa alur kerja kita sudah selesai dengan memanggil complete_workflow_execution pada tugas.

Jika daftar masih memiliki entri, kita akan menjadwalkan aktivitas berikutnya dalam daftar (sekali lagi, di posisi terakhir). Namun, kali ini, kita akan melihat untuk memeriksa apakah aktivitas sebelumnya mengembalikan data hasil apa pun ke Amazon SWF setelah diselesaikan, yang disediakan untuk alur kerja dalam atribut kejadian, di kunci `result`. Jika aktivitas menghasilkan hasil, kita akan meneruskannya sebagai opsi `input` ke aktivitas terjadwal berikutnya bersama dengan daftar tugas aktivitas.

Dengan mengambil nilai-nilai `result` aktivitas yang sudah selesai, dan dengan menetapkan nilai-nilai `input` aktivitas yang dijadwalkan, kita dapat meneruskan data dari satu aktivitas ke aktivitas berikutnya, atau kita dapat menggunakan data dari suatu aktivitas untuk mengubah perilaku dalam decider kita berdasarkan hasil dari suatu aktivitas.

Untuk tujuan tutorial ini, dua tipe kejadian ini adalah yang paling penting dalam menentukan perilaku alur kerja kita. Namun, suatu kegiatan dapat menghasilkan peristiwa selain ActivityTaskCompleted. Kita akan membungkus kode decider kita dengan menyediakan kode demonstrasi handler untuk ActivityTaskTimedOutdan ActivityTaskFailedevent, dan untuk WorkflowExecutionCompletedevent tersebut, yang akan dihasilkan saat Amazon SWF memproses `complete_workflow_execution` panggilan yang kita buat saat kita kehabisan aktivitas untuk dijalankan.

```
          when 'ActivityTaskTimedOut'
            puts "!! Failing workflow execution! (timed out activity)"
            task.fail_workflow_execution
            return false

          when 'ActivityTaskFailed'
            puts "!! Failing workflow execution! (failed activity)"
            task.fail_workflow_execution
            return false

          when 'WorkflowExecutionCompleted'
            puts "## Yesss, workflow execution completed!"
            task.workflow_execution.terminate
            return false
        end
      end
    end
end
```

## Memulai Eksekusi Alur Kerja

Sebelum tugas keputusan akan dibuat untuk alur kerja untuk di-polling, kita perlu memulai eksekusi alur kerja.

Untuk memulai eksekusi alur kerja, panggil start_execution pada tipe alur kerja terdaftar Anda (). AWS::SimpleWorkflow::WorkflowType Kita akan menentukan pembungkus kecil di sekitarnya untuk memanfaatkan anggota instans `workflow_type` yang telah kita dapatkan dalam konstruktor kelas.

```
  def start_execution
```

```
    workflow_execution = @workflow_type.start_execution( {
      :workflowId => @workflowId } )
    poll_for_decisions
  end
end
```

Setelah alur kerja dieksekusi, kejadian keputusan akan mulai muncul di daftar tugas alur kerja, yang diteruskan sebagai opsi eksekusi alur kerja di start_execution.

Tidak seperti opsi yang disediakan ketika tipe alur kerja didaftarkan, opsi yang diteruskan ke `start_execution` tidak dianggap sebagai bagian dari tipe alur kerja. Anda bebas untuk mengubahnya per eksekusi alur kerja tanpa mengubah versi alur kerja.

Karena kita ingin alur kerja mulai mengeksekusi ketika kita menjalankan file, tambahkan beberapa kode yang membuat instance kelas dan kemudian memanggil `start_execution` metode yang baru saja kita definisikan.

```
if __FILE__ == $0
  require 'securerandom'

  # Use a different task list name every time we start a new workflow execution.
  #
  # This avoids issues if our pollers re-start before SWF considers them closed,
  # causing the pollers to get events from previously-run executions.
  workflowId = SecureRandom.uuid

  # Let the user start the activity worker first...

  puts ""
  puts "Amazon SWF Example"
  puts "------------------"
  puts ""
  puts "Start the activity worker, preferably in a separate command-line window, with"
  puts "the following command:"
  puts ""
  puts "> ruby swf_sns_activities.rb #{workflowId}-activities"
  puts ""
  puts "You can copy & paste it if you like, just don't copy the '>' character."
  puts ""
  puts "Press return when you're ready..."

  i = gets
```

```
  # Now, start the workflow.

  puts "Starting workflow execution."
  sample_workflow = SampleWorkflow.new(workflowId)
  sample_workflow.start_execution
end
```

Untuk menghindari konflik penamaan daftar tugas, kita akan menggunakan `SecureRandom.uuid` untuk menghasilkan UUID acak yang dapat kita gunakan sebagai nama daftar tugas, yang menjamin bahwa nama daftar tugas yang berbeda digunakan untuk setiap eksekusi alur kerja.

> **Note**
>
> Daftar tugas digunakan untuk merekam kejadian tentang eksekusi alur kerja, jadi jika Anda menggunakan daftar tugas yang sama untuk beberapa eksekusi dari tipe alur kerja yang sama, Anda mungkin mendapatkan kejadian yang dihasilkan selama eksekusi sebelumnya, terutama jika Anda menjalankannya dalam suksesi yang dekat satu sama lain, yang sering terjadi ketika mencoba kode baru atau menjalankan tes.

Agar terhindar dari keharusan untuk berurusan dengan artefak dari eksekusi sebelumnya, kita dapat menggunakan daftar tugas baru untuk setiap eksekusi, menentukannya ketika kita memulai eksekusi alur kerja.

Terdapat pula sedikit kode di sini untuk menyediakan instruksi bagi orang yang menjalankannya (mungkin Anda), dan untuk menyediakan versi “aktivitas” dari daftar tugas. Decider menggunakan nama daftar tugas ini untuk menjadwalkan aktivitas untuk alur kerja, dan penerapan aktivitas akan mendengarkan kejadian aktivtas pada nama daftar tugas ini untuk mengetahui kapan harus memulai aktivitas terjadwal dan untuk menyediakan pembaruan tentang eksekusi aktivitas.

Kode juga menunggu pengguna untuk mulai menjalankan starter aktivitas sebelum memulai eksekusi alur kerja, sehingga starter aktivitas akan siap merespons ketika tugas aktivitas mulai muncul di daftar tugas yang disediakan.

## Langkah selanjutnya

Anda telah menerapkan alur kerja. Selanjutnya, Anda akan menentukan aktivitas dan starter aktivitas, di Tutorial Langganan Alur Kerja Bagian 3: Menerapkan Aktivitas.

# Tutorial Langganan Alur Kerja Bagian 3: Menerapkan Aktivitas

Sekarang kita akan menerapkan setiap aktivitas dalam alur kerja kita, dimulai dengan kelas dasar yang menyediakan beberapa fitur umum untuk kode aktivitas.

Topik



## Menentukan Jenis Kegiatan Dasar

Saat mendesain alur kerja, kita mengidentifikasi aktivitas berikut:

- `get_contact_activity`
- `subscribe_topic_activity`
- `wait_for_confirmation_activity`
- `send_result_activity`

Kita akan menerapkan tiap-tiap kegiatan ini sekarang. Karena aktivitas kita akan berbagi beberapa fitur, mari kita lakukan sedikit dasar dan buat beberapa kode umum yang dapat mereka bagikan. Kami akan menyebutnya BasicActivity, dan mendefinisikannya dalam file baru bernama`basic_activity.rb`.

Seperti file sumber lainnya, kita akan menyertakan `utils.rb` untuk mengakses fungsi `init_domain` untuk menyiapkan domain sampel.

```
require_relative 'utils.rb'
```

Selanjutnya, kita akan menyatakan kelas aktivitas dasar dan beberapa data umum yang akan kita minati untuk setiap aktivitas. Kita akan menyimpan AWS::SimpleWorkflow::ActivityTypeinstance aktivitas, nama, dan hasil dalam atribut kelas.

```
class BasicActivity

  attr_accessor :activity_type
  attr_accessor :name
  attr_accessor :results
```

Atribut ini mengakses instans data yang ditentukan dalam metode `initialize` kelas, yang menggunakan nama aktivitas, dan versi opsional dan peta opsi yang akan digunakan saat mendaftarkan aktivitas dengan Amazon SWF.

```
  def initialize(name, version = 'v1', options = nil)

    @activity_type = nil
    @name = name
    @results = nil

    # get the domain to use for activity tasks.
    @domain = init_domain

    # Check to see if this activity type already exists.
    @domain.activity_types.each do | a |
      if (a.name == @name) && (a.version == version)
        @activity_type = a
      end
    end

    if @activity_type.nil?
      # If no options were specified, use some reasonable defaults.
      if options.nil?
        options = {
          # All timeouts are in seconds.
          :default_task_heartbeat_timeout => 900,
          :default_task_schedule_to_start_timeout => 120,
          :default_task_schedule_to_close_timeout => 3800,
          :default_task_start_to_close_timeout => 3600 }
      end
      @activity_type = @domain.activity_types.register(@name, version, options)
    end
  end
```

Seperti halnya pendaftaran tipe alur kerja, jika tipe aktivitas sudah terdaftar, kita dapat mengambilnya dengan melihat koleksi activity_types domain. Jika aktivitas tidak dapat ditemukan, aktivitas tersebut akan didaftarkan.

Selain itu, seperti pada tipe alur kerja, Anda dapat mengatur opsi default yang disimpan dengan tipe aktivitas Anda saat Anda mendaftarkannya.

Hal terakhir yang didapatkan aktivitas dasar kita adalah cara konsisten untuk menjalankannya. Kita akan menentukan metode `do_activity` yang menggunakan tugas aktivitas. Seperti yang ditunjukkan, kita dapat menggunakan tugas aktivitas yang diteruskan untuk menerima data melalui atribut instans `input` tugas tersebut.

```
    def do_activity(task)
      @results = task.input # may be nil
      return true
    end
  end
```

Itu membungkus BasicActivitykelas. Sekarang kita akan menggunakannya untuk membuat penentuan aktivitas kita menjadi sederhana dan konsisten.

## Mendefinisikan GetContactActivity

Aktivitas pertama yang dijalankan selama eksekusi alur kerja adalah `get_contact_activity`, yang mengambil informasi berlangganan topik Amazon SNS pengguna.

Buat file baru bernama`get_contact_activity.rb`, dan memerlukan keduanya`yaml`, yang akan kita gunakan untuk menyiapkan string untuk diteruskan ke Amazon SWF, dan`basic_activity.rb`, yang akan kita gunakan sebagai dasar untuk kelas ini GetContactActivity.

```
require 'yaml'
require_relative 'basic_activity.rb'

# **GetContactActivity** provides a prompt for the user to enter contact
# information. When the user successfully enters contact information, the
# activity is complete.
class GetContactActivity < BasicActivity
```

Karena kami memasukkan kode registrasi aktivitas BasicActivity, `initialize` metode untuk GetContactActivityini cukup sederhana. Kita cukup memanggil konstruktor kelas dasar dengan nama

aktivitas, `get_contact_activity`. Ini semua adalah hal yang diperlukan untuk mendaftarkan aktivitas kita.

```
    # initialize the activity
    def initialize
      super('get_contact_activity')
    end
```

Kita sekarang akan mendefinisikan do`_activity` metode, yang meminta nomor and/or telepon email pengguna.

```
    def do_activity(task)
      puts ""
      puts "Please enter either an email address or SMS message (mobile phone) number
to"
      puts "receive SNS notifications. You can also enter both to use both address
types."
      puts ""
      puts "If you enter a phone number, it must be able to receive SMS messages, and
must"
      puts "be 11 digits (such as 12065550101 to represent the number
1-206-555-0101)."

      input_confirmed = false
      while !input_confirmed
        puts ""
        print "Email: "
        email = $stdin.gets.strip

        print "Phone: "
        phone = $stdin.gets.strip

        puts ""
        if (email == '') && (phone == '')
          print "You provided no subscription information. Quit? (y/n)"
           confirmation = $stdin.gets.strip.downcase
           if confirmation == 'y'
             return false
           end
        else
           puts "You entered:"
           puts "  email: #{email}"
           puts "  phone: #{phone}"
```

```
            print "\nIs this correct? (y/n): "
            confirmation = $stdin.gets.strip.downcase
            if confirmation == 'y'
              input_confirmed = true
            end
          end
        end

        # make sure that @results is a single string. YAML makes this easy.
        @results = { :email => email, :sms => phone }.to_yaml
        return true
      end
    end
```

Di akhir `do_activity`, kita mengambil email dan nomor telepon yang didapatkan dari pengguna, menempatkannya di peta dan kemudian menggunakan `to_yaml` untuk mengkonversi seluruh peta ke string YAML. Ada alasan penting untuk hal ini: setiap hasil yang Anda teruskan ke Amazon SWF ketika Anda menyelesaikan suatu aktivitas harus berupa data string saja. Kemampuan Ruby untuk dengan mudah mengonversi objek ke string YAML dan kemudian kembali lagi ke objek, untungnya, sesuai untuk tujuan ini.

Itulah akhir dari implementasi `get_contact_activity`. Data ini akan digunakan selanjutnya di implementasi `subscribe_topic_activity`.

## Mendefinisikan SubscribeTopicActivity

Sekarang kita akan mendalami Amazon SNS dan membuat aktivitas yang menggunakan informasi yang dihasilkan oleh `get_contact_activity` untuk menjadikan pengguna berlangganan topik Amazon SNS.

Buat sebuah file baru bernama `subscribe_topic_activity.rb`, tambahkan same requirements (persyaratan yang sama) yang kita gunakan untuk `get_contact_activity`, declare your class (nyatakan kelas Anda), dan provide its method (sediakan metode `initialize`).

```
require 'yaml'
require_relative 'basic_activity.rb'

# **SubscribeTopicActivity** sends an SMS / email message to the user, asking for
# confirmation.  When this action has been taken, the activity is complete.
class SubscribeTopicActivity < BasicActivity
```

```
    def initialize
      super('subscribe_topic_activity')
    end
```

Sekarang setelah kita menempatkan kode agar aktivitas dapat disiapkan dan terdaftar, kita akan menambahkan beberapa kode untuk membuat topik Amazon SNS. Untuk melakukannya, kita akan menggunakan metode create_topic AWS::SNS::Clientobjek.

Tambahkan metode `create_topic` ke kelas Anda, yang membawa objek klien Amazon SNS yang diteruskan.

```
    def create_topic(sns_client)
      topic_arn = sns_client.create_topic(:name => 'SWF_Sample_Topic')[:topic_arn]

      if topic_arn != nil
        # For an SMS notification, setting `DisplayName` is *required*. Note that
        # only the *first 10 characters* of the DisplayName will be shown on the
        # SMS message sent to the user, so choose your DisplayName wisely!
        sns_client.set_topic_attributes( {
          :topic_arn => topic_arn,
          :attribute_name => 'DisplayName',
          :attribute_value => 'SWFSample' } )
      else
        @results = {
          :reason => "Couldn't create SNS topic", :detail => "" }.to_yaml
        return nil
      end

      return topic_arn
    end
```

Setelah kami memiliki Nama Sumber Daya Amazon (ARN) topik, kami dapat menggunakannya dengan metode set_topic_attributes klien Amazon SNS untuk mengatur topik, yang diperlukan untuk mengirim pesan SMS dengan DisplayNameAmazon SNS.

Terakhir, kita akan menentukan metode `do_activity`. Kami akan memulai dengan mengumpulkan data apa pun yang diteruskan melalui opsi `input` ketika aktivitas dijadwalkan. Seperti disebutkan sebelumnya, data ini harus diteruskan sebagai string, yang kita buat menggunakan `to_yaml`. Saat mengambilnya, kita akan menggunakan `YAML.load` untuk mengubah data menjadi objek Ruby.

Berikut adalah awal dari `do_activity`, di mana kita mengambil data input.

```
    def do_activity(task)
      activity_data = {
        :topic_arn => nil,
        :email => { :endpoint => nil, :subscription_arn => nil },
        :sms => { :endpoint => nil, :subscription_arn => nil },
      }

      if task.input != nil
        input = YAML.load(task.input)
        activity_data[:email][:endpoint] = input[:email]
        activity_data[:sms][:endpoint] = input[:sms]
      else
        @results = { :reason => "Didn't receive any input!", :detail => "" }.to_yaml
        puts("  #{@results.inspect}")
        return false
      end

      # Create an SNS client. This is used to interact with the service. Set the
      # region to $SMS_REGION, which is a region that supports SMS notifications
      # (defined in the file `utils.rb`).
      sns_client = AWS::SNS::Client.new(
        :config => AWS.config.with(:region => $SMS_REGION))
```

Jika kita tidak menerima input apapun, tidak banyak yang harus dilakukan, jadi kita cukup menggagalkan aktivitas.

Dengan asumsi bahwa semuanya baik-baik saja, bagaimanapun, kami akan terus mengisi `do_activity` metode kami, mendapatkan klien Amazon SNS dengan AWS SDK for Ruby, dan meneruskannya ke metode `create_topic` kami untuk membuat topik Amazon SNS.

```
      # Create the topic and get the ARN
      activity_data[:topic_arn] = create_topic(sns_client)

      if activity_data[:topic_arn].nil?
        return false
      end
```

Ada beberapa hal yang perlu diperhatikan di sini:

- Kita menggunakan `AWS.config.with` untuk mengatur wilayah untuk klien Amazon SNS kita. Karena kita ingin mengirim pesan SMS, kita menggunakan wilayah yang mengaktifkan SMS yang kita nyatakan di `utils.rb`.

- Kita menyimpan topik ARN di peta `activity_data` kita. Ini adalah bagian dari data yang akan diteruskan ke aktivitas selanjutnya dalam alur kerja kita.

Terakhir, aktivitas ini menjadikan pengguna berlangganan topik Amazon SNS, menggunakan titik akhir yang diteruskan (email dan SMS). Kita tidak mengharuskan pengguna untuk masuk ke kedua titik akhir, tapi kita perlu setidaknya satu.

```
      # Subscribe the user to the topic, using either or both endpoints.
      [:email, :sms].each do | x |
        ep = activity_data[x][:endpoint]
        # don't try to subscribe an empty endpoint
        if (ep != nil && ep != "")
          response = sns_client.subscribe( {
            :topic_arn => activity_data[:topic_arn],
            :protocol => x.to_s, :endpoint => ep } )
          activity_data[x][:subscription_arn] = response[:subscription_arn]
        end
      end
```

AWS::SNS::Client.subscribe mengambil topik ARN, protokol (yang, dengan cerdik, kami menyamar sebagai `activity_data` kunci peta untuk titik akhir yang sesuai).

Akhirnya, kita memaketkan kembali informasi untuk aktivitas berikutnya dalam format YAML, sehingga kita dapat mengirimkannya kembali ke Amazon SWF.

```
      # if at least one subscription arn is set, consider this a success.
      if (activity_data[:email][:subscription_arn] != nil) or (activity_data[:sms]
[:subscription_arn] != nil)
        @results = activity_data.to_yaml
      else
        @results = { :reason => "Couldn't subscribe to SNS topic", :detail =>
 "" }.to_yaml
        puts("  #{@results.inspect}")
        return false
      end
      return true
    end
  end
```

Hal ini menyelesaikan penerapan `subscribe_topic_activity`. Selanjutnya, kita akan menentukan `wait_for_confirmation_activity`.

# Mendefinisikan WaitForConfirmationActivity

Setelah pengguna berlangganan topik Amazon SNS, pelanggan masih perlu mengkonfirmasi permintaan berlangganan. Dalam hal ini, kita akan menunggu pengguna untuk mengonfirmasi melalui email atau pesan SMS.

Aktivitas yang menunggu pengguna untuk mengonfirmasi langganan disebut `wait_for_confirmation_activity`, dan kita akan menentukannya di sini. Untuk memulai, buat file baru bernama `wait_for_confirmation_activity.rb` dan siapkan file seperti kita menyiapkan aktivitas sebelumnya.

```
require 'yaml'
require_relative 'basic_activity.rb'

# **WaitForConfirmationActivity** waits for the user to confirm the SNS
# subscription.  When this action has been taken, the activity is complete. It
# might also time out...
class WaitForConfirmationActivity < BasicActivity

  # Initialize the class
  def initialize
    super('wait_for_confirmation_activity')
  end
```

Selanjutnya, kita akan mulai menentukan metode `do_activity` dan mengambil data input apapun ke dalam variabel lokal yang disebut `subscription_data`.

```
  def do_activity(task)
    if task.input.nil?
      @results = { :reason => "Didn't receive any input!", :detail => "" }.to_yaml
      return false
    end

    subscription_data = YAML.load(task.input)
```

Sekarang kita memiliki topik ARN, kita dapat mengambil topik dengan membuat instance baru AWS::SNS::Topicdan meneruskannya ARN.

```
    topic = AWS::SNS::Topic.new(subscription_data[:topic_arn])

    if topic.nil?
```

```
        @results = {
          :reason => "Couldn't get SWF topic ARN",
          :detail => "Topic ARN: #{topic.arn}" }.to_yaml
        return false
      end
```

Sekarang, kita akan memeriksa topik untuk melihat apakah pengguna telah mengonfirmasi langganan menggunakan salah satu titik akhir. Kita hanya akan membutuhkan satu titik akhir yang terkonfirmasi untuk menganggap aktivitas ini berhasil.

Topik Amazon SNS memelihara daftar langganan untuk topik tersebut, dan kita dapat memeriksa apakah pengguna telah mengkonfirmasi langganan tertentu dengan memeriksa untuk melihat apakah ARN langganan diatur menjadi apa pun selain `PendingConfirmation`.

```
      # loop until we get some indication that a subscription was confirmed.
      subscription_confirmed = false
      while(!subscription_confirmed)
        topic.subscriptions.each do | sub |
          if subscription_data[sub.protocol.to_sym][:endpoint] == sub.endpoint
            # this is one of the endpoints we're interested in. Is it subscribed?
            if sub.arn != 'PendingConfirmation'
              subscription_data[sub.protocol.to_sym][:subscription_arn] = sub.arn
              puts "Topic subscription confirmed for (#{sub.protocol}:
#{sub.endpoint})"
              @results = subscription_data.to_yaml
              return true
            else
              puts "Topic subscription still pending for (#{sub.protocol}:
#{sub.endpoint})"
            end
          end
        end
```

Jika kita mendapatkan ARN untuk langganan, kita akan menyimpannya dalam data hasil aktivitas, mengonversinya menjadi YAML, dan mengembalikan true dari do`_activity`, yang menandakan bahwa aktivitas berhasil diselesaikan.

Karena menunggu langganan dikonfirmasi mungkin memakan waktu cukup lama, kami sesekali akan memanggil `record_heartbeat` tugas aktivitas. Ini menandakan Amazon SWF bahwa aktivitas masih diproses, dan juga dapat digunakan untuk memberikan pembaruan tentang kemajuan aktivitas (jika Anda melakukan sesuatu, seperti memproses file, yang dapat Anda laporkan kemajuannya).

```
        task.record_heartbeat!(
          { :details => "#{topic.num_subscriptions_confirmed} confirmed,
#{topic.num_subscriptions_pending} pending" })
        # sleep a bit.
        sleep(4.0)
      end
```

Langkah ini mengakhiri loop `while` kita. Jika kita entah bagaimana keluar dari loop tanpa berhasil menyelesaikan proses, kita akan melaporkan kegagalan dan menyelesaikan metode do`_activity`.

```
      if (subscription_confirmed == false)
        @results = {
          :reason => "No subscriptions could be confirmed",
          :detail => "#{topic.num_subscriptions_confirmed} confirmed,
#{topic.num_subscriptions_pending} pending" }.to_yaml
        return false
      end
    end
  end
```

Hal ini mengakhiri penerapan `wait_for_confirmation_activity`. Hanya ada satu aktivitas lagi yang perlu ditentukan: `send_result_activity`.

## Mendefinisikan SendResultActivity

Jika alur kerja telah berkembang sejauh ini, kita telah berhasil menjadikan pengguna berlangganan ke topik Amazon SNS dan pengguna telah mengkonfirmasi langganan.

Aktivitas terakhir kita, `send_result_activity`, mengiri konfirmasi langganan topik yang berhasil kepada pengguna, menggunakan topik yang menjadi langganan pengguna dan titik akhir yang digunakan pengguna untuk mengonfirmasi langganan.

Buat sebuah file baru bernama `send_result_activity.rb` dan siapkan seperti kita menyiapkan semua aktivitas sejauh ini.

```
  require 'yaml'
  require_relative 'basic_activity.rb'

  # **SendResultActivity** sends the result of the activity to the screen, and, if
  # the user successfully registered using SNS, to the user using the SNS contact
```

```
  # information collected.
  class SendResultActivity < BasicActivity

    def initialize
      super('send_result_activity')
    end
```

do_activityMetode kami dimulai dengan cara yang sama, juga, mendapatkan data input dari alur kerja, mengonversinya dari YAMAL, dan kemudian menggunakan topik ARN untuk membuat instance. AWS::SNS::Topic

```
    def do_activity(task)
      if task.input.nil?
        @results = { :reason => "Didn't receive any input!", :detail => "" }
        return false
      end

      input = YAML.load(task.input)

      # get the topic, so we publish a message to it.
      topic = AWS::SNS::Topic.new(input[:topic_arn])

      if topic.nil?
        @results = {
          :reason => "Couldn't get SWF topic",
          :detail => "Topic ARN: #{topic.arn}" }
        return false
      end
```

Setelah kita memiliki topik, kita akan mempublikasikan pesan ke topik (dan meneruskannya ke layar, juga).

```
      @results = "Thanks, you've successfully confirmed registration, and your
workflow is complete!"

      # send the message via SNS, and also print it on the screen.
      topic.publish(@results)
      puts(@results)

      return true
    end
  end
```

Penerbitan ke topik Amazon SNS mengirimkan pesan yang Anda suplai ke semua titik akhir langganan dan yang terkonfirmasi yang ada untuk topik tersebut. Jadi, jika pengguna mengonfirmasi dengandua cara yaitu email dan nomor SMS, pelanggan akan menerima dua pesan konfirmasi, satu di setiap titik akhir.

## Langkah selanjutnya

Langkah ini menyelesaikan penerapan `send_result_activity`. Sekarang, Anda akan mengikat semua aktivitas ini bersama-sama dalam aplikasi aktivitas yang menangani tugas-tugas aktivitas dan dapat meluncurkan aktivitas sebagai respons, di Tutorial Alur Kerja Langganan Bagian 4: Menerapkan Poller Tugas Aktivitas.

# Tutorial Alur Kerja Langganan Bagian 4: Menerapkan Poller Tugas Aktivitas

Di Amazon SWF, tugas aktivitas untuk eksekusi alur kerja yang berjalan muncul di daftar tugas aktivitas, yang disediakan saat Anda menjadwalkan aktivitas di alur kerja.

Kita akan menerapkan poller aktivitas dasar untuk menangani tugas-tugas ini untuk alur kerja kita, dan menggunakannya untuk meluncurkan aktivitas kita ketika Amazon SWF menempatkan tugas pada daftar tugas aktivitas untuk memulai aktivitas.

Untuk memulai, buat file baru bernama `swf_sns_activities.rb`. Kita akan menggunakannya untuk:

- Menginstansiasi kelas aktivitas yang telah kita buat.
- Mendaftarkan setiap aktivitas dengan Amazon SWF.
- Lakukan poll untuk aktivitas dan panggil `do_activity` untuk setiap aktivitas ketika namanya muncul di daftar tugas aktivitas.

Di `swf_sns_activities.rb`, tambahkan pernyataan berikut untuk mengharuskan masing-masing kelas aktivitas yang kita tentukan.

```
require_relative 'get_contact_activity.rb'
require_relative 'subscribe_topic_activity.rb'
require_relative 'wait_for_confirmation_activity.rb'
require_relative 'send_result_activity.rb'
```

Sekarang, kita akan membuat kelas dan menyediakan beberapa kode inisialisasi.

```
class ActivitiesPoller

  def initialize(domain, workflowId)
    @domain = domain
    @workflowId = workflowId
    @activities = {}

    # These are the activities we'll run
    activity_list = [
      GetContactActivity,
      SubscribeTopicActivity,
      WaitForConfirmationActivity,
      SendResultActivity ]

    activity_list.each do | activity_class |
      activity_obj = activity_class.new
      puts "** initialized and registered activity: #{activity_obj.name}"
      # add it to the hash
      @activities[activity_obj.name.to_sym] = activity_obj
    end
  end
```

Selain menyimpan domain dan daftar tugas yang diteruskan, kode ini menginstansiasi masing-masing kelas aktivitas yang kita buat. Karena setiap kelas mendaftarkan aktivitas terkait (lihat `basic_activity.rb` jika Anda perlu meninjau kode tersebut), hal ini sudah cukup untuk mengizinkan Amazon SWF mengetahui semua kegiatan yang akan kita jalankan.

Untuk setiap aktivitas yang diinstansiasi, kita menyimpannya di peta menggunakan nama aktivitas (seperti `get_contact_activity`) sebagai kunci, sehingga kita dapat dengan mudah mencarinya di kode poller aktivitas, yang akan kita tentukan selanjutnya.

Buat sebuah metode baru bernama `poll_for_activities` dan panggil poll pada activity_tasks yang dipegang oleh domain untuk mendapatkan tugas aktivitas.

```
  def poll_for_activities
    @domain.activity_tasks.poll(@workflowId) do | task |
      activity_name = task.activity_type.name
```

Kita bisa mendapatkan nama aktivitas dari anggota activity_type tugas. Selanjutnya, kita akan menggunakan nama aktivitas yang terkait dengan tugas ini untuk mencari kelas untuk menjalankan `do_activity`, meneruskan tugasnya (yang mencakup data input yang harus dikirim ke aktivitas).

```
      # find the task on the activities list, and run it.
      if @activities.key?(activity_name.to_sym)
        activity = @activities[activity_name.to_sym]
        puts "** Starting activity task: #{activity_name}"
        if activity.do_activity(task)
          puts "++ Activity task completed: #{activity_name}"
          task.complete!({ :result => activity.results })
          # if this is the final activity, stop polling.
          if activity_name == 'send_result_activity'
             return true
          end
        else
          puts "-- Activity task failed: #{activity_name}"
          task.fail!(
            { :reason => activity.results[:reason],
              :details => activity.results[:detail] } )
        end
      else
        puts "couldn't find key in @activities list: #{activity_name}"
        puts "contents: #{@activities.keys}"
      end
    end
  end
end
```

Kode hanya menunggu `do_activity` diselesaikan, dan kemudian memanggil baik complete! (selesai!) maupun fail! (gagal!) pada tugas berdasarkan kode yang kembali.

> **Note**
>
> Kode ini keluar dari poller setelah kegiatan akhir diluncurkan, karena telah menyelesaikan misinya dan telah meluncurkan semua kegiatan. Dalam kode Amazon SWF Anda sendiri, jika aktivitas Anda mungkin dijalankan lagi, Anda mungkin ingin menjaga poller aktivitas berjalan tanpa batas waktu.

Itulah akhir dari kode untuk ActivitiesPollerkelas kita, tetapi kita akan menambahkan sedikit lebih banyak kode di akhir file untuk memungkinkan pengguna menjalankannya dari baris perintah.

```
if __FILE__ == $0
  if ARGV.count < 1
    puts "You must supply a task-list name to use!"
    exit
  end
  poller = ActivitiesPoller.new(init_domain, ARGV[0])
  poller.poll_for_activities
  puts "All done!"
end
```

Jika pengguna menjalankan file dari baris perintah (meneruskan daftar tugas aktivitas sebagai argumen pertama), kode ini akan menginstansiasi kelas poller dan memerintahkan kelas untuk melakukan polling untuk aktivitas. Setelah poller selesai (setelah meluncurkan aktivitas akhir), kita cukup mencetak pesan dan keluar.

Itu saja untuk poller aktivitas. Anda hanya perlu menjalankan kode dan melihat cara kerjanya, di Tutorial Alur Kerja Beerlangganan: Menjalankan Alur Kerja.

# Tutorial Alur Kerja Beerlangganan: Menjalankan Alur Kerja

Setelah menyelesaikan penerapan alur kerja, aktivitas, serta poller alur kerja dan aktivitas, Anda siap untuk menjalankan alur kerja.

Jika Anda belum melakukannya, Anda harus memberikan kunci AWS akses Anda dalam `aws-config.txt` file, seperti Mengkonfigurasi Sesi AWS di Bagian 1 tutorial.

Sekarang, buka command line (baris perintah) Anda dan ubah menjadi directory (direktori) di mana file sumber tutorial berada. Anda harus memiliki file-file berikut:

```
.
|-- aws-config.txt
|-- basic_activity.rb
|-- get_contact_activity.rb
|-- send_result_activity.rb
|-- subscribe_topic_activity.rb
|-- swf_sns_activities.rb
|-- swf_sns_workflow.rb
```

```
|-- utils.rb
`-- wait_for_confirmation_activity.rb
```

Sekarang, mulai alur kerja dengan perintah berikut ini.

```
ruby swf_sns_workflow.rb
```

Ini akan memulai alur kerja, dan akan mencetak pesan dengan baris yang dapat Anda salin dan tempel ke jendela baris perintah terpisah (atau bahkan di komputer lain, jika Anda telah menyalin file sumber tutorial ke dalamnya).

```
Amazon SWF Example
------------------

Start the activity worker, preferably in a separate command-line window, with
the following command:

> ruby swf_sns_activities.rb 87097e76-7c0c-41c7-817b-92527bb0ea85-activities

You can copy & paste it if you like, just don't copy the '>' character.

Press return when you're ready...
```

Kode alur kerja akan menunggu Anda untuk memulai poller aktivitas di jendela terpisah.

Buka new command-line window (jendela baris perintah baru), ubah menjadi directory (direktori) di mana file sumber terletak lagi, dan kemudian gunakan perintah yang disediakan oleh file `swf_sns_workflow.rb` untuk memulai poller aktivitas. Sebagai contoh, jika Anda menerima output sebelumnya, Anda akan mengetik (atau menyalian) berikut ini.

```
ruby swf_sns_activities.rb 87097e76-7c0c-41c7-817b-92527bb0ea85-activities
```

Setelah Anda mulai menjalankan poller aktivitas Anda, poller akan mulai menghasilkan informasi tentang pendaftaran kegiatan.

```
** initialized and registered activity: get_contact_activity
** initialized and registered activity: subscribe_topic_activity
** initialized and registered activity: wait_for_confirmation_activity
** initialized and registered activity: send_result_activity
```

Sekarang Anda dapat kembali ke jendela baris perintah asli Anda, dan tekan return (kembali) untuk memulai eksekusi alur kerja Anda. Langkah ini akan mendaftarkan alur kerja dan menjadwalkan aktivitas pertama.

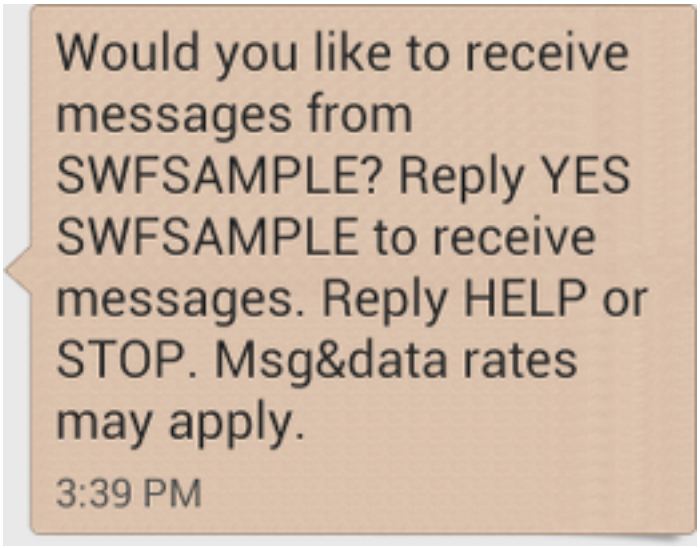
```
Starting workflow execution.
** registered workflow: swf-sns-workflow
** scheduling activity task: get_contact_activity
```

Kembali ke other window (jendela lain), di mana poller aktivitas Anda sedang berjalan. Hasil dari aktivitas pertama yang dijalankan akan ditampilkan, menyediakan prompt bagi Anda untuk memasukkan email atau nomor telepon SMS Anda. Masukkan salah satu, atau kedua, bagian data ini, dan kemudian konfirmasikan entri teks Anda.

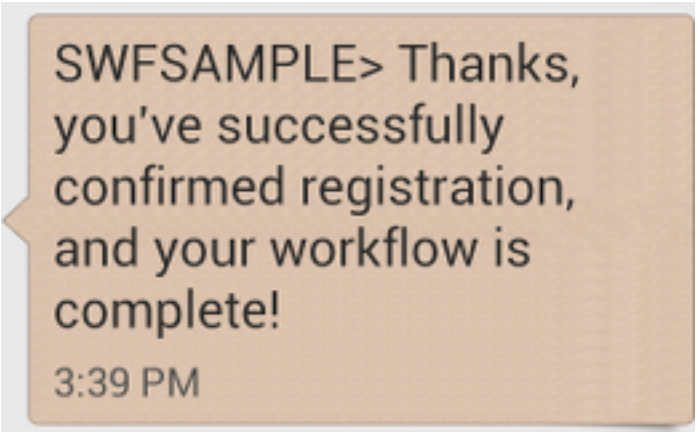
```
activity task received: <AWS::SimpleWorkflow::ActivityTask>
** Starting activity task: get_contact_activity

Please enter either an email address or SMS message (mobile phone) number to
receive Amazon SNS notifications. You can also enter both to use both address types.

If you enter a phone number, it must be able to receive SMS messages, and must
be 11 digits (such as 12065550101 to represent the number 1-206-555-0101).

Email: me@example.com
Phone: 12065550101

You entered:
  email: me@example.com
  phone: 12065550101

Is this correct? (y/n): y
```

> **Note**
>
> Nomor telepon yang disediakan adalah fiktif, dan hanya digunakan untuk tujuan ilustrasi. Gunakan nomor telepon dan alamat email Anda sendiri di sini!

Segera setelah memasukkan informasi ini, Anda akan menerima email atau pesan teks dari Amazon SNS, yang meminta Anda untuk mengkonfirmasi langganan topik Anda. Jika Anda memasukkan nomor SMS, Anda akan melihat sesuatu seperti berikut ini muncul di telepon Anda.




Jika Anda membalas pesan ini dengan YES, Anda akan mendapatkan respons yang kita sediakan di `send_result_activity`.




Sementara semua ini terjadi, apakah Anda melihat apa yang terjadi di jendela baris perintah Anda? Baik poller alur kerja maupun aktivitas telah bekerja keras.

Berikut adalah output dari poller alur kerja.

```
** scheduling activity task: subscribe_topic_activity
** scheduling activity task: wait_for_confirmation_activity
** scheduling activity task: send_result_activity
!! All activities complete! Sending complete_workflow_execution...
```

Berikut adalah output dari poller aktivitas, yang terjadi pada saat yang sama di jendela baris perintah lain.

```
++ Activity task completed: get_contact_activity
** Starting activity task: subscribe_topic_activity
++ Activity task completed: subscribe_topic_activity
** Starting activity task: wait_for_confirmation_activity
Topic subscription still pending for (email: me@example.com)
Topic subscription confirmed for (sms: 12065550101)
++ Activity task completed: wait_for_confirmation_activity
** Starting activity task: send_result_activity
Thanks, you've successfully confirmed registration, and your workflow is complete!
++ Activity task completed: send_result_activity
All done!
```

Selamat, alur kerja Anda selesai, dan begitu juga tutorial ini!

Anda mungkin ingin kembali menjalankan alur kerja untuk melihat cara kerja batas waktu, atau memasukkan data yang berbeda. Ingatlah bahwa begitu Anda berlangganan topik, Anda masih berlangganan sampai Anda berhenti berlangganan. Menjalankan kembali alur kerja sebelum berhenti berlangganan topik mungkin akan menghasilkan keberhasilan otomatis, karena `wait_for_confirmation_activity` akan melihat bahwa langganan Anda sudah dikonfirmasi.

Untuk berhenti berlangganan dari topik Amazon SNS

- Respons secara negatif (kirim `STOP`) ke pesan teks.
- Pilih tautan berhenti berlangganan yang Anda terima di email Anda.

Sekarang Anda siap untuk berlangganan kembali topik tersebut.

## Kemana Saya Pergi Dari Sini?

Tutorial ini telah membahas banyak hal, tetapi masih banyak lagi yang dapat Anda pelajari tentang AWS SDK for Ruby, Amazon SWF, atau Amazon SNS. Untuk informasi selengkapnya dan banyak contoh lainnya, lihat dokumentasi resmi untuk masing-masing:

- Dokumentasi AWS SDK for Ruby
- Dokumentasi Layanan Pemberitahuan Sederhana Amazon
- Dokumentasi Layanan Alur Kerja Sederhana Amazon

# Bekerja di konsol Amazon SWF

Konsol Amazon SWF menyediakan opsi untuk mengonfigurasi, memulai, dan mengelola eksekusi alur kerja.

Dengan konsol Amazon SWF, Anda dapat:

- Mendaftarkan domain alur kerja.
- Daftarkan jenis alur kerja, dan jenis aktivitas.
- Mulai, lihat, beri sinyal, batalkan, hentikan, dan mulai ulang eksekusi alur kerja.

## Mendaftarkan domain

Alur kerja berjalan dalam AWS sumber daya yang disebut domain, yang mengontrol cakupan alur kerja. Akun AWS dapat memiliki beberapa domain, masing-masing dapat berisi beberapa alur kerja, tetapi alur kerja di domain yang berbeda tidak dapat berinteraksi.

Registrasi domain adalah satu-satunya fungsi yang awalnya tersedia di konsol. Setelah setidaknya satu domain terdaftar, Anda dapat melakukan tindakan berikut untuk domain:

- Daftarkan alur kerja dan jenis aktivitas.
- Memulai eksekusi alur kerja.
- Membatalkan, mengakhiri, dan mengirim sinyal untuk menjalankan eksekusi alur kerja.
- Memulai ulang eksekusi alur kerja yang tertutup.

Anda juga dapat melakukan tindakan pengelolaan domain, seperti menghentikan dan membatalkan penggunaan domain.

Setelah menghentikan domain, Anda tidak dapat menggunakannya untuk membuat eksekusi alur kerja baru atau mendaftarkan alur kerja baru. Menghentikan domain juga menghentikan semua aktivitas dan alur kerja yang terdaftar di domain. Eksekusi yang dimulai sebelum domain tidak digunakan lagi terus berjalan.

Setelah membatalkan penggunaan domain yang sebelumnya tidak digunakan lagi, Anda dapat melanjutkan penggunaan domain untuk mendaftarkan jenis alur kerja dan memulai eksekusi alur kerja baru.

Untuk informasi selengkapnya tentang tindakan pengelolaan domain ini, lihat DeprecateDomaindan UndeprecateDomain.

# Mendaftarkan jenis alur kerja

Anda dapat mendaftarkan jenis alur kerja di konsol Amazon SWF setelah Anda mendaftarkan setidaknya satu domain.

Jenis alur kerja adalah seperangkat jenis aktivitas yang melaksanakan tujuan dan berisi logika yang mengoordinasikan aktivitas. Jenis alur kerja mengoordinasikan dan mengelola eksekusi aktivitas yang dapat dijalankan secara asinkron di beberapa perangkat komputasi dan menampilkan metode pemrosesan sekuensial dan paralel.

Untuk mendaftarkan tipe alur kerja Amazon SWF menggunakan konsol

1. Buka domain tempat Anda ingin mendaftarkan alur kerja.
2. Pilih Daftar, lalu pilih Daftar Alur Kerja.
3. Pada halaman Daftar Alur Kerja, masukkan nama Alur Kerja dan versi Alur Kerja. Secara opsional, Anda juga dapat menentukan daftar tugas default yang akan digunakan untuk menjadwalkan tugas keputusan untuk eksekusi alur kerja ini.
4. (Opsional) Pilih Opsi lanjutan untuk menentukan detail berikut untuk alur kerja Anda:
   - Prioritas Tugas Default - Prioritas tugas default untuk ditetapkan ke alur kerja.
   - Eksekusi default mulai menutup batas waktu - Durasi maksimum default untuk eksekusi alur kerja ini.
   - Tugas Default mulai menutup batas waktu - Durasi maksimum tugas keputusan default untuk alur kerja ini.
   - Kebijakan Anak Default — Kebijakan default yang digunakan untuk eksekusi alur kerja anak.
   - Peran Lambda default - Peran IAM default yang melekat pada alur kerja ini.
5. Pilih Daftarkan alur kerja.

# Mendaftarkan jenis aktivitas

Aktivitas adalah tugas yang Anda ingin jenis alur kerja Anda untuk mengoordinasikan dan mengeksekusi (misalnya: memverifikasi pesanan pelanggan, membebankan kartu kredit, dll.). Urutan kegiatan yang dilakukan ditentukan oleh logika koordinasi tipe alur kerja.

Anda dapat mendaftarkan jenis aktivitas setelah setidaknya satu domain terdaftar.

Untuk mendaftarkan tipe aktivitas Amazon SWF menggunakan konsol

1. Buka domain tempat Anda ingin mendaftarkan aktivitas.
2. Pilih Register, lalu pilih Register Activity.
3. Pada halaman Daftarkan aktivitas, masukkan nama Aktivitas dan versi Aktivitas. Secara opsional, Anda juga dapat menentukan daftar tugas default yang akan digunakan untuk menjadwalkan tugas aktivitas ini.
4. (Opsional) Pilih Opsi lanjutan untuk menentukan detail berikut untuk aktivitas Anda:
   - Prioritas Tugas Default - Prioritas tugas default untuk ditetapkan ke aktivitas.
   - Jadwal tugas default untuk memulai batas waktu — Durasi maksimum default yang dapat ditunggu oleh tugas aktivitas ini sebelum ditetapkan ke pekerja.
   - Tugas Default mulai menutup batas waktu — Durasi maksimum default yang dapat diambil pekerja untuk memproses tugas aktivitas ini.
   - Jadwal tugas default untuk menutup batas waktu — Durasi maksimum default untuk tugas aktivitas ini.
   - Batas waktu detak jantung tugas default — Waktu maksimum default sebelum pekerja memproses tugas jenis ini harus melaporkan kemajuan dengan menelepon. RecordActivityTaskHeartbeat
5. Pilih Daftarkan aktivitas.

# Memulai alur kerja

Anda dapat memulai eksekusi alur kerja dari konsol Amazon SWF. Anda tidak dapat memulai eksekusi alur kerja sampai Anda telah mendaftarkan setidaknya satu alur kerja.

## Untuk memulai eksekusi alur kerja menggunakan konsol tersebut

1. Buka konsol Amazon SWF, dan di panel navigasi kiri, pilih Domain.
2. Di bawah nama domain, pilih Alur kerja.
3. Pada halaman Alur Kerja, pilih alur kerja yang ingin Anda jalankan.
4. Pilih Mulai Eksekusi.

5. Pada halaman Mulai eksekusi, masukkan nama Alur Kerja dan ID Eksekusi untuk mengidentifikasi eksekusi Anda dengan nama. Secara opsional, Anda juga dapat menentukan daftar Tugas yang akan digunakan untuk tugas keputusan yang dihasilkan untuk eksekusi alur kerja ini.
6. (Opsional) Pilih Opsi lanjutan untuk menentukan detail berikut untuk eksekusi alur kerja Anda:
   - Prioritas tugas - Prioritas tugas yang digunakan untuk eksekusi alur kerja ini.
   - Eksekusi mulai menutup batas waktu - Total durasi untuk eksekusi alur kerja ini.
   - Tugas mulai menutup batas waktu - Durasi maksimum tugas keputusan untuk eksekusi alur kerja ini.
   - Kebijakan anak — Kebijakan yang digunakan untuk eksekusi alur kerja anak dari eksekusi alur kerja ini jika dihentikan, dengan memanggil TerminateWorkflowExecutiontindakan secara eksplisit atau karena batas waktu kedaluwarsa.
   - Peran Lambda - Peran IAM untuk dilampirkan ke eksekusi alur kerja ini.
7. Pilih Mulai Eksekusi.

# Mengelola eksekusi alur kerja

Anda dapat memfilter eksekusi alur kerja berdasarkan nama, status, ID, dan tag. Anda dapat mengirim sinyal dengan input ke eksekusi alur kerja aktif. Jika Anda perlu membatalkan atau mengakhiri alur kerja, Anda dapat menggunakan opsi Try-cancel. Pembatalan lebih disukai daripada menghentikan eksekusi alur kerja karena pembatalan memberi alur kerja kesempatan untuk melakukan tugas pembersihan dan kemudian menutup dengan benar.

Di konsol, Anda dapat mengelola eksekusi alur kerja yang sedang berjalan and/or tertutup.

Untuk mengelola eksekusi alur kerja Anda

1. Buka domain untuk mengelola eksekusi alur kerjanya.
2. Pilih Temukan Eksekusi.
3. Pada halaman eksekusi alur kerja, pilih Filter eksekusi berdasarkan properti, dan kemudian di bawah Properti pilih salah satu filter berikut:

| Pilih | Untuk menerapkan filter ini |
| --- | --- |
| Alur kerja | Pilih filter ini untuk mencantumkan eksekusi alur kerja tertentu. Misalnya, untuk melihat eksekusi`fiction-books-order-workflow` , lakukan hal berikut:<br>1. Pilih Alur Kerja.<br>2. Di bawah Operator, pilih Equals.<br>3. Di bawah Alur kerja, pilih fiction-books-order-workflow.<br>4. (Opsional) Pilih Hapus filter untuk menghapus filter dan memulai pencarian baru untuk eksekusi. |
| Status | Pilih filter ini untuk mencantumkan eksekusi dengan status tertentu. Misalnya, untuk melihat eksekusi dengan status Gagal, lakukan hal berikut:<br>1. Pilih Status.<br>2. Di bawah Operator, pilih Equals.<br>3. Di bawah Status, pilih Gagal.<br>4. (Opsional) Pilih Hapus filter untuk menghapus filter dan memulai pencarian baru untuk eksekusi. |
| ID Eksekusi | Pilih filter ini untuk melihat eksekusi alur kerja berdasarkan ID-nya. Misalnya, untuk melihat eksekusi dengan ID`fiction-books-order-category1` , lakukan hal berikut:<br>1. Pilih ID Eksekusi.<br>2. Di bawah Operator, pilih Equals.<br>3. Di bawah Eksekusi IDs, pilih fiction-books-order-category1.<br>4. (Opsional) Pilih Hapus filter untuk menghapus filter dan memulai pencarian baru untuk eksekusi. |

| Pilih | Untuk menerapkan filter ini |
| --- | --- |
| Tag | Pilih filter ini untuk mencantumkan eksekusi dengan tag tertentu. Misalnya, untuk melihat eksekusi dengan status`purchaseOrder` , lakukan hal berikut:<br>1. Pilih Tag.<br>2. Di bawah Operator, pilih Equals.<br>3. Di bawah Tag, pilih PurchaseOrder.<br>4. (Opsional) Pilih Hapus filter untuk menghapus filter dan memulai pencarian baru untuk eksekusi. |

4. (Opsional) Setelah menerapkan filter yang diperlukan untuk mencantumkan eksekusi alur kerja, Anda dapat melakukan operasi berikut ke eksekusi Aktif:

   - Sinyal - Gunakan opsi ini untuk mengirim data tambahan eksekusi alur kerja yang sedang berjalan. Untuk melakukannya:
     1. Pilih eksekusi yang ingin Anda kirimi data tambahan.
     2. Pilih Sinyal, lalu tentukan data di kotak dialog Eksekusi sinyal.
     3. Pilih Signal.
   - Coba batalkan - Gunakan opsi ini untuk mencoba membatalkan eksekusi alur kerja. Lebih baik untuk membatalkan eksekusi alur kerja daripada mengakhirinya. Pembatalan menyediakan eksekusi alur kerja kesempatan untuk melakukan tugas pembersihan apa pun dan kemudian menutup dengan benar.
     1. Pilih eksekusi yang ingin Anda batalkan.
     2. Pilih coba-batal.
   - Mengakhiri - Gunakan opsi ini untuk mengakhiri eksekusi alur kerja. Perhatikan bahwa lebih baik untuk membatalkan eksekusi alur kerja daripada mengakhirinya.
     1. Pilih eksekusi yang ingin Anda hentikan.
     2. Untuk kebijakan Anak, pastikan Terminate dipilih.
     3. (Opsional) Tentukan Alasan dan Detail untuk mengakhiri eksekusi.
     4. Pilih Akhiri.

5. (Opsional) Re-run - Gunakan opsi ini untuk menjalankan kembali eksekusi alur kerja tertutup.

1. Dalam daftar eksekusi alur kerja, pilih eksekusi tertutup untuk dijalankan kembali. Ketika Anda memilih eksekusi tertutup, tombol Re-run menjadi diaktifkan. Pilih Re-run.
2. Pada halaman Re-run eksekusi, tentukan detail untuk eksekusi alur kerja seperti yang disebutkan dalam. Memulai alur kerja

# Konsep alur kerja dasar di Amazon SWF

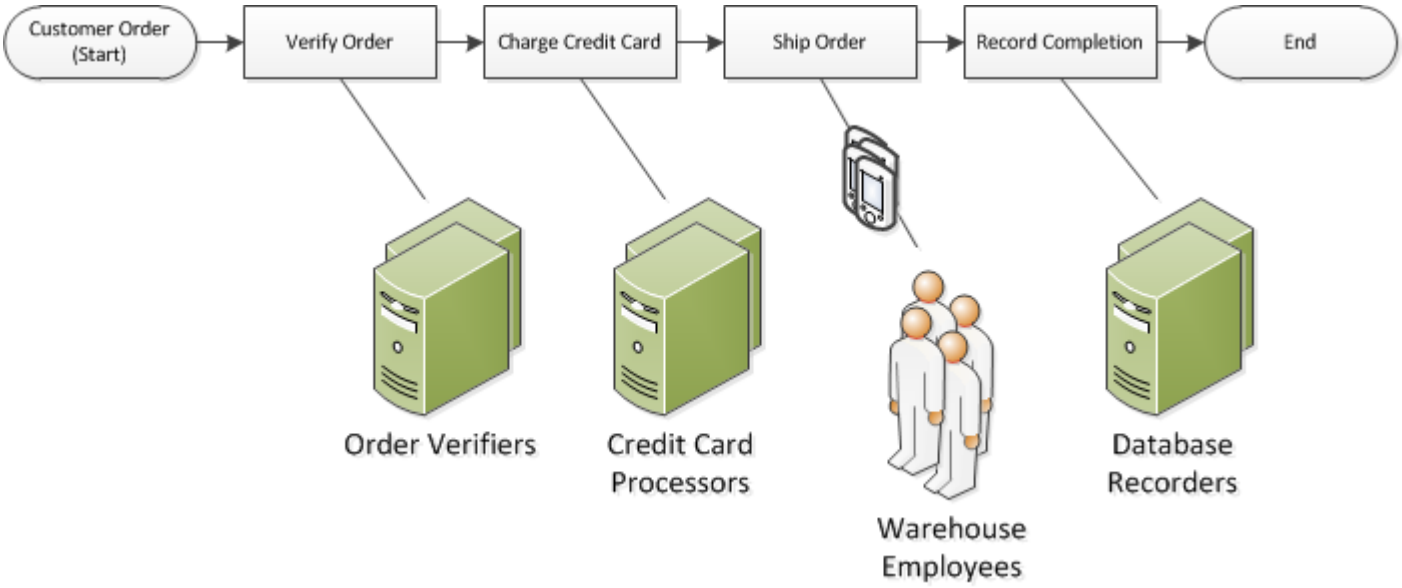
> **Note**
>
> Konsep dalam bab ini memberikan gambaran umum tentang Amazon Simple Workflow Service dan menjelaskannya fitur utama. Jika Anda mencari contoh, lihatBekerja dengan Amazon SWF APIs.

Dengan menggunakan Amazon Simple Workflow Service (Amazon SWF), Anda dapat menggunakan aplikasi asinkron terdistribusi sebagai workflows (alur kerja). Alur kerja mengkoordinasikan dan mengelola pelaksanaan aktivitas yang dapat dijalankan secara asinkron di beberapa perangkat komputasi dan yang dapat menampilkan pemrosesan sekuensial dan paralel.

Saat merancang alur kerja, Anda menganalisis aplikasi Anda untuk mengidentifikasi tasks (tugas) komponennya. Di Amazon SWF, tugas ini diwakili oleh activities (aktivitas). Urutan aktivitas yang dilakukan ditentukan oleh logika koordinasi alur kerja.

Contoh alur kerja untuk aplikasi e-commerce

Gambar berikut menunjukkan alur kerja pemrosesan pesanan e-commerce yang melibatkan orang dan proses otomatis:




Alur kerja aplikasi e-commerce dimulai ketika pelanggan melakukan pemesanan, dan mencakup empat tugas:

1. Verifikasi pesanan.
2. Jika pesanan valid, tagih biayanya ke pelanggan.
3. Jika pembayaran dilakukan, kirimkan pesanan.
4. Jika pesanan dikirim, simpan detail pesanan.

Tugas dalam alur kerja ini berurutan: pesanan harus diverifikasi sebelum kartu kredit dapat ditagih; kartu kredit harus berhasil ditagih sebelum pesanan dapat dikirim; dan pesanan harus dikirim sebelum dapat disimpan. Meski begitu, karena Amazon SWF mendukung proses terdistribusi, tugas ini dapat dilakukan di lokasi yang berbeda. Jika tugas bersifat terprogram, tugas tersebut dapat ditulis dalam bahasa pemrograman yang berbeda atau menggunakan alat yang berbeda.

Selain pemrosesan tugas secara berurutan, Amazon SWF juga mendukung alur kerja dengan pemrosesan tugas secara paralel. Tugas paralel dilakukan pada saat yang sama, dan dapat dilakukan secara independen oleh aplikasi yang berbeda atau pekerja manusia. Alur kerja Anda membuat keputusan tentang cara melanjutkan setelah satu atau lebih tugas paralel telah yang telah selesai.

Konsep tambahan

- Membuat alur kerja di Amazon SWF
- Menjalankan alur kerja di Amazon SWF
- Riwayat alur kerja di Amazon SWF
- Pengidentifikasi objek di Amazon SWF
- Domain di Amazon SWF
- Aktor di Amazon SWF
- Tugas di Amazon SWF
- Daftar tugas di Amazon SWF
- Penutupan eksekusi alur kerja di Amazon SWF
- Siklus hidup alur kerja Amazon SWF
- Polling untuk tugas di Amazon SWF

# Membuat alur kerja di Amazon SWF

Membuat alur kerja berurutan dasar melibatkan tahapan berikut.

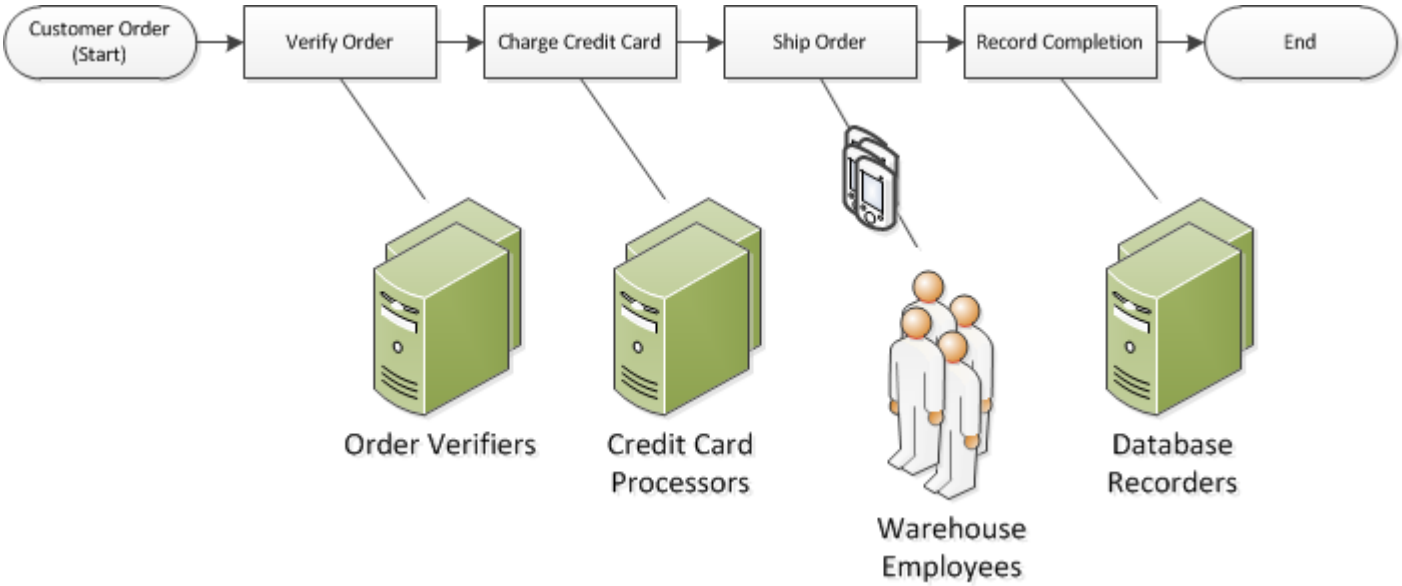
- Pemodelan alur kerja, mendaftarkan tipenya, dan tipe aktivitasnya
- Mengembangkan dan meluncurkan aktivitas pekerja yang melakukan tugas aktivitas
- Mengembangkan dan meluncurkan penentu yang menggunakan riwayat alur kerja untuk menentukan apa yang harus dilakukan selanjutnya
- Mengembangkan dan meluncurkan pemula alur kerja, yaitu aplikasi yang memulai eksekusi alur kerja

## Pemodelan Alur Kerja Anda dan Aktivitasnya

Untuk menggunakan Amazon SWF, buat model langkah-langkah logis dalam aplikasi Anda sebagai aktivitas. Aktivitas mewakili satu langkah logis atau tugas dalam alur kerja Anda. Misalnya, otorisasi kartu kredit adalah aktivitas yang melibatkan pemberian nomor kartu kredit dan informasi lainnya, serta menerima kode persetujuan atau pesan bahwa kartu ditolak.

Selain mendefinisikan aktivitas, Anda juga perlu menentukan logika koordinasi yang menangani titik keputusan. Misalnya, logika koordinasi mungkin menjadwalkan aktivitas tindak lanjut yang berbeda tergantung pada apakah kartu kredit disetujui atau ditolak.

Gambar berikut menunjukkan contoh alur kerja pesanan pelanggan berurutan dengan empat aktivitas (Verifikasi Pesanan, Baya Kartu Kredit, Pengiriman Pesanan, dan Catatan Penyelesaian).



# Menjalankan alur kerja di Amazon SWF

Setelah logika koordinasi dan aktivitas telah dirancang, Anda mendaftar komponen ini sebagai alur kerja dan jenis aktivitas dengan Amazon SWF. Selama pendaftaran, Anda menentukan nama, versi, dan nilai konfigurasi default untuk setiap jenis.

Hanya alur kerja dan jenis aktivitas terdaftar yang dapat digunakan dengan Amazon SWF. Dalam contoh e-commerce, Anda akan mendaftarkan jenis CustomerOrder alur kerja dan VerifyOrder,, ChargeCreditCard ShipOrder, dan jenis RecordCompletion aktivitas.

Setelah mendaftarkan jenis alur kerja Anda, Anda dapat menjalankannya sesering yang Anda suka. workflow execution (eksekusi alur kerja) adalah instans yang berjalan di alur kerja.

Eksekusi alur kerja dapat dimulai dengan proses atau aplikasi apa pun, bahkan eksekusi alur kerja lainnya. Dalam contoh perdagangan elektronik, eksekusi alur kerja baru dimulai dengan setiap pesanan pelanggan. Jenis aplikasi yang memulai alur kerja tergantung pada bagaimana pelanggan melakukan pemesanan. Alur kerja dapat dimulai oleh situs web atau aplikasi seluler atau oleh perwakilan layanan pelanggan menggunakan aplikasi perusahaan internal.

Dengan Amazon SWF, Anda dapat menghubungkan pengidentifikasi–-disebut `workflowId`–- dengan eksekusi alur kerja Anda, sehingga Anda dapat mengintegrasikan pengidentifikasi bisnis yang ada ke dalam alur kerja Anda. Dalam contoh perdagangan elektronik, setiap eksekusi alur kerja dapat diidentifikasi menggunakan nomor faktur pelanggan.

Selain pengidentifikasi yang Anda berikan, Amazon SWF menghubungkan pengidentifikasi unik yang dihasilkan sistem —`runId`—dengan setiap eksekusi alur kerja. Amazon SWF hanya memungkinkan satu eksekusi alur kerja dengan pengidentifikasi ini untuk dijalankan pada waktu tertentu; meskipun Anda dapat memiliki beberapa eksekusi alur kerja dari jenis alur kerja yang sama, setiap eksekusi alur kerja memiliki `runId` yang berbeda.

# Riwayat alur kerja di Amazon SWF

Amazon SWF mencatat kemajuan setiap eksekusi alur kerja dalam riwayat alur kerja - catatan terperinci, lengkap, dan konsisten dari setiap peristiwa yang terjadi sejak eksekusi alur kerja dimulai.

Peristiwa mewakili perubahan diskrit dalam status eksekusi alur kerja Anda, seperti aktivitas baru yang dijadwalkan atau aktivitas yang sedang berjalan diselesaikan. Riwayat alur kerja berisi setiap kejadian yang menyebabkan status eksekusi eksekusi alur kerja berubah, seperti aktivitas terjadwal dan selesai, batas waktu tugas, dan sinyal.

Operasi yang tidak mengubah status eksekusi alur kerja biasanya tidak muncul dalam riwayat alur kerja. Misalnya, riwayat alur kerja tidak menampilkan upaya pemilihan atau penggunaan operasi visibilitas.

Riwayat alur kerja memiliki beberapa manfaat utama:

- Aplikasi dapat bersifat stateless, karena semua informasi tentang eksekusi alur kerja disimpan dalam riwayat alur kerjanya.
- Untuk setiap eksekusi alur kerja, riwayat menyediakan catatan aktivitas yang dijadwalkan, statusnya saat ini, dan hasilnya. Eksekusi alur kerja menggunakan informasi ini untuk menentukan langkah selanjutnya.
- Riwayat menyediakan jejak audit terperinci yang dapat digunakan untuk memantau eksekusi alur kerja yang berjalan dan memverifikasi eksekusi alur kerja yang telah selesai.

Berikut ini adalah tampilan konseptual dari riwayat alur kerja perdagangan elektronik:

```
Invoice0001

Start Workflow Execution

Schedule Verify Order
Start Verify Order Activity
Complete Verify Order Activity

Schedule Charge Credit Card
Start Charge Credit Card Activity
Complete Charge Credit Card Activity

Schedule Ship Order
Start Ship Order Activity
```

Dalam contoh sebelumnya, pesanan sedang menunggu untuk dikirim. Pada contoh berikut, pesanan selesai. Karena riwayat alur kerja bersifat kumulatif, kejadian yang lebih baru ditambahkan:

```
Invoice0001

Start Workflow Execution

Schedule Verify Order
Start Verify Order Activity
```

```
Complete Verify Order Activity

Schedule Charge Credit Card
Start Charge Credit Card Activity
Complete Charge Credit Card Activity

Schedule Ship Order
Start Ship Order Activity

Complete Ship Order Activity

Schedule Record Order Completion
Start Record Order Completion Activity
Complete Record Order Completion Activity

Close Workflow
```

Secara terprogram, peristiwa dalam riwayat eksekusi alur kerja direpresentasikan sebagai JavaScript objek Object Notation (JSON). Riwayatnya merupakan array JSON dari objek-objek ini. Setiap kejadian memiliki:

- Tipe, seperti WorkflowExecutionStartedatau ActivityTaskCompleted
- Sebuah tanda waktu dalam format waktu Unix
- ID unik yang mengidentifikasi kejadian

Selain itu, setiap jenis kejadian memiliki seperangkat atribut deskriptif yang berbeda yang sesuai dengan jenis tersebut. Misalnya, `ActivityTaskCompleted` acara memiliki atribut yang berisi IDs untuk peristiwa yang sesuai dengan waktu tugas aktivitas dijadwalkan dan kapan dimulai, serta atribut yang menyimpan data hasil.

Anda dapat memperoleh salinan status saat ini dari riwayat eksekusi alur kerja dengan menggunakan GetWorkflowExecutionHistorytindakan. Selain itu, sebagai bagian dari interaksi antara Amazon SWF dan pengambil keputusan untuk alur kerja Anda, pengambil keputusan secara berkala menerima salinan riwayat.

Di bawah ini adalah bagian dari contoh riwayat eksekusi alur kerja dalam format JSON.

```
[  {
      "eventId": 11,
      "eventTimestamp": 1326671603.102,
```

```
      "eventType": "WorkflowExecutionTimedOut",
      "workflowExecutionTimedOutEventAttributes": {
         "childPolicy": "TERMINATE",
         "timeoutType": "START_TO_CLOSE"
      }
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 10,
      "eventTimestamp": 1326670566.124,
      "eventType": "DecisionTaskScheduled"
   }, {
      "activityTaskTimedOutEventAttributes": {
         "details": "Waiting for confirmation",
         "scheduledEventId": 8,
         "startedEventId": 0,
         "timeoutType": "SCHEDULE_TO_START"
      },
      "eventId": 9,
      "eventTimestamp": 1326670566.124,
      "eventType": "ActivityTaskTimedOut"
   }, {
      "activityTaskScheduledEventAttributes": {
         "activityId": "verification-27",
         "activityType": {
            "name": "activityVerify",
            "version": "1.0"
         },
         "control": "digital music",
         "decisionTaskCompletedEventId": 7,
         "heartbeatTimeout": "120",
         "input": "5634-0056-4367-0923,12/12,437",
         "scheduleToCloseTimeout": "900",
         "scheduleToStartTimeout": "300",
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 8,
```

```
      "eventTimestamp": 1326670266.115,
      "eventType": "ActivityTaskScheduled"
   }, {
      "decisionTaskCompletedEventAttributes": {
         "executionContext": "Black Friday",
         "scheduledEventId": 5,
         "startedEventId": 6
      },
      "eventId": 7,
      "eventTimestamp": 1326670266.103,
      "eventType": "DecisionTaskCompleted"
   }, {
      "decisionTaskStartedEventAttributes": {
         "identity": "Decider01",
         "scheduledEventId": 5
      },
      "eventId": 6,
      "eventTimestamp": 1326670161.497,
      "eventType": "DecisionTaskStarted"
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 5,
      "eventTimestamp": 1326668752.66,
      "eventType": "DecisionTaskScheduled"
   }, {
      "decisionTaskTimedOutEventAttributes": {
         "scheduledEventId": 2,
         "startedEventId": 3,
         "timeoutType": "START_TO_CLOSE"
      },
      "eventId": 4,
      "eventTimestamp": 1326668752.66,
      "eventType": "DecisionTaskTimedOut"
   }, {
      "decisionTaskStartedEventAttributes": {
         "identity": "Decider01",
         "scheduledEventId": 2
      },
      "eventId": 3,
```

```
      "eventTimestamp": 1326668152.648,
      "eventType": "DecisionTaskStarted"
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 2,
      "eventTimestamp": 1326668003.094,
      "eventType": "DecisionTaskScheduled"
   }
]
```

Untuk daftar detail berbagai jenis peristiwa yang dapat muncul dalam riwayat eksekusi alur kerja, lihat tipe HistoryEventdata di Referensi API Layanan Alur Kerja Amazon Sederhana.

Amazon SWF menyimpan riwayat lengkap dari semua eksekusi alur kerja selama beberapa hari yang dapat dikonfigurasi setelah eksekusi ditutup. Periode ini, yang dikenal sebagai periode penyimpanan riwayat alur kerja, ditentukan saat Anda mendaftarkan Domain untuk alur kerja Anda. Domain dibahas lebih terperinci nanti di bagian ini.

## Pengidentifikasi objek di Amazon SWF

Daftar berikut menjelaskan bagaimana objek Amazon SWF, seperti eksekusi alur kerja, diidentifikasi secara unik.

- Workflow Type (Jenis Alur Kerja) – Jenis alur kerja terdaftar diidentifikasi oleh domain, nama, dan versinya. Jenis alur kerja ditentukan dalam panggilan ke `RegisterWorkflowType`.
- Activity Type (Jenis Aktivitas) – Jenis aktivitas terdaftar diidentifikasi berdasarkan domain, nama, dan versinya. Jenis aktivitas ditentukan dalam panggilan ke `RegisterActivityType`.
- Decision Tasks and Activity Tasks (Tugas Keputusan dan Tugas Aktivitas) – Setiap tugas keputusan dan tugas aktivitas diidentifikasi oleh token tugas yang unik. Token tugas dibuat oleh Amazon SWF dan dikembalikan dengan informasi lain tentang tugas dalam respons dari `PollForDecisionTask` atau `PollForActivityTask`. Meskipun token paling sering digunakan oleh proses yang menerima tugas, proses tersebut dapat meneruskan token ke proses lain, yang kemudian dapat melaporkan penyelesaian atau kegagalan tugas.

- Workflow Execution (Eksekusi Alur Kerja) – Eksekusi tunggal alur kerja diidentifikasi oleh domain, ID alur kerja, dan ID proses. Dua yang pertama adalah parameter yang diteruskan ke StartWorkflowExecution. ID proses dikembalikan oleh `StartWorkflowExecution`.

# Domain di Amazon SWF

Alur kerja berjalan dalam AWS sumber daya yang disebut domain yang menyediakan cara untuk melingkupi sumber daya Amazon SWF dalam akun Anda. AWS Semua komponen alur kerja, seperti jenis alur kerja dan jenis aktivitas, harus ditentukan untuk berada di domain.

Akun AWS dapat memiliki beberapa domain, masing-masing dapat berisi beberapa alur kerja, tetapi alur kerja di domain yang berbeda tidak dapat berinteraksi.

Saat menyiapkan alur kerja baru, sebelum Anda menyiapkan komponen alur kerja lainnya, Anda perlu mendaftarkan domain jika belum melakukannya.

Saat Anda mendaftarkan domain, tentukan workflow history retention period (periode retensi riwayat alur kerja). Periode retensi adalah lamanya waktu Amazon SWF akan terus menyimpan informasi tentang eksekusi alur kerja setelah eksekusi alur kerja selesai.

Registrasi domain adalah satu-satunya fungsi yang awalnya tersedia di konsol. Setelah setidaknya satu domain terdaftar, Anda dapat melakukan tindakan berikut untuk domain:

- Daftarkan alur kerja dan jenis aktivitas.
- Memulai eksekusi alur kerja.
- Membatalkan, mengakhiri, dan mengirim sinyal untuk menjalankan eksekusi alur kerja.
- Memulai ulang eksekusi alur kerja yang tertutup.

Anda juga dapat melakukan tindakan pengelolaan domain, seperti menghentikan dan membatalkan penggunaan domain.

Setelah menghentikan domain, Anda tidak dapat menggunakannya untuk membuat eksekusi alur kerja baru atau mendaftarkan alur kerja baru. Menghentikan domain juga menghentikan semua aktivitas dan alur kerja yang terdaftar di domain. Eksekusi yang dimulai sebelum domain tidak digunakan lagi terus berjalan.

Setelah membatalkan penggunaan domain yang sebelumnya tidak digunakan lagi, Anda dapat melanjutkan penggunaan domain untuk mendaftarkan jenis alur kerja dan memulai eksekusi alur kerja baru.

Untuk informasi selengkapnya tentang tindakan pengelolaan domain ini, lihat DeprecateDomaindan UndeprecateDomain.

# Aktor di Amazon SWF

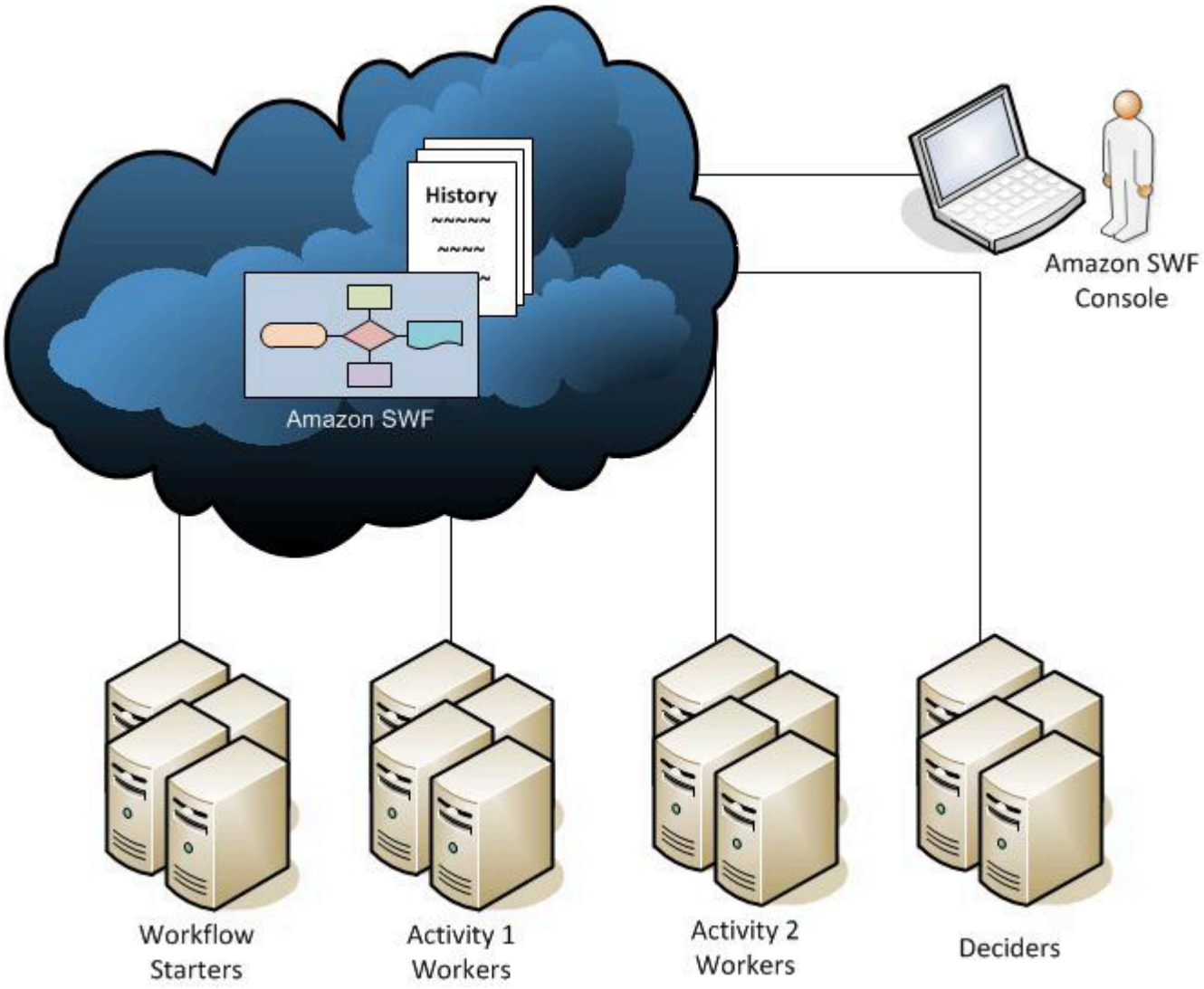
Topik

- Apa yang dimaksud dengan Aktor di Amazon SWF?
- Pemulai Alur Kerja
- Pemecah Keputusan
- Pekerja Aktivitas
- Data Exchange Antar Aktor

## Apa yang dimaksud dengan Aktor di Amazon SWF?

Selama operasinya, Amazon SWF berinteraksi dengan sejumlah jenis actors (aktor) terprogram yang berbeda. Aktor merupakan workflow starters (pemulai alur kerja), deciders (pengambil keputusan), atau activity workers (pekerja aktivitas). Aktor-aktor ini berkomunikasi dengan Amazon SWF melalui API-nya. Anda dapat mengembangkan aktor-aktor ini dalam bahasa pemrograman apa pun.

Diagram berikut menunjukkan arsitektur Amazon SWF, termasuk Amazon SWF dan aktornya.




## Pemulai Alur Kerja

Pemulai alur kerja merupakan aplikasi yang dapat memulai eksekusi alur kerja. Dalam contoh perdagangan elektronik, satu pemulai alur kerja bisa menjadi situs di mana pelanggan menempatkan pesanan. Pemulai alur kerja lainnya berupa aplikasi seluler atau sistem yang digunakan oleh perwakilan layanan pelanggan untuk melakukan pemesanan atas nama pelanggan.

## Pemecah Keputusan

Pengambil keputusan merupakan implementasi dari logika koordinasi alur kerja. Pengambil keputusan mengontrol aliran tugas aktivitas dalam eksekusi alur kerja. Setiap kali perubahan terjadi selama eksekusi alur kerja, seperti penyelesaian tugas, tugas keputusan termasuk seluruh riwayat alur kerja akan diteruskan ke pengambil keputusan. Saat menerima tugas keputusan dari Amazon

SWF, pengambil keputusan menganalisis riwayat eksekusi alur kerja untuk menentukan langkah berikutnya yang sesuai dalam eksekusi alur kerja. Pengambik keputusan mengkomunikasikan langkah-langkah ini kembali ke Amazon SWF menggunakan decisions (keputusan). Keputusan merupakan jenis data Amazon SWF yang dapat mewakili berbagai tindakan selanjutnya. Untuk daftar kemungkinan keputusan, buka Decision (Keputusan) di Referensi API Amazon Simple Workflow Service.

Berikut adalah contoh keputusan dalam format JSON, format yang ditransmisikan ke Amazon SWF. Keputusan ini jadwal tugas aktivitas baru.

```
{
   "decisionType" : "ScheduleActivityTask",
   "scheduleActivityTaskDecisionAttributes" : {
      "activityType" : {
         "name" : "activityVerify",
         "version" : "1.0"
      },
      "activityId" : "verification-27",
      "control" : "digital music",
      "input" : "5634-0056-4367-0923,12/12,437",
      "scheduleToCloseTimeout" : "900",
      "taskList" : {
         "name": "specialTaskList"
      },
      "scheduleToStartTimeout" : "300",
      "startToCloseTimeout" : "600",
      "heartbeatTimeout" : "120"
   }
}
```

Pengambil keputusan menerima tugas keputusan ketika eksekusi alur kerja dimulai dan setiap kali perubahan status terjadi dalam eksekusi alur kerja. Pengambil keputusan terus melanjutkan eksekusi alur kerja dengan menerima tugas keputusan dan menanggapi Amazon SWF dengan lebih banyak keputusan sampai pengambil keputusan menentukan bahwa eksekusi alur kerja selesai. Kemudian menanggapi dengan keputusan untuk menutup eksekusi alur kerja. Setelah eksekusi alur kerja ditutup, Amazon SWF tidak akan menjadwalkan tugas tambahan untuk eksekusi tersebut.

Dalam contoh perdagangan elektronik, pengambil keputusan menentukan apakah setiap langkah dilakukan dengan benar, lalu menjadwalkan langkah berikutnya atau mengelola kondisi kesalahan apa pun.

Pengambil keputusan mewakili satu proses komputer atau utas. Beberapa pengambil keputusan dapat memproses tugas untuk jenis alur kerja yang sama.

## Pekerja Aktivitas

Seorang pekerja aktivitas merupakan proses atau utas yang melakukan activity tasks (tugas aktivitas) yang merupakan bagian dari alur kerja Anda. Tugas aktivitas merupakan salah satu tugas yang Anda identifikasi dalam aplikasi Anda.

Untuk menggunakan tugas aktivitas dalam alur kerja, Anda harus mendaftarkannya menggunakan konsol Amazon SWF atau RegisterActivityTypetindakan.

Setiap pekerja aktivitas memilih Amazon SWF untuk tugas baru yang sesuai untuk dilakukan oleh pekerja aktivitas tersebut; tugas tertentu hanya dapat dilakukan oleh pekerja aktivitas tertentu. Setelah menerima tugas, pekerja aktivitas memproses tugas hingga selesai dan kemudian melaporkannya ke Amazon SWF bahwa tugas telah selesai dan memberikan hasilnya. Pekerja aktivitas kemudian memilih tugas baru. Para pekerja aktivitas yang terhubung dengan eksekusi alur kerja terus melakukan cara ini, yaitu memproses tugas hingga eksekusi alur kerja tersebut selesai dengan sendirinya. Dalam contoh perdagangan elektronik, pekerja aktivitas merupakan proses independen dan aplikasi yang digunakan oleh orang-orang, seperti pemroses kartu kredit dan karyawan gudang, yang melakukan langkah-langkah individual dalam proses.

Pekerja aktivitas mewakili satu proses komputer (atau utas). Beberapa pekerja aktivitas dapat memproses tugas dari jenis aktivitas yang sama.

## Data Exchange Antar Aktor

Data input dapat diberikan ke eksekusi alur kerja saat dimulai. Demikian pula, data input dapat diberikan kepada pekerja aktivitas saat mereka menjadwalkan tugas aktivitas. Ketika tugas aktivitas selesai, pekerja aktivitas dapat mengembalikan hasilnya ke Amazon SWF. Demikian pula, pengambil keputusan dapat melaporkan hasil eksekusi alur kerja ketika eksekusi selesai. Setiap aktor dapat mengirim data ke, dan menerima data dari, Amazon SWF melalui string, yang bentuknya ditentukan oleh pengguna. Bergantung pada ukuran dan sensitivitas data, Anda dapat meneruskan data secara langsung atau meneruskan petunjuk ke data yang disimpan pada sistem atau layanan lain (seperti Amazon S3 atau DynamoDB). Baik data yang diteruskan secara langsung dan petunjuk ke penyimpanan data lain dicatat dalam riwayat eksekusi alur kerja; Namun, Amazon SWF tidak menyalin atau menyimpan data apa pun dari penyimpanan eksternal sebagai bagian dari riwayat.

Karena Amazon SWF mempertahankan status eksekusi lengkap dari setiap eksekusi alur kerja, termasuk input dan hasil tugas, semua aktor bisa menjadi stateless. Sehingga, pemrosesan alur kerja

sangat terukur. Seiring bertambahnya beban pada sistem Anda, Anda cukup menambahkan lebih banyak aktor untuk meningkatkan kapasitas.

# Tugas di Amazon SWF

Amazon SWF berinteraksi dengan pekerja aktivitas dan pengambil keputusan dengan menyediakan mereka tugas kerja yang dikenal sebagai tugas. Ada tiga jenis tugas di Amazon SWF:

- Activity task (Tugas aktivitas) – Tugas Activity (Aktivitas) memberitahu pekerja aktivitas untuk menjalankan fungsinya, seperti untuk memeriksa inventaris atau menagih kartu kredit. Tugas aktivitas berisi semua informasi yang dibutuhkan pekerja aktivitas untuk menjalankan fungsinya.
- Lambda task (Tugas Lambda) – Tugas Lambda mirip dengan tugas Aktivitas, tetapi menjalankan fungsi Lambda bukan aktivitas Amazon SWF tradisional. Selengkapnya tentang cara menentukan tugas Lambda, lihat AWS Lambda tugas di Amazon SWF.
- Decision task (Tugas keputusan) – Tugas Decision (Keputusan) memberitahu pengambil keputusan bahwa keadaan eksekusi alur kerja telah berubah sehingga pengambil keputusan dapat menentukan aktivitas berikutnya yang perlu dilakukan. Tugas keputusan berisi riwayat alur kerja saat ini.

Amazon SWF menjadwalkan tugas keputusan saat alur kerja dimulai dan kapan pun status alur kerja berubah, seperti saat tugas aktivitas selesai. Setiap tugas keputusan berisi tampilan pemberian nomor halaman dari seluruh riwayat eksekusi alur kerja. Pengambil keputusan menganalisis riwayat eksekusi alur kerja dan merespon kembali ke Amazon SWF dengan serangkaian keputusan yang menentukan apa yang harus terjadi selanjutnya dalam eksekusi alur kerja. Pada dasarnya, setiap tugas keputusan memberikan kesempatan kepada pengambil keputusan untuk menilai alur kerja dan memberikan arahan kembali ke Amazon SWF.

Untuk memastikan bahwa tidak ada keputusan yang bertentangan diproses, Amazon SWF menetapkan setiap tugas keputusan untuk satu pengambil keputusan dan hanya mengizinkan satu tugas keputusan pada suatu waktu untuk menjadi aktif dalam eksekusi alur kerja.

Tabel berikut menunjukkan hubungan antara konstruksi yang berbeda terhubung dengan alur kerja dan pengambil keputusan.

| Desain Logis | Terdaftar Sebagai | Dilakukan oleh | Menerima & Melakukan | Menghasilkan |
|---|---|---|---|---|
| Alur Kerja | Jenis Alur Kerja | Pengambil Keputusan | Tugas Keputusan | Keputusan |

Ketika pekerja aktivitas telah menyelesaikan tugas aktivitas, pekerja aktivitas melaporankan ke Amazon SWF bahwa tugas selesai, dan mencakup hasil relevan yang dihasilkan. Amazon SWF memperbarui riwayat eksekusi alur kerja dengan sebuah kejadian yang menunjukkan tugas selesai dan kemudian menjadwalkan tugas keputusan untuk mengirimkan riwayat yang diperbarui ke pengambil keputusan.

Amazon SWF menetapkan setiap tugas aktivitas secara tepat ke satu pekerja aktivitas. Setelah tugas ditetapkan, tidak ada pekerja aktivitas lain yang dapat mengklaim atau melakukan tugas itu.

Tabel berikut menunjukkan hubungan antara konstruksi yang berbeda terhubung dengan aktivitas.

| Desain Logis | Terdaftar Sebagai | Dilakukan oleh | Menerima & Melakukan | Menghasilkan |
|---|---|---|---|---|
| Aktivitas | Jenis Aktivitas | Pekerja Aktivitas | Tugas Aktivitas | Data Hasil |

# Daftar tugas di Amazon SWF

Daftar tugas menyediakan cara untuk mengatur berbagai tugas yang terhubung dengan alur kerja. Anda dapat memikirkan daftar tugas sebagai antrean dinamis. Ketika tugas dijadwalkan di Amazon SWF, Anda dapat menentukan antrean (daftar tugas) untuk dimasukkan. Demikian pula, ketika Anda melakukan pemilihan di Amazon SWF untuk tugas, Anda menyebutkan antrean mana (daftar tugas) untuk mendapatkan tugas.

Daftar tugas menyediakan mekanisme yang fleksibel untuk mengarahkan tugas ke pekerja sesuai kebutuhan kasus penggunaan Anda. Daftar tugas bersifat dinamis sehingga Anda tidak perlu mendaftarkan daftar tugas atau membuatnya secara eksplisit melalui tindakan: cukup menjadwalkan tugas untuk membuat daftar tugas jika belum ada.

Ada daftar terpisah tugas activity (aktivitas) dan tugas decision (keputusan). Tugas selalu dijadwalkan hanya pada satu daftar tugas; tugas tidak dibagikan di seluruh daftar. Selain itu, seperti aktivitas dan alur kerja, daftar tugas dicakup ke AWS wilayah tertentu dan domain Amazon SWF.

Topik



## Daftar Tugas Keputusan

Setiap eksekusi alur kerja dihubungkan dengan daftar tugas keputusan tertentu. Ketika jenis alur kerja terdaftar (RegisterWorkflowTypetindakan), Anda dapat menentukan daftar tugas default untuk eksekusi jenis alur kerja tersebut. Saat pemulai alur kerja memulai eksekusi alur kerja (tindakan `StartWorkflowExecution`), pemulai alur kerja memiliki opsi untuk menentukan daftar tugas yang berbeda untuk eksekusi alur kerja tersebut.

Saat pengambil keputusan melakukan pemilihan untuk tugas keputusan baru (tindakan `PollForDecisionTask`), pengambil keputusan menentukan daftar tugas keputusan yang akan diambil. Satu pengambil keputusan dapat menangani beberapa eksekusi alur kerja dengan memanggil `PollForDecisionTask` beberapa kali, menggunakan daftar tugas yang berbeda di setiap panggilan, di mana setiap daftar tugas khusus untuk eksekusi alur kerja tertentu. Atau, pengambil keputusan dapat melakukan pemilihan daftar tugas keputusan tunggal yang menyediakan tugas keputusan untuk beberapa eksekusi alur kerja. Anda juga dapat memiliki beberapa pengambil keputusan untuk melayani eksekusi alur kerja tunggal dengan melakukan pemiliha daftar tugas untuk eksekusi alur kerja tersebut.

## Daftar Tugas Aktivitas

Daftar tugas aktivitas tunggal dapat berisi tugas dari jenis aktivitas yang berbeda. Tugas dijadwalkan pada daftar tugas secara berurutan. Amazon SWF mengembalikan tugas dari daftar secara berurutan berdasarkan upaya terbaik. Dalam beberapa keadaan, tugas mungkin tidak keluar dari daftar secara berurutan.

Ketika jenis aktivitas terdaftar (RegisterActivityTypetindakan), Anda dapat menentukan daftar tugas default untuk jenis aktivitas tersebut. Secara default, tugas aktivitas jenis ini akan dijadwalkan pada daftar tugas yang ditentukan; Namun, ketika penentu menjadwalkan tugas aktivitas (ScheduleActivityTaskkeputusan), penentu secara opsional dapat menentukan daftar tugas yang

berbeda untuk menjadwalkan tugas. Jika pengambil keputusan tidak menentukan daftar tugas, daftar tugas default akan digunakan. Sehingga, Anda dapat menempatkan tugas aktivitas pada daftar tugas tertentu sesuai dengan atribut tugas. Misalnya, Anda dapat menempatkan semua instans tugas aktivitas untuk jenis kartu kredit tertentu pada daftar tugas tertentu.

### Perutean Tugas

Saat pekerja aktivitas melakukan polling untuk tugas baru (PollForActivityTasktindakan), ia dapat menentukan daftar tugas aktivitas yang akan diambil. Jika ya, pekerja aktivitas hanya akan menerima tugas dari daftar itu. Dengan cara ini, Anda dapat memastikan bahwa tugas tertentu hanya diberikan kepada pekerja aktivitas tertentu. Misalnya, Anda dapat membuat daftar tugas yang berisi tugas yang memerlukan penggunaan komputer dengan performa tinggi. Hanya pekerja aktivitas yang berjalan pada perangkat keras yang sesuai akan melakukan pemilihan pada daftar tugas tersebut. Contoh lain adalah membuat daftar tugas untuk wilayah geografis tertentu. Anda kemudian dapat memastikan bahwa hanya pekerja ditempatkan di wilayah tersebut yang akan mengambil tugas tersebut. Atau Anda dapat membuat daftar tugas untuk pesanan prioritas tinggi dan selalu periksa daftar tersebut terlebih dahulu.

Menugaskan tugas tertentu untuk pekerja aktivitas tertentu dengan cara ini disebuttask routing (perutean tugas). Perutean tugas bersifat opsional; jika Anda tidak menentukan daftar tugas saat menjadwalkan tugas aktivitas, tugas secara otomatis ditempatkan pada daftar tugas default.

## Penutupan eksekusi alur kerja di Amazon SWF

Setelah Anda memulai eksekusi alur kerja, itu terbuka. Eksekusi alur kerja terbuka dapat ditutup sebagai selesai, dibatalkan, gagal, atau habis waktu. Selain itu, dapat dilanjutkan sebagai eksekusi baru atau diakhiri. Eksekusi alur kerja dapat ditutup oleh pengambil keputusan, oleh orang yang mengelola alur kerja, atau oleh Amazon SWF.

Jika pengambil keputusan menentukan bahwa aktivitas alur kerja telah selesai, eksekusi alur kerja harus ditutup sebagai selesai dengan menggunakan tindakan `RespondDecisionTaskCompleted` dan meneruskan keputusan `CompleteWorkflowExecution`.

Atau, pengambil keputusan mungkin menutup eksekusi alur kerja sebagai dibatalkan atau gagal. Untuk membatalkan eksekusi, pengambil keputusan harus menggunakan tindakan `RespondDecisionTaskCompleted` dan meneruskan keputusan `CancelWorkflowExecution`.

Pengambil keputusan harus menggagalkan eksekusi alur kerja jika memasuki keadaan di luar ranah penyelesaian normal. Untuk menggagalkan eksekusi, pengambil keputusan harus

menggunakan tindakan `RespondDecisionTaskCompleted` dan meneruskan keputusan `FailWorkflowExecution`.

Amazon SWF memantau eksekusi alur kerja untuk memastikan bahwa mereka tidak melebihi pengaturan batas waktu yang ditentukan pengguna. Jika waktu eksekusi alur kerja habis, Amazon SWF secara otomatis menutupnya. Selengkapnya tentang nilai batas waktu, lihat bagian Tipe Batas Waktu Amazon SWF.

Pengambil keputusan mungkin juga menutup eksekusi dan secara logis melanjutkannya sebagai eksekusi baru menggunakan tindakan `RespondDecisionTaskCompleted` dan meneruskan keputusan `ContinueAsNewWorkflowExecution`. Strategi ini berguna untuk eksekusi alur kerja yang berjalan lama di mana riwayatnya dapat tumbuh terlalu besar dari waktu ke waktu.

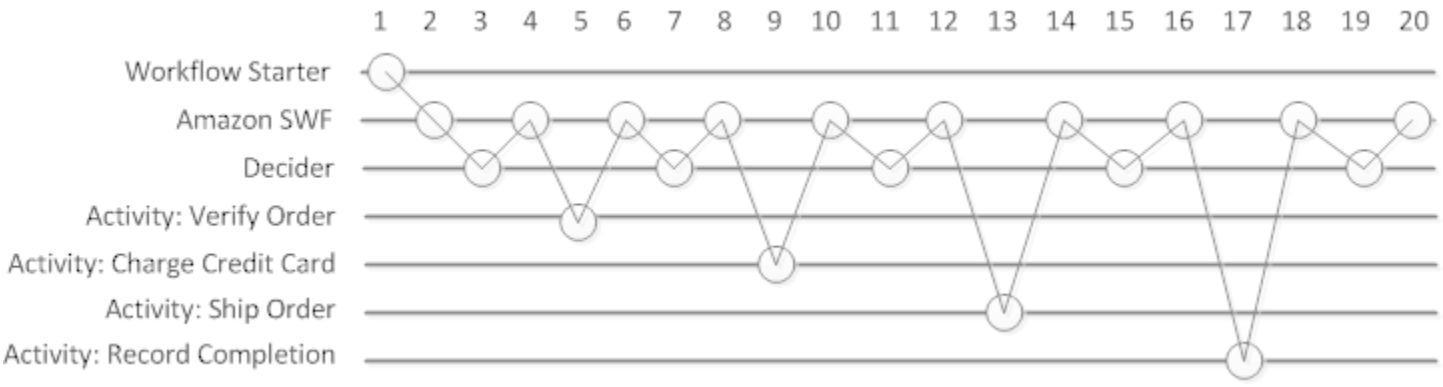
Akhirnya, Anda dapat mengakhiri eksekusi alur kerja langsung dari konsol Amazon SWF atau secara terprogram menggunakan API `TerminateWorkflowExecution`. Penghentian memaksa penutupan eksekusi alur kerja. Pembatalan lebih disarankan daripada penghentian, karena pengambil keputusan Anda dapat mengelola penutupan eksekusi alur kerja.

Amazon SWF mengakhiri eksekusi alur kerja jika eksekusi melebihi batas yang ditentukan layanan tertentu. Amazon SWF berakhir alur kerja turunan (anak) jika alur kerja induk telah dihentikan dan kebijakan turunan (anak) yang berlaku menunjukkan bahwa alur kerja turunan (anak) juga harus dihentikan.

## Siklus hidup alur kerja Amazon SWF

Dari awal eksekusi alur kerja hingga penyelesaiannya, Amazon SWF berinteraksi dengan aktor dengan menugaskan mereka tugas yang tepat, baik tugas aktivitas atau tugas keputusan.

Diagram berikut menunjukkan siklus hidup eksekusi alur kerja pemrosesan pesanan dari perspektif komponen yang bertindak di atasnya.



# Siklus Hidup Eksekusi Alur Kerja

Tabel berikut menjelaskan setiap tugas pada gambar sebelumnya.

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 1. Pemulai alur kerja memanggil tindakan Amazon SWF yang sesuai memulai eksekusi alur kerja untuk pesanan, memberikan informasi pesanan. | Tindakan `StartWorkflowExecution`. |
| 2. Amazon SWF menerima permintaan eksekusi awal alur kerja dan kemudian menjadwalkan tugas keputusan pertama. | Kejadian `WorkflowExecutionStarted` dan kejadian `DecisionTaskScheduled`. |
| 3. Pengambil keputusan menerima tugas dari Amazon SWF, meninjau riwayat, menerapkan logika koordinasi untuk menentukan bahwa tidak ada aktivitas sebelumnya yang terjadi, membuat keputusan untuk menjadwalkan aktivitas Verifikasi Pesanan dengan informasi yang | Tindakan `PollForDecisionTask`. Tindakan `RespondDecisionTaskCompleted` dan keputusan `ScheduleActivityTask`. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| dibutuhkan pekerja aktivitas untuk memproses tugas, dan mengembalikan keputusan ke Amazon SWF. | |
| 4. Amazon SWF menerima keputusan , menjadwalkan tugas aktivitas Verifikasi Pesanan, dan menunggu tugas aktivitas untuk menyelesaikan atau hingga habis waktunya. | Kejadian `ActivityTaskScheduled` |
| 5. Pekerja aktivitas yang dapat melakukan aktivitas Verifikasi Pesanan menerima tugas, menjalankannya, dan mengembalikan hasilnya ke Amazon SWF. | Tindakan `PollForActivityTask` dan tindakan `RespondActivityTaskCompleted`. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 6. Amazon SWF menerima hasil dari aktivitas Verifikasi Pesanan, menambahkannya ke riwayat alur kerja, dan menjadwalkan tugas keputusan. | Kejadian ActivityTaskCompleted dan kejadian DecisionTaskScheduled. |
| 7. Penentu menerima tugas dari Amazon SWF, meninjau riwayat, menerapkan logika koordinasi, membuat keputusan untuk menjadwalkan ChargeCreditCard tugas aktivitas dengan informasi yang dibutuhkan pekerja aktivitas untuk memproses tugas, dan mengembalikan keputusan ke Amazon SWF. | Tindakan PollForDecisionTask. Tindakan RespondDecisionTaskCompleted dengan keputusan ScheduleActivityTask. |
| 8. Amazon SWF menerima keputusan, menjadwalkan tugas ChargeCreditCard aktivitas, dan menunggu sampai selesai atau habis waktu. | Kejadian DecisionTaskCompleted dan kejadian ActivityTaskScheduled. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 9. Pekerja aktivitas yang dapat melakukan ChargeCreditCard aktivitas menerima tugas, menjalank annya, dan mengembalikan hasilnya ke Amazon SWF. | PollForActivityTask dan tindakan RespondActivityTaskCompleted. |
| 10. Amazon SWF menerima hasil tugas ChargeCreditCard aktivitas, menambahk annya ke riwayat alur kerja, dan menjadwal kan tugas keputusan. | Kejadian ActivityTaskCompleted dan kejadian DecisionTaskScheduled. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 11. Penentu menerima tugas dari Amazon SWF, meninjau riwayat, menerapkan logika koordinasi, membuat keputusan untuk menjadwalkan ShipOrder tugas aktivitas dengan informasi yang dibutuhkan pekerja aktivitas untuk melakukan tugas, dan mengembalikan keputusan ke Amazon SWF. | Tindakan PollForDecisionTask. RespondDecisionTaskCompleted dengan keputusan ScheduleActivityTask. |
| 12. Amazon SWF menerima keputusan, menjadwalkan tugas ShipOrder aktivitas, dan menunggu sampai selesai atau habis waktu. | Kejadian DecisionTaskCompleted dan kejadian ActivityTaskScheduled. |
| 13. Pekerja aktivitas yang dapat melakukan ShipOrder aktivitas menerima tugas, menjalankannya, dan mengembalikan hasilnya ke Amazon SWF. | Tindakan PollForActivityTask dan tindakan RespondActivityTaskCompleted. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 14. Amazon SWF menerima hasil tugas ShipOrder aktivitas, menambahkannya ke riwayat alur kerja, dan menjadwalkan tugas keputusan. | Kejadian ActivityTaskCompleted dan kejadian DecisionTaskScheduled. |
| 15. Penentu menerima tugas dari Amazon SWF, meninjau riwayat, menerapkan logika koordinasi, membuat keputusan untuk menjadwalkan RecordCompletion tugas aktivitas dengan informasi yang dibutuhkan pekerja aktivitas untuk melakukan tugas, dan mengembalikan keputusan ke Amazon SWF. | Tindakan PollForDecisionTask. Tindakan RespondDecisionTaskCompleted dengan keputusan ScheduleActivityTask. |
| 16. Amazon SWF menerima keputusan, menjadwalkan tugas RecordCompletion aktivitas, dan menunggu sampai selesai atau habis waktu. | Kejadian DecisionTaskCompleted dan kejadian ActivityTaskScheduled. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 17. Pekerja aktivitas yang dapat melakukan RecordCompletion aktivitas menerima tugas, menjalankannya, dan mengembalikan hasilnya ke Amazon SWF. | Tindakan PollForActivityTask dan tindakan RespondActivityTaskCompleted. |
| 18. Amazon SWF menerima hasil tugas RecordCompletion aktivitas, menambahkannya ke riwayat alur kerja, dan menjadwalkan tugas keputusan. | Kejadian ActivityTaskCompleted dan kejadian DecisionTaskScheduled. |
| 19. Pengambil keputusan menerima tugas dari Amazon SWF, meninjau riwayat, menerapkan logika koordinasi, membuat keputusan untuk menutup eksekusi alur kerja dan mengembalikan keputusan bersama dengan hasil apa pun ke Amazon SWF. | Tindakan PollForDecisionTask. Tindakan RespondDecisionTaskCompleted dengan keputusan CompleteWorkflowExecution. |

| Deskripsi | Tindakan, Keputusan, atau Kejadian |
| --- | --- |
| 20. Amazon SWF menutup eksekusi alur kerja dan mengarsipkan riwayat untuk referensi di masa mendatang. | Kejadian `WorkflowExecutionCompleted` . |

# Polling untuk tugas di Amazon SWF

Deciders dan pekerja aktivitas berkomunikasi dengan Amazon SWF menggunakan long polling (proses pemilihan yang panjang). Pengambil keputusan atau pekerja aktivitas secara berkala memulai komunikasi dengan Amazon SWF, memberi tahu Amazon SWF tentang ketersediaannya untuk menerima tugas, lalu menentukan daftar tugas untuk mendapatkan tugas.

Jika tugas tersedia pada daftar tugas yang ditentukan, Amazon SWF segera mengembalikannya sebagai respons. Jika tidak ada tugas yang tersedia, Amazon SWF menahan koneksi TCP terbuka hingga 60 detik sehingga, jika tugas tersedia selama waktu tersebut, tugas tersebut dikembalikan dalam koneksi yang sama. Jika tidak ada tugas yang tersedia dalam waktu 60 detik, respons kosong dikembalikan dan koneksi ditutup. (Respons kosong adalah struktur Tugas di mana nilai taskToken adalah string kosong.) Jika ini terjadi, pengambil keputusan atau pekerja aktivitas harus melakukan pemilihan lagi.

Proses pemilihan yang panjang bekerja dengan baik untuk pemrosesan tugas volume tinggi. Pengambil keputusan dan pekerja aktivitas dapat mengelola kapasitas mereka sendiri, dan mudah digunakan ketika pengambil keputusan dan pekerja aktivitas berada di belakang firewall.

Selengkapnya, lihat Polling untuk Tugas Keputusan dan Polling untuk Tugas Aktivitas.

# Konsep alur kerja lanjutan di Amazon SWF

Contoh perdagangan elektronik di bagian ??? menunjukkan skenario alur kerja yang disederhanakan. Pada kenyataannya, Anda mungkin ingin alur kerja Anda melakukan tugas bersamaan (mengirim email konfirmasi pesanan saat mengotorisasi kartu kredit), merekam kejadian besar (semua item dikemas), memperbarui pesanan dengan perubahan (menambah atau menghapus item), dan buat keputusan lanjutan lainnya sebagai bagian dari eksekusi alur kerja Anda. Bagian ini menjelaskan konsep alur kerja lanjutan yang dapat Anda gunakan untuk membangun alur kerja Anda.

Konsep lanjutan

- Penentuan versi
- Sinyal
- Alur kerja anak di Amazon SWF
- Penanda di Amazon SWF
- Tag di Amazon SWF
- Menerapkan pilihan eksklusif dengan Amazon SWF
- Timer di Amazon SWF
- Membatalkan tugas aktivitas di Amazon SWF

## Penentuan versi

Kebutuhan bisnis sering kali mengharuskan Anda memiliki implementasi atau variasi berbeda dari alur kerja atau aktivitas yang sama yang berjalan secara bersamaan. Misalnya, Anda mungkin ingin menguji implementasi baru dari alur kerja saat yang lain sedang dalam produksi. Anda mungkin juga ingin menjalankan dua implementasi berbeda dengan dua set fitur yang berbeda, seperti implementasi dasar dan premium. Versioning memungkinkan Anda menjalankan beberapa implementasi alur kerja dan aktivitas secara bersamaan, untuk tujuan apa pun yang memenuhi persyaratan Anda.

Jenis alur kerja dan aktivitas memiliki versi yang terkait dengannya yang ditentukan pada waktu pendaftaran. Versi adalah string bentuk bebas dan Anda dapat memilih skema versioning Anda sendiri. Untuk membuat versi baru dari jenis terdaftar, Anda harus mendaftarkannya dengan nama yang sama dan versi yang berbeda. Daftar tugas di Amazon SWF yang dijelaskan sebelumnya, dapat lebih membantu Anda untuk mengimplementasikan versioning. Pertimbangkan situasi di

mana Anda memiliki eksekusi alur kerja yang sudah berjalan lama dari jenis tertentu yang sudah berlangsung, dan keadaan mengharuskan Anda merevisi alur kerja, seperti menambahkan fitur baru. Anda dapat mengimplementasikan fitur baru dengan membuat versi baru dari jenis aktivitas dan pekerja, dan pengambil keputusan baru. Kemudian Anda dapat meluncurkan eksekusi versi alur kerja baru menggunakan kumpulan daftar tugas yang berbeda. Dengan cara ini, Anda dapat menjalankan alur kerja dari berbagai versi secara bersamaan tanpa mempengaruhi satu sama lain.

## Sinyal

Sinyal memungkinkan Anda memasukkan informasi ke dalam eksekusi alur kerja yang sedang berjalan. Dalam beberapa skenario, Anda mungkin ingin menambahkan informasi ke eksekusi alur kerja yang sedang berjalan untuk memberi tahu bahwa ada sesuatu yang berubah atau untuk menginformasikan kejadian eksternal. Setiap proses dapat mengirim sinyal ke eksekusi alur kerja terbuka. Misalnya, satu eksekusi alur kerja mungkin menandakan yang lain.

> **Note**
>
> Upaya untuk mengirim sinyal ke eksekusi alur kerja yang tidak terbuka mengakibatkan `SignalWorkflowExecution` gagal dengan `UnknownResourceFault`.

Untuk menggunakan sinyal, tentukan nama sinyal dan data yang akan diteruskan ke sinyal—jika ada. Kemudian, program penentu untuk mengenali peristiwa sinyal (WorkflowExecutionSignaled) dalam sejarah dan memprosesnya dengan tepat. Saat proses ingin memberi sinyal eksekusi alur kerja, proses akan membuat panggilan ke Amazon SWF (menggunakan SignalWorkflowExecutiontindakan atau, dalam kasus penentu, menggunakan SignalExternalWorkflowExecutionkeputusan) yang menentukan pengidentifikasi untuk eksekusi alur kerja target, nama sinyal, dan data sinyal. Amazon SWF kemudian menerima sinyal, mencatatnya dalam riwayat eksekusi alur kerja target, dan menjadwalkan tugas keputusan untuknya. Saat pengambil keputusan menerima tugas keputusan, juga menerima sinyal di dalam riwayat eksekusi alur kerja. Pengambil keputusan kemudian dapat mengambil tindakan yang tepat berdasarkan sinyal dan datanya.

Terkadang Anda mungkin ingin menunggu sinyal. Misalnya, pengguna dapat membatalkan pesanan dengan mengirimkan sinyal, tetapi hanya dalam waktu satu jam setelah melakukan pemesanan. Amazon SWF tidak memiliki primitif untuk memungkinkan pengambil keputusan menunggu sinyal dari layanan. Fungsionalitas jeda perlu diimplementasikan di dalam pengambil keputusan itu sendiri. Untuk menjeda, pengambil keputusan harus memulai pengatur waktu, menggunakan keputusan `StartTimer`, yang menentukan durasi pengambil keputusan akan menunggu sinyal sambil

melanjutkan proses pemiilihan untuk tugas keputusan. Ketika menerima tugas keputusan, pengambil keputusan harus memeriksa riwayat untuk melihat apakah sinyal telah diterima atau pengatur waktu telah diaktifkan. Jika sinyal telah diterima, maka pengambil keputusan harus membatalkan pengatur waktu. Namun, jika sebaliknya, pengatur waktu telah menyala, maka sinyal tidak tiba dalam waktu yang ditentukan. Untuk meringkas, dalam menunggu sinyal tertentu, lakukan hal berikut.

1. Buat pengatur waktu untuk jumlah waktu yang harus ditunggu oleh pengambil keputusan.
2. Ketika tugas keputusan diterima, periksa riwayat untuk melihat apakah sinyal telah tiba atau apakah pengatur waktu telah diaktifkan.
3. Jika sinyal telah tiba, batalkan pengatur waktu menggunakan keputusan `CancelTimer` dan proses sinyal. Bergantung pada waktunya, riwayat mungkin berisi kejadian `TimerFired` dan `WorkflowExecutionSignaled`. Dalam kasus seperti itu, Anda dapat mengandalkan urutan relatif kejadian dalam riwayat untuk menentukan mana yang terjadi lebih dulu.
4. Jika pengatur waktu telah menyala, sebelum sinyal diterima, maka pengambil keputusan telah kehabisan waktu untuk menunggu sinyal. Anda dapat gagal dalam eksekusi atau melakukan logika lain apa pun yang sesuai dengan kasus penggunaan Anda.

Untuk kasus di mana alur kerja harus dibatalkan—misalnya, pesanan itu sendiri dibatalkan oleh pelanggan—tindakan `RequestCancelWorkflowExecution` harus digunakan daripada mengirim sinyal ke alur kerja.

Beberapa aplikasi untuk sinyal antara lain sebagai berikut:

- Menjeda eksekusi alur kerja dari kemajuan hingga sinyal diterima (misalnya, menunggu pengiriman inventaris).
- Memberikan informasi ke eksekusi alur kerja yang mungkin memengaruhi logika bagaimana pengambil keputusan membuat keputusan. Hal ini berguna untuk alur kerja yang dipengaruhi oleh kejadian eksternal (misalnya, mencoba menyelesaikan penjualan saham setelah pasar tutup).
- Memperbarui eksekusi alur kerja saat Anda mengantisipasi bahwa perubahan mungkin terjadi (misalnya, mengubah jumlah pesanan setelah pesanan dilakukan dan sebelum dikirim).

Dalam contoh berikut, eksekusi alur kerja dikirim sinyal untuk membatalkan pesanan.

```
https://swf.us-east-1.amazonaws.com
SignalWorkflowExecution
{"domain": "867530901",
 "workflowId": "20110927-T-1",
```

```
 "runId": "f5ebbac6-941c-4342-ad69-dfd2f8be6689",
 "signalName": "CancelOrder",
 "input": "order 3553"}
```

Jika eksekusi alur kerja menerima sinyal, Amazon SWF mengembalikan respons HTTP yang berhasil serupa dengan berikut ini. Amazon SWF akan menghasilkan tugas keputusan untuk memberi tahu pengambil keputusan untuk memproses sinyal.

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: bf78ae15-3f0c-11e1-9914-a356b6ea8bdf
```

# Alur kerja anak di Amazon SWF

Alur kerja yang rumit dapat dipecah menjadi komponen yang lebih kecil, lebih mudah dikelola, dan berpotensi dapat digunakan kembali dengan menggunakan alur kerja turunan (anak). Alur kerja anak adalah eksekusi alur kerja yang dimulai oleh eksekusi alur kerja (induk) lain. Untuk memulai alur kerja anak, pengambil keputusan alur kerja induk menggunakan keputusan `StartChildWorkflowExecution`. Data masukan yang ditentukan dengan keputusan ini tersedia untuk alur kerja anak melalui riwayatnya.

Atribut untuk keputusan `StartChildWorkflowExecution` juga menentukan child policy (kebijakan anak), yaitu, bagaimana Amazon SWF harus menangani situasi di mana eksekusi alur kerja induk berakhir sebelum eksekusi alur kerja anak. Ada tiga nilai yang mungkin:

- TERMINATE: Amazon SWF akan mengakhiri eksekusi anak.
- REQUEST_CANCEL: Amazon SWF akan mencoba untuk membatalkan eksekusi anak dengan menempatkan kejadian `WorkflowExecutionCancelRequested` dalam riwayat eksekusi alur kerja anak.
- ABANDON: Amazon SWF tidak akan mengambil tindakan apa pun; eksekusi anak akan terus berjalan.

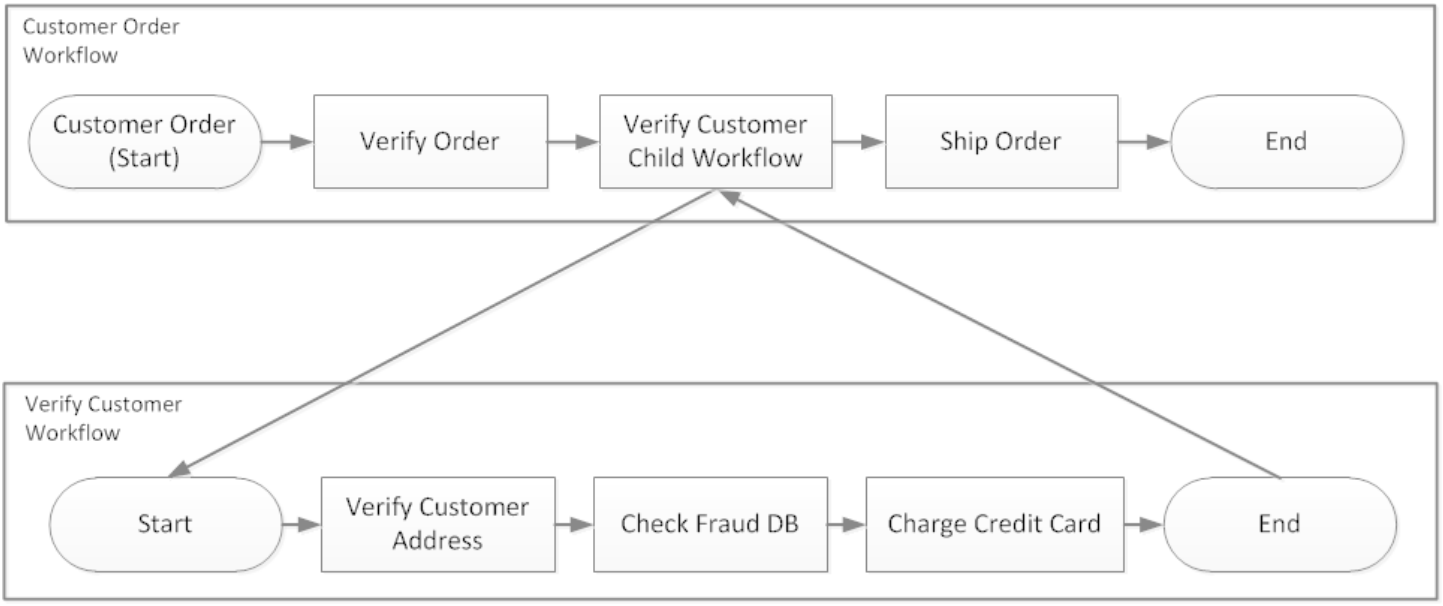
Setelah eksekusi alur kerja anak dimulai, prosesnya berjalan seperti eksekusi biasa. Saat selesai, Amazon SWF mencatat penyelesaian, beserta hasilnya, dalam riwayat alur kerja eksekusi alur kerja induk. Contoh alur kerja anak meliputi:

- Alur kerja anak pemrosesan kartu kredit yang digunakan oleh alur kerja di berbagai situs web

- Alur kerja anak email yang memverifikasi alamat email pelanggan, memeriksa daftar penyisihan, mengirim email, dan memverifikasi bahwa email tidak hilang atau gagal.
- Penyimpanan basis data dan pengambilan alur kerja anak yang menggabungkan koneksi, penyiapan, transaksi, dan verifikasi.
- Alur kerja anak kompilasi kode sumber yang menggabungkan pembuatan, pengemasan, dan verifikasi.

Dalam contoh perdagangan elektronik, Anda mungkin ingin menjadikan aktivitas Charge Credit Card (Biaya Kartu Kredit) sebagai alur kerja anak. Untuk melakukan ini, Anda dapat mendaftarkan alur kerja Verify Customer (Verifikasi Pelanggan) baru, mendaftarkan aktivitas Verify Customer Address (Verifikasi Alamat Pelanggan) dan aktivitas Check Fraud DB (Periksa DB Penipuan), dan menentukan logika koordinasi untuk tugas tersebut. Kemudian, pengambil keputusan dalam alur kerja Customer Order (Pesanan Pelanggan) dapat memulai alur kerja anak Verify Customer (Verifikasi Pelanggan) dengan menjadwalkan keputusan `StartChildWorkflowExecution` yang menentukan jenis alur kerja ini.

Gambar berikut memperlihatkan alur kerja pesanan pelanggan yang menyertakan alur kerja anak Verify Customer (Verifikasi Pelanggan) baru, yang memeriksa alamat pelanggan, memeriksa basis data penipuan, dan menagih kartu kredit.




Beberapa alur kerja dapat membuat eksekusi alur kerja anak menggunakan jenis alur kerja yang sama. Misalnya, alur kerja anak Verify Customer (Verifikasi Pelanggan) juga bisa digunakan di bagian lain organisasi. Kejadian untuk alur kerja anak berada di riwayat alur kerjanya sendiri dan tidak disertakan dalam riwayat alur kerja induk.

Karena alur kerja anak hanyalah eksekusi alur kerja yang dimulai oleh pengambil keputusan, sehingga bisa dimulai sebagai eksekusi alur kerja yang berdiri sendiri secara normal.

# Penanda di Amazon SWF

Terkadang, Anda mungkin ingin merekam informasi dalam riwayat alur kerja dari eksekusi alur kerja yang khusus untuk kasus penggunaan Anda. Penanda memungkinkan Anda untuk merekam informasi dalam riwayat eksekusi alur kerja yang dapat Anda gunakan untuk tujuan khusus atau skenario apa pun.

Untuk menggunakan spidol, penentu menggunakan RecordMarker keputusan, memberi nama penanda, melampirkan data yang diinginkan pada keputusan, dan memberi tahu Amazon SWF menggunakan tindakan tersebut. `RespondDecisionTaskCompleted` Amazon SWF menerima permintaan, mencatat penanda dalam riwayat alur kerja, dan memberlakukan keputusan lain dalam permintaan. Sejak saat itu, pengambil keputusan dapat melihat penanda dalam riwayat alur kerja dan menggunakannya dengan cara apa pun yang Anda programkan.

Merekam penanda tidak dengan sendirinya untuk memulai tugas keputusan. Untuk mencegah eksekusi alur kerja macet, sesuatu harus dilakukan untuk melanjutkan eksekusi alur kerja. Misalnya, pengambil keputusan yang menjadwalkan tugas aktivitas lain, eksekusi alur kerja yang menerima sinyal, atau penyelesaian tugas aktivitas yang dijadwalkan sebelumnya.

Contoh penanda antara lain sebagai berikut:

- Penghitung yang menghitung jumlah loop (putaran) dalam alur kerja rekursif.
- Kemajuan eksekusi alur kerja berdasarkan hasil aktivitas.
- Informasi yang diringkas dari kejadian riwayat alur kerja sebelumnya.

Dalam contoh perdagangan elektronik, Anda dapat menambahkan aktivitas yang memeriksa inventaris setiap hari dan menambah jumlah penanda setiap kali. Kemudian, Anda dapat menambahkan logika keputusan yang mengirim email kepada pelanggan atau memberi tahu manajer saat jumlah melebihi lima, tanpa harus meninjau seluruh riwayat.

Dalam contoh berikut, pengambil keputusan menyelesaikan tugas keputusan dan merespons dengan tindakan `RespondDecisionTaskCompleted` yang berisi keputusan `RecordMarker`.

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
```

```
{
  "taskToken":"12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[{
          "decisionType":"RecordMarker",
          "recordMarkerDecisionAttributes":{
              "markerName":"customer elected special shipping offer"
          }
      },
  ]
}
```

Jika Amazon SWF berhasil merekam penanda, itu mengembalikan respons HTTP yang berhasil serupa dengan berikut ini.

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

# Tag di Amazon SWF

Amazon SWF mendukung penandaan eksekusi alur kerja. Ini sangat berguna ketika Anda memiliki banyak sumber daya.

Amazon SWF mendukung penandaan eksekusi alur kerja hingga lima penanda. Setiap penanda adalah string bentuk bebas dan panjangnya dapat mencapai 256 karakter. Jika ingin menggunakan penanda, Anda harus menetapkannya saat memulai eksekusi alur kerja. Anda tidak dapat menambahkan penanda ke eksekusi alur kerja setelah dimulai, Anda juga tidak dapat mengedit atau menghapus penanda yang telah ditetapkan untuk eksekusi alur kerja.

IAM mendukung pengontrolan akses ke domain Amazon SWF berdasarkan penanda. Untuk mengontrol akses berdasarkan penanda, berikan informasi tentang penanda Anda di elemen kondisi kebijakan IAM.

## Kelola tag

Kelola tag Amazon Simple Workflow Service menggunakan AWS SDKs atau dengan berinteraksi langsung dengan Amazon SWF API. Dengan menggunakan API, Anda dapat menambahkan penanda saat mendaftarkan domain, mencantumkan penanda untuk domain yang ada, dan menambahkan atau menghapus penanda untuk domain yang ada.

> **Note**
>
> Ada batas 50 penanda per sumber daya. Lihat Kuota Akun Umum untuk Amazon SWF

- `RegisterDomain`
- `ListTagsForResource`
- `TagResource`
- `UntagResource`

Untuk informasi selengkapnya, lihat Bekerja dengan Amazon SWF APIs, dan Amazon Simple Workflow Service API Reference (Referensi API Amazon Simple Workflow Service).

## Menandai eksekusi alur kerja

Dengan Amazon SWF, Anda dapat mengaitkan tag dengan eksekusi alur kerja dan kemudian meminta eksekusi alur kerja berdasarkan tag ini. Anda dapat memfilter listi saat Anda menggunakan operasi visibilitas. Dengan hati-hati memilih tag yang Anda tetapkan untuk eksekusi, Anda dapat menggunakannya untuk menyediakan daftar yang berarti.

Misalnya, Anda menjalankan beberapa pusat pemenuhan. Dengan tag, Anda dapat mencantumkan proses yang terjadi di pusat pemenuhan tertentu. Atau, jika pelanggan mengonversi berbagai jenis file media, tag dapat menunjukkan proses yang berbeda saat mengonversi file video, audio, dan gambar.

Anda dapat menghubungkan hingga lima penanda dengan eksekusi alur kerja saat Anda memulai eksekusi menggunakan tindakan `StartWorkflowExecution`, keputusan `StartChildWorkflowExecution`, atau keputusan `ContinueAsNewWorkflowExecution`. Saat Anda menggunakan tindakan visibilitas untuk membuat daftar atau menghitung eksekusi alur kerja, Anda dapat memfilter hasil berdasarkan tag Anda.

Untuk menggunakan penandaan

1. Rancang strategi penandaan. Pikirkan tentang persyaratan bisnis Anda dan buat daftar penanda yang berarti bagi Anda. Tentukan eksekusi mana yang akan mendapatkan penanda mana. Meskipun eksekusi dapat ditetapkan maksimal lima penanda, perpustakaan penanda Anda dapat memiliki sejumlah penanda. Karena setiap penanda dapat berupa nilai string apa pun hingga 256 karakter, sebuah penanda dapat menggambarkan hampir semua konsep bisnis.

2. Beri penanda pada eksekusi hingga lima tag saat Anda membuatnya.
3. Buat daftar atau hitung eksekusi yang ditandai dengan penanda tertentu dengan menetapkan parameter tagFilter dengan tindakan `ListOpenWorkflowExecutions`, `ListClosedWorkflowExecutions`, `CountOpenWorkflowExecutions`, dan `CountClosedWorkflowExecutions`. Tindakan akan memfilter eksekusi berdasarkan penanda yang ditentukan.

Saat Anda menghubungkan penanda dengan eksekusi alur kerja, penanda akan dihubungkan secara permanen dengan eksekusi tersebut, dan tidak dapat dihapus.

Anda hanya dapat menentukan satu penanda dalam parameter `tagFilter` dengan `ListWorkflowExecutions`. Selain itu, pencocokan penanda peka terhadap huruf besar-kecil, dan hanya pencocokan tepat yang mengembalikan hasil.

Asumsikan Anda telah menyiapkan dua eksekusi yang ditandai sebagai berikut.

| Nama Eksekusi | Tag yang Ditugaskan |
|---|---|
| Eksekusi-Satu | Konsumen, 2011-Februari |
| Eksekusi-Dua | Grosir, 2011-Maret |

Anda dapat memfilter daftar eksekusi yang dikembalikan oleh `ListOpenWorkflowExecutions` pada penanda Konsumen. Nilai `oldestDate` dan `latestDate` ditetapkan sebagai nilai Unix Time (Waktu Unix).

```
https://swf.us-east-1.amazonaws.com
  RespondDecisionTaskCompleted
  {
    "domain":"867530901",
    "startTimeFilter":{
        "oldestDate":1262332800,
        "latestDate":1325348400
    },
    "tagFilter":{
      "tag":"Consumer"
      }
  }
```

## Kontrol akses ke domain dengan tag

Anda dapat mengontrol akses ke domain Amazon Simple Workflow Service dengan merujuk penanda yang terkait dengan domain Amazon SWF di IAM.

Misalnya, Anda dapat membatasi domain Amazon SWF yang menyertakan tag dengan `environment` kunci dan `production` nilai dengan kondisi berikut:

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/environment": "production"}
}
```

Untuk informasi selengkapnya, lihat:

- Mengontrol Akses Menggunakan Tag IAM
- Kebijakan Berbasis Tag

# Menerapkan pilihan eksklusif dengan Amazon SWF

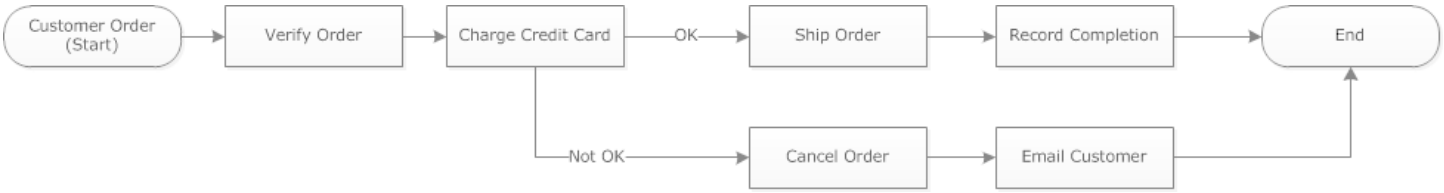
Dalam beberapa skenario, Anda mungkin ingin menjadwalkan rangkaian aktivitas yang berbeda berdasarkan hasil aktivitas sebelumnya. Dengan pola pilihan eksklusif, Anda dapat membuat alur kerja fleksibel yang memenuhi persyaratan kompleks aplikasi Anda.

Amazon SWF tidak memiliki tindakan pilihan eksklusif tertentu. Untuk menerapkan pilihan eksklusif, Anda harus menulis logika penentu Anda untuk membuat keputusan berdasarkan hasil aktivitas sebelumnya. Beberapa aplikasi untuk pilihan eksklusif meliputi:

- Melakukan kegiatan pembersihan jika hasil aktivitas sebelumnya tidak berhasil
- Menjadwalkan aktivitas yang berbeda berdasarkan apakah pelanggan membeli paket dasar atau lanjutan
- Melakukan aktivitas autentikasi pelanggan yang berbeda berdasarkan riwayat pemesanan pelanggan

Dalam contoh perdagangan elektronik, Anda dapat menggunakan pilihan eksklusif untuk mengirim atau membatalkan pesanan berdasarkan hasil penagihan kartu kredit. Pada gambar berikut, pengambil keputusan menjadwalkan tugas aktivitas Ship Order and Record Completion (Pengiriman Pesanan dan Penyelesaian Catatan) jika kartu kredit berhasil ditagih. Jika tidak, prosesnya akan

menjadwalkan tugas aktivitas Cancel Order and Email Customer (Batalkan Pesanan dan Email Pelanggan).




Pengambil keputusan menjadwalkan aktivitas `ShipOrder` jika kartu kredit berhasil ditagih. Jika tidak, pengambil keputusan menjadwalkan aktivitas `CancelOrder`.

Dalam hal ini, programkan pengambil keputusan untuk menafsirkan riwayat dan menentukan apakah kartu kredit berhasil ditagih. Untuk melakukan ini, Anda mungkin memiliki logika yang mirip dengan berikut:

```
IF lastEvent = "WorkflowExecutionStarted"
 addToDecisions ScheduleActivityTask(ActivityType = "VerifyOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "VerifyOrderActivity"
 addToDecisions ScheduleActivityTask(ActivityType = "ChargeCreditCardActivity")

#Successful Credit Card Charge Activities
ELSIF lastEvent = "ActivityTaskCompleted"
      AND ActivityType = "ChargeCreditCardActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "ShipOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "ShipOrderActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "RecordOrderCompletionActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "RecordOrderCompletionActivity"
 addToDecisions CompleteWorkflowExecution

#Unsuccessful Credit Card Charge Activities
ELSIF lastEvent = "ActivityTaskFailed"
      AND ActivityType = "ChargeCreditCardActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "CancelOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "CancelOrderActivity"
 addToDecisions ScheduleActivityTask(ActivityType = "EmailCustomerActivity")
```

```
ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "EmailCustomerActivity"
 addToDecisions CompleteWorkflowExecution

ENDIF
```

Jika kartu kredit berhasil ditagih, pengambil keputusan harus merespons dengan `RespondDecisionTaskCompleted` untuk menjadwalkan aktivitas `ShipOrder`.

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[
      {
          "decisionType":"ScheduleActivityTask",
          "scheduleActivityTaskDecisionAttributes":{
              "control":"OPTIONAL_DATA_FOR_DECIDER",
              "activityType":{
                  "name":"ShipOrder",
                  "version":"2.4"
              },
              "activityId":"3e2e6e55-e7c4-fee-deed-aa815722b7be",
              "scheduleToCloseTimeout":"3600",
              "taskList":{
                  "name":"SHIPPING"
              },
              "scheduleToStartTimeout":"600",
              "startToCloseTimeout":"3600",
              "heartbeatTimeout":"300",
              "input": "123 Main Street, Anytown, United States"
          }
      }
  ]
}
```

Jika kartu kredit tidak berhasil ditagih, pengambil keputusan harus merespons dengan `RespondDecisionTaskCompleted` untuk menjadwalkan aktivitas `CancelOrder`.

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
```

```
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[
      {
          "decisionType":"ScheduleActivityTask",
          "scheduleActivityTaskDecisionAttributes":{
              "control":"OPTIONAL_DATA_FOR_DECIDER",
              "activityType":{
                  "name":"CancelOrder",
                  "version":"2.4"
              },
              "activityId":"3e2e6e55-e7c4-fee-deed-aa815722b7be",
              "scheduleToCloseTimeout":"3600",
              "taskList":{
                  "name":"CANCELLATIONS"
              },
              "scheduleToStartTimeout":"600",
              "startToCloseTimeout":"3600",
              "heartbeatTimeout":"300",
              "input": "Out of Stock"
          }
      }
  ]
}
```

Jika Amazon SWF dapat memvalidasi data dalam tindakan `RespondDecisionTaskCompleted`, Amazon SWF mengembalikan respons HTTP yang berhasil serupa dengan berikut ini.

```
HTTP/1.1 200 OK
Content-Length: 11
Content-Type: application/json
x-amzn-RequestId: 93cec6f7-0747-11e1-b533-79b402604df1
```

# Timer di Amazon SWF

Dengan pengatur waktu, Anda dapat memberi tahu penentu Anda ketika sejumlah waktu telah berlalu.

Saat merespons tugas keputusan, pengambil keputusan memiliki opsi untuk merespons dengan keputusan `StartTimer`. Keputusan ini menentukan jumlah waktu setelah pengatur waktu akan menyala. Setelah waktu yang ditentukan berlalu, Amazon SWF akan menambahkan kejadian

`TimerFired` ke riwayat eksekusi alur kerja dan menjadwalkan tugas keputusan. Pengambil keputusan kemudian dapat menggunakan informasi ini untuk menginformasikan keputusan lebih lanjut. Salah satu aplikasi umum untuk pengatur waktu adalah untuk menunda pelaksanaan tugas aktivitas. Misalnya, pelanggan mungkin ingin menunda pengiriman barang.

## Membatalkan tugas aktivitas di Amazon SWF

Pembatalan tugas aktivitas menginformasikan penentu untuk mengakhiri kegiatan yang tidak perlu lagi dilakukan. Amazon SWF menggunakan mekanisme pembatalan kooperatif dan tidak secara paksa mengganggu tugas aktivitas yang sedang berjalan. Anda harus memprogram pekerja aktivitas Anda untuk menangani permintaan pembatalan.

Pengambil keputusan dapat memutuskan untuk membatalkan tugas aktivitas saat sedang memproses tugas keputusan. Untuk membatalkan tugas aktivitas, pengambil keputusan menggunakan tindakan `RespondDecisionTaskCompleted` dengan keputusan `RequestCancelActivityTask`.

Jika tugas aktivitas belum diperoleh oleh pekerja aktivitas, layanan akan membatalkan tugas. Perhatikan bahwa ada kondisi balapan potensial di mana pekerja aktivitas dapat memperoleh tugas kapan saja. Jika tugas telah ditetapkan ke pekerja aktivitas, pekerja aktivitas akan diminta untuk membatalkan tugas.

Dalam contoh ini, eksekusi alur kerja dikirimi sinyal untuk membatalkan pesanan.

```
https://swf.us-east-1.amazonaws.com
SignalWorkflowExecution
{"domain": "867530901",
 "workflowId": "20110927-T-1",
 "runId": "9ba33198-4b18-4792-9c15-7181fb3a8852",
 "signalName": "CancelOrder",
 "input": "order 3553"}
```

Jika eksekusi alur kerja menerima sinyal, Amazon SWF mengembalikan respons HTTP yang berhasil serupa dengan berikut ini. Amazon SWF akan menghasilkan tugas keputusan untuk memberi tahu pengambil keputusan untuk memproses sinyal.

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
```

```
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

Saat pengambil keputusan memproses tugas keputusan dan melihat sinyal dalam riwayat, pengambil keputusan mencoba untuk membatalkan aktivitas luar biasa yang memiliki ID aktivitas `ShipOrderActivity0001`. ID aktivitas disediakan dalam riwayat alur kerja dari kejadian tugas aktivitas jadwal.

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken":"12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[{
          "decisionType":"RequestCancelActivityTask",
          "RequestCancelActivityTaskDecisionAttributes":{
              "ActivityID":"ShipOrderActivity0001"
          }
      }
  ]
}
```

Jika Amazon SWF berhasil menerima permintaan pembatalan, Amazon SWF mengembalikan respons HTTP yang berhasil serupa dengan berikut ini:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

Upaya pembatalan dicatat dalam riwayat sebagai kejadian `ActivityTaskCancelRequested`.

Jika tugas berhasil dibatalkan—seperti yang ditunjukkan oleh kejadian `ActivityTaskCanceled`—programkan pengambil keputusan Anda untuk mengambil langkah-langkah yang sesuai yang harus mengikuti pembatalan tugas seperti menutup eksekusi alur kerja.

Jika tugas aktivitas tidak dapat dibatalkan—misalnya, jika tugas selesai, gagal, atau habis waktu daripada dibatalkan—pengambil keputusan Anda harus menerima hasil aktivitas atau melakukan pembersihan atau mitigasi yang diperlukan oleh kasus penggunaan Anda.

Jika tugas aktivitas telah diperoleh oleh pekerja aktivitas, maka permintaan untuk membatalkan dikirimkan melalui mekanisme detak jantung tugas. `RecordActivityTaskHeartbeat` secara berkala dapat melaporkan ke Amazon SWF bahwa tugas masih berlangsung.

Perhatikan bahwa pekerja aktivitas tidak diharuskan untuk detak jantung, meskipun dianjurkan untuk tugas-tugas yang berjalan lama. Pembatalan tugas membutuhkan detak jantung berkala untuk direkam; jika pekerja tidak berdetak, tugas tidak dapat dibatalkan.

Jika pengambil keputusan meminta pembatalan tugas, Amazon SWF menetapkan nilai objek `cancelRequest` ke benar. Objek `cancelRequest` adalah bagian dari objek `ActivityTaskStatus` yang dikembalikan oleh layanan sebagai respons terhadap `RecordActivityTaskHeartbeat`.

Amazon SWF tidak mencegah keberhasilan penyelesaian tugas aktivitas yang pembatalannya telah diminta; terserah pada aktivitas untuk menentukan bagaimana menangani permintaan pembatalan. Bergantung pada kebutuhan Anda, program pekerja aktivitas untuk membatalkan tugas aktivitas atau mengabaikan permintaan pembatalan.

Jika Anda ingin pekerja aktivitas menunjukkan bahwa pekerjaan untuk tugas aktivitas dibatalkan, programlah untuk merespons dengan `RespondActivityTaskCanceled`. Jika Anda ingin pekerja aktivitas menyelesaikan tugas, programlah untuk merespons dengan `RespondActivityTaskCompleted` standar.

Saat Amazon SWF menerima permintaan `RespondActivityTaskCompleted` atau `RespondActivityTaskCanceled`, Amazon SWF memperbarui riwayat eksekusi alur kerja dan menjadwalkan tugas keputusan untuk memberi tahu pengambil keputusan.

Programkan pengambil keputusan untuk memproses tugas keputusan dan mengembalikan keputusan tambahan apa pun. Jika tugas aktivitas berhasil dibatalkan, programkan pengambil keputusan untuk melakukan tugas yang diperlukan untuk melanjutkan atau menutup eksekusi alur kerja. Jika tugas aktivitas tidak berhasil dibatalkan, programkan pengambil keputusan untuk menerima hasil, abaikan hasil, atau jadwalkan pembersihan yang diperlukan.

# Keamanan di Amazon Simple Workflow Service

Bagian ini menyediakan informasi mengenai keamanan Amazon Simple Workflow Service dan autentikasi.

Topik



Amazon SWF menggunakan IAM untuk mengontrol akses ke AWS layanan dan sumber daya lain. Untuk gambaran umum mengenai cara kerja IAM, lihat Gambaran Umum Manajemen Akses di Panduan Pengguna IAM. Untuk ikhtisar kredenal keamanan, lihat Kredensial AWS Keamanan di. Referensi Umum Amazon Web

## Perlindungan data di Amazon Simple Workflow Service

Model tanggung jawab AWS bersama model tanggung berlaku untuk perlindungan data di Amazon Simple Workflow Service. 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. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensi 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 pencatatan 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 Amazon SWF atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau. AWS SDKs 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.

# Enkripsi di Amazon Simple Workflow Service

## Enkripsi saat istirahat

Amazon SWF selalu mengenkripsi data at rest Anda. Data di Amazon Simple Workflow Service dienkripsi saat istirahat menggunakan enkripsi sisi server transparan. Hal ini membantu mengurangi beban operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Dengan enkripsi at rest, Anda dapat membangun aplikasi yang sensitif terhadap keamanan yang memenuhi persyaratan kepatuhan enkripsi dan peraturan

### Enkripsi dalam transit

Semua data yang melewati antara Amazon SWF dan layanan lainnya dienkripsi menggunakan Keamanan Lapisan Pengangkutan (TLS).

# Identity and Access Management di Amazon Simple Workflow Service

Akses ke Amazon SWF memerlukan kredensil yang AWS dapat digunakan untuk mengautentikasi permintaan Anda. Kredensi ini harus memiliki izin untuk mengakses AWS sumber daya, seperti mengambil data peristiwa dari sumber daya lain. AWS Bagian berikut memberikan rincian tentang bagaimana Anda dapat menggunakan AWS Identity and Access Management (IAM) dan Amazon SWF untuk membantu mengamankan sumber daya Anda dengan mengontrol akses ke sumber daya Anda.

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 diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon SWF. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Peran terkait layanan untuk Amazon SWF
- Kebijakan berbasis identitas untuk Amazon SWF
- Kebijakan berbasis sumber daya dalam Amazon SWF
- Bagaimana Amazon Simple Workflow Service bekerja dengan IAM
- Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service
- Prinsip Basic
- Kebijakan IAM Amazon SWF
- Ringkasan API
- Kebijakan Berbasis Tag
- Titik akhir Amazon VPC untuk Amazon SWF
- Memecahkan masalah identitas dan akses Amazon Simple Workflow Service

## 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 Memecahkan masalah identitas dan akses Amazon Simple Workflow Service)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Amazon Simple Workflow Service bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service)

## Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensyal dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensyal. 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 kredensi sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensyal 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 kredensyal 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.

### Kebijakan berbasis identitas

Kebijakan berbasis identitas 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.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

### Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya 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.

Kebijakan berbasis sumber daya merupakan 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 (SCPs) — Tentukan 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 (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- 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.

## Kontrol Akses

Anda dapat memiliki kredensial yang valid untuk mengautentikasi permintaan, tetapi kecuali jika Anda memiliki izin, Anda tidak dapat membuat atau mengakses sumber daya Amazon SWF. Misalnya, Anda harus memiliki izin untuk memanggil, Amazon Simple Notification Service ( AWS Lambda Amazon SNS), dan Amazon Simple Queue Service (Amazon SQS) menargetkan Amazon Simple Service (Amazon SQS) yang terkait dengan aturan Amazon SWF Anda.

Bagian berikut menjelaskan cara mengelola izin untuk Amazon SWF. Kami merekomendasikan agar Anda membaca dulu gambaran umum tersebut.

- Prinsip Basic
- Kebijakan IAM Amazon SWF
- Kebijakan penulisan untuk Amazon SWF

## Tindakan kebijakan untuk Amazon SWF

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 Amazon SWF, lihat Sumber Daya yang Ditentukan oleh Amazon Simple Workflow Service di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon SWF menggunakan awalan berikut sebelum tindakan:

```
swf
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "swf:action1",
      "swf:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Amazon SWF, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service

## Sumber daya kebijakan untuk Amazon SWF

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 sumber daya Amazon SWF dan jenisnya ARNs, lihat Tindakan yang Ditentukan oleh Amazon Simple Workflow Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Sumber Daya yang Ditentukan oleh Amazon Simple Workflow Service.

Untuk melihat contoh kebijakan berbasis identitas Amazon SWF, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service

## Kunci kondisi kebijakan untuk Amazon SWF

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 Amazon SWF, lihat Kunci Kondisi untuk Amazon Simple Workflow Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Sumber Daya yang Ditentukan oleh Amazon Simple Workflow Service.

Untuk melihat contoh kebijakan berbasis identitas Amazon SWF, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service

## ACLs di Amazon SWF

Mendukung ACLs: Tidak

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

## ABAC dengan Amazon SWF

Mendukung ABAC (tag dalam kebijakan): Sebagian

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut tanda. 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 kredensi sementara dengan Amazon SWF

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.

## Izin utama lintas layanan untuk Amazon SWF

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 Amazon SWF

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 Amazon SWF. Edit peran layanan hanya jika Amazon SWF memberikan panduan untuk melakukannya.

## Peran terkait layanan untuk Amazon SWF

Mendukung peran terkait layanan: Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan 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. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

## Kebijakan berbasis identitas untuk Amazon SWF

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. 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.

### Contoh kebijakan berbasis identitas untuk Amazon SWF

Untuk melihat contoh kebijakan berbasis identitas Amazon SWF, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service

## Kebijakan berbasis sumber daya dalam Amazon SWF

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya 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.

## Bagaimana Amazon Simple Workflow Service bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon SWF, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon SWF.

Fitur IAM yang dapat Anda gunakan dengan Amazon Simple Workflow Service

| Fitur IAM | Dukungan Amazon SWF |
| --- | --- |
| Kebijakan berbasis identitas | Ya |

| Fitur IAM | Dukungan Amazon SWF |
| --- | --- |
| Kebijakan berbasis sumber daya | Tidak |
| Tindakan kebijakan | Ya |
| Sumber daya kebijakan | Ya |
| kunci-kunci persyaratan kebijakan (spesifik layanan) | Ya |
| ACLs | Tidak |
| ABAC (tanda dalam kebijakan) | Parsial |
| Kredensial sementara | Ya |
| Izin principal | Ya |
| Peran layanan | Ya |
| Peran terkait layanan | Tidak |

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon SWF dan layanan AWS lainnya dengan sebagian besar fitur IAM, AWS lihat layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

## Contoh kebijakan berbasis identitas untuk Amazon Simple Workflow Service

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon SWF. 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 Amazon SWF, termasuk format ARNs untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk Layanan Alur Kerja Sederhana Amazon di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Amazon SWF
- Mengizinkan pengguna melihat izin mereka sendiri

## Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon SWF 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 Amazon SWF

Untuk mengakses konsol Amazon Simple Workflow Service, Anda harus memiliki seperangkat izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Amazon SWF 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 Amazon SWF, lampirkan juga Amazon *ConsoleAccess* SWF *ReadOnly* AWS atau kebijakan terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambah izin untuk pengguna dalam Panduan Pengguna IAM.

## Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## Prinsip Basic

Kontrol akses Amazon SWF didasarkan terutama pada dua jenis izin:

- Izin sumber daya: Sumber daya Amazon SWF yang dapat diakses pengguna.

  Anda dapat mengekspresikan izin sumber daya hanya untuk domain.
- Izin API: Tindakan Amazon SWF yang dapat dipanggil pengguna.

Pendekatan yang paling sederhana adalah memberikan akses akun penuh—panggil tindakan Amazon SWF di domain apapun—atau tolak akses sepenuhnya. Namun, IAM mendukung

pendekatan yang lebih terperinci untuk kontrol akses yang sering lebih berguna. Misalnya, Anda dapat:

- Mengizinkan pengguna untuk memanggil tindakan Amazon SWF tanpa batasan, tetapi hanya dalam domain tertentu. Anda dapat menggunakan kebijakan tersebut untuk mengizinkan aplikasi alur kerja yang sedang dikembangkan untuk menggunakan tindakan apa pun, tetapi hanya domain "sandbox".
- Mengizinkan pengguna untuk mengakses domain apa pun, namun membatasi cara mereka menggunakan API. Anda dapat menggunakan kebijakan tersebut untuk mengizinkan aplikasi "auditor" dalam memanggil API di domain apa pun, tetapi hanya mengizinkan akses baca.
- Mengizinkan pengguna untuk memanggil hanya serangkaian tindakan terbatas di domain tertentu. Anda dapat menggunakan kebijakan tersebut untuk mengizinkan pemulai alur kerja untuk memanggil hanya tindakan `StartWorkflowExecution` dalam domain tertentu.

Amazon SWF kontrol akses didasarkan pada prinsip-prinsip berikut:

- Keputusan kontrol akses hanya didasarkan pada kebijakan IAM; semua audit dan manipulasi kebijakan dilakukan melalui IAM.
- Model kontrol akses menggunakan deny-by-default kebijakan; akses apa pun yang tidak diizinkan secara eksplisit ditolak.
- Anda mengontrol akses ke sumber daya Amazon SWF dengan melampirkan kebijakan IAM yang sesuai untuk aktor alur kerja.
- Izin sumber daya dapat dinyatakan hanya untuk domain.
- Anda dapat lebih membatasi penggunaan beberapa tindakan dengan menerapkan syarat untuk satu atau lebih parameter.
- Jika Anda memberikan izin untuk menggunakan RespondDecisionTaskCompleted, Anda dapat menyatakan izin untuk daftar keputusan yang disertakan dalam tindakan tersebut.

  Setiap keputusan memiliki satu atau lebih parameter, seperti panggilan API umum. Untuk mengizinkan kebijakan dapat dibaca semudah mungkin, Anda dapat mengekspresikan izin pada keputusan seolah-olah mereka adalah panggilan API yang sebenarnya, termasuk menerapkan syarat untuk beberapa parameter. Jenis-jenis izin ini disebut izin API semu.

Untuk ringkasan parameter API umum dan semu yang dapat dibatasi dengan menggunakan syarat, lihat Ringkasan API.

# Kebijakan IAM Amazon SWF

Kebijakan IAM berisi satu atau beberapa elemen `Statement`, masing-masing berisi satu set elemen yang menentukan kebijakan. Untuk daftar lengkap elemen d an diskusi umum tentang cara membangun kebijakan, lihat Bahasa Kebijakan Akses. Amazon SWF kontrol akses didasarkan pada unsur-unsur berikut:

Efek

(Diperlukan) Efek dari pernyataan: `deny` atau `allow`.

> **Note**
>
> Anda harus mengizinkan akses secara eksplisit; IAM menolak akses secara default.

Sumber daya

(Wajib) Sumber daya — entitas dalam AWS layanan yang dapat berinteraksi dengan pengguna — yang berlaku untuk pernyataan tersebut.

Anda dapat mengekspresikan izin sumber daya hanya untuk domain. Misalnya, kebijakan dapat mengizinkan akses hanya ke domain tertentu di akun Anda. Untuk menyatakan izin domain, setel `Resource` ke Amazon Resource Name (ARN) domain, yang memiliki format "*Region*arn:aws:swf::: /domain/". *AccountID DomainName Region*adalah AWS wilayah, *AccountID* adalah ID akun tanpa tanda hubung, dan *DomainName* merupakan nama domain.

Tindakan

(Wajib) Tindakan yang berlaku untuk pernyataan tersebut, yang Anda rujuk dengan menggunakan format berikut:*serviceId*:*action*. Untuk Amazon SWF, atur *serviceID* ke. `swf` Misalnya, `swf:StartWorkflowExecution` mengacu pada StartWorkflowExecutiontindakan, dan digunakan untuk mengontrol pengguna mana yang diizinkan untuk memulai alur kerja.

Jika Anda memberikan izin untuk menggunakan RespondDecisionTaskCompleted, Anda juga dapat mengontrol akses ke daftar keputusan yang disertakan dengan menggunakan `Action` untuk menyatakan izin untuk API semu. Karena IAM menolak akses secara default, keputusan decider harus diizinkan secara eksplisit atau tidak akan diterima. Anda dapat menggunakan nilai * untuk mengizinkan semua keputusan.

Kondisi

(Opsional) Mengekspresikan kendala pada satu atau lebih parameter tindakan ini, yang membatasi nilai-nilai yang diizinkan.

Tindakan Amazon SWF sering memiliki cakupan yang luas, yang dapat Anda kurangi dengan menggunakan syarat IAM. Misalnya, untuk membatasi daftar tugas mana yang diizinkan untuk diakses PollForActivityTasktindakan, Anda menyertakan `Condition` dan menggunakan `swf:taskList.name` kunci untuk menentukan daftar yang diizinkan.

Anda dapat mengekspresikan kendala untuk entitas berikut.

- Tipe alur kerja. Nama dan versi memiliki kunci terpisah.
- Tipe aktivitas. Nama dan versi memiliki kunci terpisah.
- Daftar tugas.
- Tag. Anda dapat menentukan beberapa tag untuk beberapa tindakan. Dalam hal ini, setiap tag memiliki kunci yang terpisah.

> **Note**
>
> Untuk Amazon SWF, semua nilai merupakan string sehingga Anda membatasi parameter dengan menggunakan operator string seperti `StringEquals`, yang membatasi parameter untuk string tertentu. Namun, operator perbandingan string umum seperti `StringEquals` memerlukan semua permintaan untuk memasukkan parameter. Jika Anda tidak menyertakan parameter secara eksplisit, dan tidak ada nilai default seperti daftar tugas default yang disediakan selama pendaftaran tipe, akses akan ditolak. Hal ini sering berguna untuk menganggap syarat sebagai opsional, sehingga Anda dapat memanggil tindakan tanpa harus menyertakan parameter terkait. Misalnya, Anda mungkin ingin mengizinkan penentu menentukan serangkaian RespondDecisionTaskCompletedkeputusan, tetapi juga mengizinkannya untuk menentukan hanya satu dari mereka untuk panggilan tertentu. Dalam hal ini, Anda membatasi parameter yang sesuai menggunakan operator `StringEqualsIfExists`, yang mengizinkan akses jika parameter memenuhi syarat, tetapi tidak menolak akses jika parameter tidak ada.

Untuk daftar lengkap parameter yang dapat dibatasi dan kunci yang terkait, lihat Ringkasan API.

Bagian berikut memberikan contoh mengenai cara membangun kebijakan Amazon SWF. Untuk detailnya, lihat Syarat String.

## Kebijakan penulisan untuk Amazon SWF

Alur kerja terdiri dari beberapa aktor — aktivitas, penentu, dan sebagainya. Anda dapat mengontrol akses untuk setiap aktor dengan melampirkan kebijakan IAM yang sesuai.

Dengan tindakan berikut, aktor akan diberikan akses akun penuh di semua wilayah:

- Tindakan: `swf:*`
- Sumber daya: `arn:aws:swf:*:123456789012:/domain/*`

Anda dapat menggunakan wildcard untuk memiliki nilai tunggal yang mewakili beberapa sumber daya, tindakan, atau wilayah.

- Wildcard (*****) pertama dalam **`Resource`** nilai menunjukkan bahwa izin sumber daya berlaku untuk semua wilayah.

  Untuk membatasi izin untuk satu wilayah, gantilah wildcard dengan string wilayah yang sesuai, seperti us-east-1.
- Wildcard kedua (`*`) dalam nilai `Resource` mengizinkan aktor untuk mengakses salah satu domain akun di wilayah tertentu.
- Wildcard (`*`) dalam nilai `Action` mengizinkan aktor untuk memanggil tindakan Amazon SWF.

Untuk detail tentang cara menggunakan wildcard, lihat Deskripsi Elemen

Perizinan Domain

Untuk membatasi alur kerja departemen ke domain tertentu, Anda dapat memberikan izin yang memungkinkan aktor untuk memanggil tindakan apa pun, tetapi hanya untuk departemen tertentu.

Untuk mendapatkan akses aktor ke lebih dari satu domain, berikan izin untuk setiap domain sebagai daftar Pernyataan:

- Tindakan: `swf:*`
- Sumber daya: `arn:aws:swf:*:123456789012:/domain/department1`
- Sumber daya: `arn:aws:swf:*:123456789012:/domain/department2`

Anda dapat mengizinkan aktor untuk menggunakan tindakan Amazon SWF apa pun di domain `department1` dan`department2`. Anda juga dapat menggunakan wildcard sesekali untuk mewakili beberapa domain.

Izin dan Kendala API

Anda mengontrol tindakan mana yang dapat digunakan aktor dengan menentukan tindakan dalam `Action` elemen.

Dengan tindakan berikut, seorang aktor hanya dapat menelepon `StartWorkflowExecution` untuk memulai alur kerja. Itu tidak bisa menggunakan tindakan lain.

- Tindakan: `swf:StartWorkflowExecution`

Ketentuan

Anda dapat secara opsional membatasi nilai parameter tindakan yang diizinkan dengan menggunakan elemen. `Condition`

Untuk membatasi alur kerja yang dapat dimulai aktor, batasi satu atau lebih nilai `StartWorkflowExecution` parameter, sebagai berikut:

```
"Condition" : {
   "StringEquals" : {
      "swf:workflowType.name" : "workflow1",
      "swf:workflowType.version" : "version2"
    }
}
```

Seorang aktor dengan kendala sebelumnya hanya `version2` dapat menjalankan `workflow1` dan kedua parameter harus disertakan dalam permintaan.

Anda dapat membatasi parameter tanpa harus memasukkanya dalam permintaan dengan menggunakan operator `StringEqualsIfExists`, sebagai berikut:

```
"Condition" : {
   "StringEqualsIfExists" : { "swf:taskList.name" : "task_list_name" }
}
```

Aktor dengan kebijakan sebelumnya dapat menentukan daftar tugas secara opsional saat memulai eksekusi alur kerja.

Anda dapat membatasi daftar tag untuk beberapa tindakan. Setiap tag memiliki kunci terpisah, jadi Anda gunakan `swf:tagList.member.0` untuk membatasi tag pertama dalam daftar, `swf:tagList.member.1` untuk membatasi tag kedua dalam daftar, dan seterusnya, hingga maksimum 5.

Anda harus berhati-hati bagaimana Anda membatasi daftar tag. Misalnya, kondisi berikut tidak disarankan.

Kondisi berikut tidak disarankan karena memungkinkan Anda untuk secara opsional menentukan salah satu `some_ok_tag` atau`another_ok_tag`. Namun, Kondisi hanya membatasi elemen pertama dari daftar tag. Daftar dapat memiliki elemen tambahan dengan nilai arbitrer yang semuanya diizinkan karena kondisi tidak menerapkan kondisi apa pun`swf:tagList.member.1`,`swf:tagList.member.2`, dan seterusnya.

```
// Example to illustrate an insecure Condition
"Condition" : {
   "StringEqualsIfExists" : {
      "swf:tagList.member.0" : "some_ok_tag", "another_ok_tag"
   }
}
```

Salah satu cara untuk mengatasi masalah sebelumnya adalah dengan melarang penggunaan daftar tag.

Kebijakan berikut memastikan bahwa hanya `some_ok_tag` atau `another_ok_tag` yang diperbolehkan dengan mewajibkan daftar hanya memiliki satu elemen.

```
"Condition" : {
   "StringEqualsIfExists" : {
      "swf:tagList.member.0" : "some_ok_tag", "another_ok_tag"
    },
    "Null" : { "swf:tagList.member.1" : "true" }
}
```

Izin dan Kendala API Semu

Untuk membatasi keputusan yang tersedia`RespondDecisionTaskCompleted`, Anda harus terlebih dahulu mengizinkan aktor untuk menelepon`RespondDecisionTaskCompleted`. Anda kemudian menyatakan izin untuk anggota API semu yang sesuai menggunakan sintaks yang sama seperti untuk API biasa, sebagai berikut:

- Pernyataan 1

  Sumber daya: `arn:aws:swf:*:123456789012:/domain/*`

  Tindakan: `swf:RespondDecisionTaskCompleted`

- Pernyataan 2

  Sumber daya: *

  Tindakan: `swf:ScheduleActivityTask`

  Kondisi: `"StringEquals" : { "swf:activityType.name" : "SomeActivityType" }`

Yang pertama `Statement` memungkinkan aktor untuk meneleponRespondDecisionTaskCompleted. Pernyataan kedua memungkinkan aktor untuk menggunakan `ScheduleActivityTask` keputusan untuk mengarahkan Amazon SWF untuk menjadwalkan tugas kegiatan. Untuk mengizinkan semua keputusan, ganti “swf:ScheduleActivityTask" dengan “swf: *”.

Anda dapat menggunakan operator Syarat untuk membatasi parameter seperti dengan API umum. `StringEquals`Operator dalam contoh sebelumnya `Condition` memungkinkan `RespondDecisionTaskCompleted` untuk menjadwalkan tugas aktivitas untuk `SomeActivityType` aktivitas tersebut, dan harus menjadwalkan tugas itu. Jika Anda ingin mengizinkan `RespondDecisionTaskCompleted` untuk menggunakan nilai parameter tetapi tidak memerlukan untuk melakukannya, Anda sebaliknya dapat menggunakan operator `StringEqualsIfExists`.

## AWS kebijakan terkelola: SimpleWorkflowFullAccess

Anda dapat melampirkan kebijakan `SimpleWorkflowFullAccess` ke identitas IAM Anda.

Kebijakan ini menyediakan akses penuh ke layanan konfigurasi Amazon SWF.

## Layanan Model Pembatasan pada Kebijakan IAM

Anda harus mempertimbangkan kendala model layanan saat membuat kebijakan IAM. Mungkin untuk membuat kebijakan valid IAM secara sintaksis yang mewakili permintaan Amazon SWF yang tidak valid; permintaan yang diizinkan dalam hal kontrol akses masih bisa gagal karena merupakan permintaan yang tidak valid.

Misalnya, model layanan Amazon SWF tidak mengizinkan `tagFilter` parameter `typeFilter` dan digunakan dalam permintaan yang samaListOpenWorkflowExecutions. Kondisi berikut akan memungkinkan panggilan yang akan ditolak oleh layanan—dengan melempar `ValidationException` —sebagai permintaan yang tidak valid:

```
"Condition" : {
   "StringEquals" : {
      "swf:typeFilter.name" : "workflow_name",
      "swf:typeFilter.version" : "workflow_version",
      "swf:tagFilter.tag" : "some_tag"
    }
}
```

## Ringkasan API

Bagian ini menjelaskan secara singkat mengenai cara Anda menggunakan kebijakan IAM untuk mengontrol seorang aktor untuk dapat menggunakan setiap API dan API semu untuk mengakses sumber daya Amazon SWF.

- Untuk semua tindakan kecuali `RegisterDomain` dan `ListDomains`, Anda dapat mengizinkan atau menolak akses ke salah satu atau semua domain akun dengan menyatakan izin untuk sumber daya domain.
- Anda dapat mengizinkan atau menolak izin untuk setiap anggota API umum dan, jika Anda memberikan izin untuk memanggil `RespondDecisionTaskCompleted`, semua anggota dari API semu.
- Anda dapat menggunakan Syarat untuk membatasi nilai-nilai yang diizinkan beberapa parameter.

Bagian berikut mendaftar parameter yang dapat dibatasi untuk setiap anggota API umum dan semu serta memberikan kunci terkait, dan memperhatikan pembatasan pada Anda untuk dapat mengontrol akses domain.

### API Reguler

Bagian ini berisi daftar anggota API umum, dan menjelaskan secara singkat parameter yang dapat dibatasi dan kunci yang terkait. Bagian ini juga mencatat batasan apa pun mengenai cara Anda untuk dapat mengontrol akses domain.

`CountClosedWorkflowExecutions`

- `tagFilter.tag` – Kendala String. Kuncinya adalah `swf:tagFilter.tag`
- `typeFilter.name` – Kendala String. Kuncinya adalah `swf:typeFilter.name`.
- `typeFilter.version` – Kendala String. Kuncinya adalah `swf:typeFilter.version`.

> **Note**
>
> `CountClosedWorkflowExecutions` membutuhkan `typeFilter` dan `tagFilter` untuk menjadi saling eksklusif.

### `CountOpenWorkflowExecutions`

- `tagFilter.tag` – Kendala String. Kuncinya adalah `swf:tagFilter.tag`
- `typeFilter.name` – Kendala String. Kuncinya adalah `swf:typeFilter.name`.
- `typeFilter.version` – Kendala String. Kuncinya adalah `swf:typeFilter.version`.

> **Note**
>
> `CountOpenWorkflowExecutions` membutuhkan `typeFilter` dan `tagFilter` untuk menjadi saling eksklusif.

### `CountPendingActivityTasks`

- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.

### `CountPendingDecisionTasks`

- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.

### `DeleteActivityType`

- `activityType.name` – Kendala String. Kuncinya adalah `swf:activityType.name`.
- `activityType.version` – Kendala String. Kuncinya adalah `swf:activityType.version`.

### DeprecateActivityType

- `activityType.name` – Kendala String. Kuncinya adalah `swf:activityType.name`.
- `activityType.version` – Kendala String. Kuncinya adalah `swf:activityType.version`.

### DeprecateDomain

- Anda tidak dapat membatasi parameter tindakan ini.

### DeleteWorkflowType

- `workflowType.name` – Kendala String. Kuncinya adalah `swf:workflowType.name`.
- `workflowType.version` – Kendala String. Kuncinya adalah `swf:workflowType.version`.

### DeprecateWorkflowType

- `workflowType.name` – Kendala String. Kuncinya adalah `swf:workflowType.name`.
- `workflowType.version` – Kendala String. Kuncinya adalah `swf:workflowType.version`.

### DescribeActivityType

- `activityType.name` – Kendala String. Kuncinya adalah `swf:activityType.name`.
- `activityType.version` – Kendala String. Kuncinya adalah `swf:activityType.version`.

### DescribeDomain

- Anda tidak dapat membatasi parameter tindakan ini.

### DescribeWorkflowExecution

- Anda tidak dapat membatasi parameter tindakan ini.

### DescribeWorkflowType

- `workflowType.name` – Kendala String. Kuncinya adalah `swf:workflowType.name`.
- `workflowType.version` – Kendala String. Kuncinya adalah `swf:workflowType.version`.

### GetWorkflowExecutionHistory

- Anda tidak dapat membatasi parameter tindakan ini.

### ListActivityTypes

- Anda tidak dapat membatasi parameter tindakan ini.

### ListClosedWorkflowExecutions

- `tagFilter.tag` – Kendala String. Kuncinya adalah `swf:tagFilter.tag`
- `typeFilter.name` – Kendala String. Kuncinya adalah `swf:typeFilter.name`.
- `typeFilter.version` – Kendala String. Kuncinya adalah `swf:typeFilter.version`.

> **Note**
>
> `ListClosedWorkflowExecutions` membutuhkan `typeFilter` dan `tagFilter` untuk menjadi saling eksklusif.

### ListDomains

- Anda tidak dapat membatasi parameter tindakan ini.

### ListOpenWorkflowExecutions

- `tagFilter.tag` – Kendala String. Kuncinya adalah `swf:tagFilter.tag`
- `typeFilter.name` – Kendala String. Kuncinya adalah `swf:typeFilter.name`.
- `typeFilter.version` – Kendala String. Kuncinya adalah `swf:typeFilter.version`.

> **Note**
>
> `ListOpenWorkflowExecutions` membutuhkan `typeFilter` dan `tagFilter` untuk menjadi saling eksklusif.

### ListWorkflowTypes

- Anda tidak dapat membatasi parameter tindakan ini.

### PollForActivityTask

- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.

### PollForDecisionTask

- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.

### RecordActivityTaskHeartbeat

- Anda tidak dapat membatasi parameter tindakan ini.

### RegisterActivityType

- `defaultTaskList.name` – Kendala String. Kuncinya adalah `swf:defaultTaskList.name`.
- `name` – Kendala String. Kuncinya adalah `swf:name`.
- `version` – Kendala String. Kuncinya adalah `swf:version`.

### RegisterDomain

- `name` – Nama domain yang terdaftar tersedia sebagai sumber daya dari tindakan ini.

### RegisterWorkflowType

- `defaultTaskList.name` – Kendala String. Kuncinya adalah `swf:defaultTaskList.name`.
- `name` – Kendala String. Kuncinya adalah `swf:name`.
- `version` – Kendala String. Kuncinya adalah `swf:version`.

### RequestCancelWorkflowExecution

- Anda tidak dapat membatasi parameter tindakan ini.

### RespondActivityTaskCanceled

- Anda tidak dapat membatasi parameter tindakan ini.

### RespondActivityTaskCompleted

- Anda tidak dapat membatasi parameter tindakan ini.

### RespondActivityTaskFailed

- Anda tidak dapat membatasi parameter tindakan ini.

### RespondDecisionTaskCompleted

- `decisions.member.N` – Dibatasi secara tidak langsung melalui izin API semu. Untuk detail selengkapnya, lihat API Semu.

### SignalWorkflowExecution

- Anda tidak dapat membatasi parameter tindakan ini.

### StartWorkflowExecution

- `tagList.member.0` – Kendala String. Kuncinya adalah `swf:tagList.member.0`
- `tagList.member.1` – Kendala String. Kuncinya adalah `swf:tagList.member.1`
- `tagList.member.2` – Kendala String. Kuncinya adalah `swf:tagList.member.2`
- `tagList.member.3` – Kendala String. Kuncinya adalah `swf:tagList.member.3`
- `tagList.member.4` – Kendala String. Kuncinya adalah `swf:tagList.member.4`
- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.
- `workflowType.name` – Kendala String. Kuncinya adalah `swf:workflowType.name`.
- `workflowType.version` – Kendala String. Kuncinya adalah `swf:workflowType.version`.

> **Note**
>
> Anda tidak dapat membatasi lebih dari lima tag.

### `TerminateWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

## API Semu

Bagian ini berisi daftar anggota API semu, yang mewakili keputusan yang disertakan dalam `RespondDecisionTaskCompleted`. Jika Anda telah memberikan izin untuk menggunakan `RespondDecisionTaskCompleted`, kebijakan Anda dapat menyatakan izin untuk anggota API ini dengan cara yang sama seperti API umum. Anda dapat lebih membatasi beberapa anggota API semu dengan menetapkan syarat pada satu atau lebih parameter. Bagian ini berisi daftar anggota API semu, dan menjelaskan secara singkat parameter yang dapat dibatasi dan kunci yang terkait.

> **Note**
>
> Kunci `aws:SourceIP`, `aws:UserAgent`, dan `aws:SecureTransport` tidak tersedia untuk API semu. Jika kebijakan keamanan yang Anda inginkan memerlukan kunci ini untuk mengontrol akses ke API semu, Anda dapat menggunakannya dengan tindakan `RespondDecisionTaskCompleted`.

### `CancelTimer`

- Anda tidak dapat membatasi parameter tindakan ini.

### `CancelWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

### `CompleteWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

`ContinueAsNewWorkflowExecution`

- `tagList.member.0` – Kendala String. Kuncinya adalah `swf:tagList.member.0`
- `tagList.member.1` – Kendala String. Kuncinya adalah `swf:tagList.member.1`
- `tagList.member.2` – Kendala String. Kuncinya adalah `swf:tagList.member.2`
- `tagList.member.3` – Kendala String. Kuncinya adalah `swf:tagList.member.3`
- `tagList.member.4` – Kendala String. Kuncinya adalah `swf:tagList.member.4`
- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.
- `workflowTypeVersion` – Kendala String. Kuncinya adalah `swf:workflowTypeVersion`.

> **Note**
>
> Anda tidak dapat membatasi lebih dari lima tag.

`FailWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

`RecordMarker`

- Anda tidak dapat membatasi parameter tindakan ini.

`RequestCancelActivityTask`

- Anda tidak dapat membatasi parameter tindakan ini.

`RequestCancelExternalWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

`ScheduleActivityTask`

- `activityType.name` – Kendala String. Kuncinya adalah `swf:activityType.name`.
- `activityType.version` – Kendala String. Kuncinya adalah `swf:activityType.version`.

- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.

`SignalExternalWorkflowExecution`

- Anda tidak dapat membatasi parameter tindakan ini.

`StartChildWorkflowExecution`

- `tagList.member.0` – Kendala String. Kuncinya adalah `swf:tagList.member.0`
- `tagList.member.1` – Kendala String. Kuncinya adalah `swf:tagList.member.1`
- `tagList.member.2` – Kendala String. Kuncinya adalah `swf:tagList.member.2`
- `tagList.member.3` – Kendala String. Kuncinya adalah `swf:tagList.member.3`
- `tagList.member.4` – Kendala String. Kuncinya adalah `swf:tagList.member.4`
- `taskList.name` – Kendala String. Kuncinya adalah `swf:taskList.name`.
- `workflowType.name` – Kendala String. Kuncinya adalah `swf:workflowType.name`.
- `workflowType.version` – Kendala String. Kuncinya adalah `swf:workflowType.version`.

> **Note**
>
> Anda tidak dapat membatasi lebih dari lima tag.

`StartTimer`

- Anda tidak dapat membatasi parameter tindakan ini.

## Kebijakan Berbasis Tag

Amazon SWF mendukung kebijakan berdasarkan tag. Misalnya, Anda dapat membatasi domain Amazon SWF yang menyertakan tag dengan `environment` kunci dan `production` nilai dengan kondisi berikut:

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/environment": "production"}
}
```

Untuk informasi selengkapnya mengenai penandaan, lihat:

- Tag di Amazon SWF
- Mengontrol Akses Menggunakan Tag IAM

## Titik akhir Amazon VPC untuk Amazon SWF

> **Note**
>
> AWS PrivateLink dukungan saat ini tersedia di AWS Top Secret - Timur, Wilayah AWS Rahasia, dan Wilayah Tiongkok saja.

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meng-host AWS sumber daya, Anda dapat membuat sambungan antara alur kerja Amazon VPC dan Amazon Simple Workflow Service. Anda dapat menggunakan koneksi ini dengan alur kerja Amazon SWF Anda tanpa melintasi internet publik.

Amazon VPC memungkinkan Anda meluncurkan AWS sumber daya di jaringan virtual khusus. Anda dapat menggunakan VPC untuk mengendalikan pengaturan jaringan, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Untuk informasi selengkapnya VPCs, lihat Panduan Pengguna Amazon VPC.

Untuk menghubungkan VPC Amazon Anda ke Amazon SWF, Anda harus terlebih dahulu menentukan titik akhir VPC antarmuka, yang memungkinkan Anda menghubungkan VPC Anda ke yang lain. Layanan AWS Titik akhir memberikan konektivitas yang dapat andal, dapat diskalakan, tanpa memerlukan gateway internet, instans terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi selengkapnya, lihat VPC Endpoint Antarmuka (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.

### Membuat Titik Akhir

Anda dapat membuat titik akhir Amazon SWF di VPC menggunakan, AWS Command Line Interface (AWS CLI) Konsol Manajemen AWS, SDK, Amazon SWF AWS API, atau. CloudFormation

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan konsol Amazon VPC atau AWS CLI, lihat Membuat Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

> **Note**
>
> Saat Anda membuat titik akhir, tentukan Amazon SWF sebagai layanan yang Anda inginkan untuk disambungkan oleh VPC Anda. Di konsol Amazon VPC, nama layanan bervariasi berdasarkan Wilayah AWS . Misalnya, di AWS Top Secret - East Region, nama layanan untuk Amazon SWF adalah com.amazonaws. us-iso-east-1.swf.

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan CloudFormation, lihat VPCEndpoint sumber daya AWS:EC2:: di CloudFormation Panduan Pengguna.

### Kebijakan Amazon VPC Endpoint

Untuk mengontrol akses konektivitas ke Amazon SWF, Anda dapat melampirkan kebijakan endpoint AWS Identity and Access Management (IAM) saat membuat endpoint Amazon VPC. Anda dapat membuat aturan IAM kompleks dengan melampirkan beberapa kebijakan titik akhir. Untuk informasi selengkapnya, lihat:

- Kebijakan Titik Akhir Amazon Virtual Private Cloud untuk Amazon SWF
- Mengontrol Akses ke Layanan dengan Titik Akhir VPC

### Kebijakan Titik Akhir Amazon Virtual Private Cloud untuk Amazon SWF

Anda dapat membuat kebijakan titik akhir Amazon VPC untuk Amazon SWF di mana Anda menentukan yang berikut ini:

- Kepala sekolah yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Contoh berikut menambahkan peran IAM tertentu ke kebijakan:

```
"Principal": {
   "AWS": "arn:aws:iam::123456789012:role/MyRole"
}
```

- Untuk informasi selengkapnya tentang membuat kebijakan titik akhir, lihat Mengontrol Akses ke Layanan dengan Titik Akhir VPC.

- Untuk informasi tentang cara menggunakan IAM untuk mengontrol akses ke sumber daya Amazon SWF Anda AWS dan Amazon, lihat. Identity and Access Management di Amazon Simple Workflow Service

# Memecahkan masalah identitas dan akses Amazon Simple Workflow Service

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon SWF dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon SWF
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon SWF saya

## Saya tidak berwenang untuk melakukan tindakan di Amazon SWF

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 `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `swf:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 swf:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan Mateo harus diperbarui untuk memungkinkannya mengakses *`my-example-widget`* sumber daya menggunakan `swf:`*`GetWidget`* tindakan.

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 bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon SWF.

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 Amazon SWF. 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 Amazon SWF 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 (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon SWF mendukung fitur-fitur ini, lihat. Bagaimana Amazon Simple Workflow Service 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.

# Pembuatan Log dan Pemantauan

Bagian ini memberikan informasi tentang pencatatan dan pemantauan Amazon SWF.

Topik

- Metrik Amazon SWF untuk CloudWatch
- Melihat Metrik Amazon SWF untuk menggunakan CloudWatch Konsol Manajemen AWS
- Merekam panggilan API dengan AWS CloudTrail
- EventBridge untuk perubahan status eksekusi Amazon SWF
- Menggunakan Notifikasi Pengguna AWS dengan Amazon Simple Workflow Service

## Metrik Amazon SWF untuk CloudWatch

Amazon SWF kini menyediakan metrik CloudWatch yang dapat Anda gunakan untuk melacak alur kerja dan aktivitas serta menetapkan alarm pada nilai ambang batas yang Anda pilih. Anda dapat melihat metrik menggunakan. Konsol Manajemen AWS Untuk informasi selengkapnya, lihat Melihat Metrik Amazon SWF untuk menggunakan CloudWatch Konsol Manajemen AWS.

Topik

- Pelaporan Unit untuk Metrik Amazon SWF
- Metrik Kejadian API dan Keputusan
- Metrik Amazon SWF
- Amazon SWF nama dan dimensi sumber daya non-ASCII CloudWatch

### Pelaporan Unit untuk Metrik Amazon SWF

Metrik yang Melaporkan Interval Waktu

Beberapa metrik Amazon SWF untuk CloudWatch adalah interval waktu, selalu diukur dalam milidetik. CloudWatch Unit ini dilaporkan sebagai`Time`. Metrik ini umumnya sesuai dengan tahapan eksekusi alur kerja Anda yang dapat mengatur alur kerja dan batas waktu aktivitas, dan memiliki nama yang serupa.

Misalnya, metrik `DecisionTaskStartToCloseTime` mengukur waktu yang dibutuhkan untuk tugas keputusan untuk menyelesaikan setelah mulai mengeksekusi, yang merupakan periode waktu yang sama tempat Anda dapat mengatur nilai `DecisionTaskStartToCloseTimeout`.

Untuk diagram masing-masing tahapan alur kerja ini dan untuk mempelajari ketika terjadi selama alur kerja dan siklus hidup aktivitas, lihat Tipe Batas Waktu Amazon SWF.

Metrik yang Melaporkan Jumlah

Beberapa metrik Amazon SWF untuk CloudWatch melaporkan hasil sebagai hitungan. Misalnya, `WorkflowsCanceled`, mencatat hasil sebagai satu atau nol, yang menunjukkan apakah alur kerja dibatalkan atau tidak. Nilai nol tidak menunjukkan bahwa metrik tidak dilaporkan, hanya saja syarat yang dijelaskan metrik tidak terjadi.

Beberapa metrik Amazon SWF untuk laporan CloudWatch tersebut CloudWatch adalah `Count` hitungan per detik. Misalnya,`ProvisionedRefillRate`, yang dilaporkan sebagai `Count` in CloudWatch, mewakili *`Count`*tingkat permintaan per detik.

Untuk jumlah metrik, minimum dan maksimum akan selalu baik nol atau satu, tetapi rata-rata akan menjadi nilai mulai dari nol ke satu.

## Metrik Kejadian API dan Keputusan

Anda dapat memantau peristiwa API dan Keputusan CloudWatch untuk memberikan wawasan tentang penggunaan dan kapasitas Anda. Lihat decider dalam bagian Konsep alur kerja dasar di Amazon SWF, dan topik Keputusan di Referensi API Amazon Simple Workflow Service.

Anda juga dapat memantau batas-batas ini untuk alarm ketika Anda mendekati batas throttling Amazon SWF Anda. Lihat Kuota throttling Amazon SWF untuk deskripsi batas ini dan pengaturan default mereka. Batas ini dirancang untuk mencegah alur kerja yang salah mengkonsumsi sumber daya sistem yang berlebihan. Untuk meminta peningkatan batas Anda, lihat: ???.

Sebagai praktik terbaik, Anda harus mengonfigurasi CloudWatch alarm sekitar 60% dari kapasitas API atau peristiwa keputusan Anda. Tindakan ini akan mengizinkan Anda untuk menyesuaikan alur kerja, atau meminta peningkatan batas layanan, sebelum throttling Amazon SWF diaktifkan. Tergantung pada lonjakan panggilan, Anda dapat mengkonfigurasi alarm yang berbeda untuk memberitahu ketika Anda mendekati Kuota Layanan Anda:

- Jika lalu lintas Anda memiliki lonjakan yang signifikan, atur alarm pada 60% dari batas `ProvisionedBucketSize`.

- Jika panggilan Anda memiliki tingkat yang relatif stabil, atur alarm pada 60% dari batas `ProvisionedRefillRate` untuk API terkait dan kejadian keputusan Anda.

## Metrik Amazon SWF

Metrik berikut tersedia untuk Amazon SWF:

| Metrik | Deskripsi |
| --- | --- |
| `DecisionTaskScheduleToStartTime` | Interval waktu, dalam milidetik, antara waktu tugas keputusan dijadwalkan dan ketika diambil oleh pekerja dan dimulai.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |
| `DecisionTaskStartToCloseTime` | Interval waktu, dalam milidetik, antara waktu tugas keputusan dimulai dan ketika ditutup.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |
| `DecisionTasksCompleted` | Jumlah tugas keputusan yang telah selesai.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `PendingTasks` | Jumlah tugas tertunda dalam interval 1 menit untuk Daftar Tugas tertentu. |

| Metrik | Deskripsi |
| --- | --- |
| | CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, TaskListName`<br><br>Statistik yang valid: `Sum` |
| `StartedDecisionTasksTimedOutOnClose` | Jumlah tugas keputusan yang dimulai tapi berakhir pada penutupan.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `WorkflowStartToCloseTime` | Waktu, dalam milidetik, antara waktu alur kerja dimulai dan saat alur kerja ditutup.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |
| `WorkflowsCanceled` | Jumlah alur kerja yang dibatalkan.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |

| Metrik | Deskripsi |
| --- | --- |
| `WorkflowsCompleted` | Jumlah alur kerja yang selesai.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `WorkflowsContinuedAsNew` | Jumlah alur kerja yang berlanjut sebagai baru.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `WorkflowsFailed` | Jumlah alur kerja yang gagal.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `WorkflowsTerminated` | Jumlah alur kerja yang diakhiri.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Cause, Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |

| Metrik | Deskripsi |
| --- | --- |
| `WorkflowsTimedOut` | Jumlah alur kerja yang waktunya habis, karena alasan apa pun.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, WorkflowTypeName, WorkflowTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `ActivityTaskScheduleToCloseTime` | Interval waktu, dalam milidetik, antara waktu ketika aktivitas dijadwalkan dan ketika ditutup.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |
| `ActivityTaskScheduleToStartTime` | Interval waktu, dalam milidetik, antara waktu ketika tugas aktivitas dijadwalkan dan ketika dimulai.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |
| `ActivityTaskStartToCloseTime` | Interval waktu, dalam milidetik, antara waktu ketika tugas aktivitas dimulai dan ketika ditutup.<br><br>CloudWatch Unit: `Time`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Average, Minimum, Maximum` |

| Metrik | Deskripsi |
| --- | --- |
| `ActivityTasksCanceled` | Jumlah tugas aktivitas yang dibatalkan.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `ActivityTasksCompleted` | Jumlah tugas aktivitas yang selesai.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `ActivityTasksFailed` | Jumlah tugas aktivitas yang gagal.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Sum` |
| `ScheduledActivityTasksTimedOutOnClose` | Jumlah tugas aktivitas yang dijadwalkan tetapi waktunya habis saat penutupan.<br><br>CloudWatch Unit: `Count`<br><br>Dimensi: `Domain, ActivityTypeName, ActivityTypeVersion`<br><br>Statistik yang valid: `Sum` |

| Metrik | Deskripsi |
| --- | --- |
| ScheduledActivityTasksTimedOutOnStart | Jumlah tugas aktivitas yang dijadwalkan tetapi waktunya habis saat mulai.<br><br>CloudWatch Unit: Count<br><br>Dimensi: Domain, ActivityTypeName, ActivityTypeVersion<br><br>Statistik yang valid: Sum |
| StartedActivityTasksTimedOutOnClose | Jumlah tugas aktivitas yang dimulai tetapi waktunya habis saat penutupan.<br><br>CloudWatch Unit: Count<br><br>Dimensi: Domain, ActivityTypeName, ActivityTypeVersion<br><br>Statistik yang valid: Sum |
| StartedActivityTasksTimedOutOnHeartbeat | Jumlah tugas aktivitas yang dimulai tetapi waktunya habis karena batas waktu detak jantung.<br><br>CloudWatch Unit: Count<br><br>Dimensi: Domain, ActivityTypeName, ActivityTypeVersion<br><br>Statistik yang valid: Sum |
| ThrottledEvents | Jumlah permintaan yang mengalami throttling.<br><br>CloudWatch Unit: Count<br><br>Dimensi: APIName, DecisionName, ThrottlingScope<br><br>Statistik yang valid: Sum |

| Metrik | Deskripsi |
| --- | --- |
| `ProvisionedBucketSize` | Jumlah permintaan yang tersedia per detik.<br>Dimensi: `APIName, DecisionName`<br>Statistik yang valid: `Minimum` |
| `ConsumedCapacity` | Jumlah permintaan per detik.<br>CloudWatch Unit: `Count`<br>Dimensi: `APIName, DecisionName`<br>Statistik yang valid: `Sum` |
| `ConsumedLimit` | Jumlah batas umum yang telah dikonsumsi.<br>Dimensi: `GeneralLimitType` |
| `ProvisionedRefillRate` | Jumlah permintaan per detik yang diizinkan masuk ke dalam bucket.<br>Dimensi: `APIName, DecisionName`<br>Statistik yang valid: `Minimum` |
| `ProvisionedLimit` | Jumlah batas umum yang diberikan ke akun.<br>Dimensi: `GeneralLimitType` |

| Dimensi | Deskripsi |
| --- | --- |
| `Domain` | Memfilter data ke domain Amazon SWF tempat alur kerja atau aktivitas berjalan. |
| `ActivityTypeName` | Memfilter data ke nama tipe aktivitas. |
| `ActivityTypeVersion` | Memfilter data ke versi tipe aktivitas. |

| Dimensi | Deskripsi |
| --- | --- |
| `WorkflowTypeName` | Memfilter data ke nama tipe alur kerja untuk eksekusi alur kerja ini. |
| `WorkflowTypeVersion` | Memfilter data ke versi tipe alur kerja untuk eksekusi alur kerja ini. |
| `APIName` | Memfilter data ke API dari nama API yang ditentukan. |
| `DecisionName` | Memfilter data ke nama Keputusan yang ditentukan. |
| `TaskListName` | Memfilter data ke nama Daftar Tugas yang ditentukan. |
| `TaskListClassification` | Memfilter data ke klasifikasi daftar tugas. Nilai "D" untuk Daftar Tugas Keputusan dan "A" untuk Daftar Tugas Aktivitas. |
| `ThrottlingScope` | Memfilter data ke lingkup pelambatan yang ditentukan. Nilai adalah “Akun” jika melebihi kuota tingkat akun, atau “Alur Kerja” saat melebihi kuota tingkat alur kerja. |

## Amazon SWF nama dan dimensi sumber daya non-ASCII CloudWatch

Amazon SWF memungkinkan karakter non-ASCII dalam nama sumber daya seperti dan. TaskList DomainName Namun, nilai dimensi CloudWatch metrik hanya dapat berisi karakter ASCII yang dapat dicetak. Untuk memastikan bahwa Amazon SWF menggunakan nilai dimensi yang kompatibel dengan CloudWatch persyaratan, nama sumber daya Amazon SWF yang tidak memenuhi persyaratan ini akan dikonversi dan akan memiliki checksum yang ditambahkan sebagai berikut:

- Setiap karakter non-ASCII diganti dengan ?.
- String masukan atau string yang diubah akan, jika perlu, terpotong. Ini memastikan bahwa ketika checksum ditambahkan, panjang string baru tidak akan melebihi maksimum. CloudWatch
- Karena karakter non-ASCII dikonversi?, beberapa nilai dimensi CloudWatch metrik yang berbeda sebelum konversi mungkin tampak sama setelah konversi. Untuk membantu membedakannya, garis bawah (_) diikuti oleh 16 karakter pertama SHA256 checksum dari nama sumber daya asli ditambahkan ke nama sumber daya.

Contoh perubahan:

- `test àpple` akan diubah menjadi `test ?pple_82cc5b8e3a771d12`
- `àòà` akan diubah menjadi `???_2fec5edbb2c05c22`.
- TaskList Nama-nama `àpplé` dan keduanya `âpplè` akan dikonversi menjadi`?ppl?`, dan akan identik. Menambahkan checksum mengembalikan nilai-nilai yang berbeda, `?ppl?_f39a36df9d85a69d` dan `?ppl?_da3efb4f11dd0f7f`.

> **Tip**
>
> Anda dapat membuat SHA256 checksum Anda sendiri. Misalnya, untuk menggunakan alat baris perintah `shasum`:
> echo -n "<the original resource name>" | shasum -a 256 | cut -c1-16

## Melihat Metrik Amazon SWF untuk menggunakan CloudWatch Konsol Manajemen AWS

Amazon CloudWatch menyediakan sejumlah metrik yang dapat dilihat untuk alur kerja dan aktivitas Amazon SWF. Anda dapat melihat metrik dan mengatur alarm untuk eksekusi alur kerja Amazon SWF menggunakan Konsol Manajemen AWS. Anda harus masuk ke konsol tersebut untuk melanjutkan.

Untuk deskripsi masing-masing metrik yang tersedia, lihat Metrik Amazon SWF untuk CloudWatch.

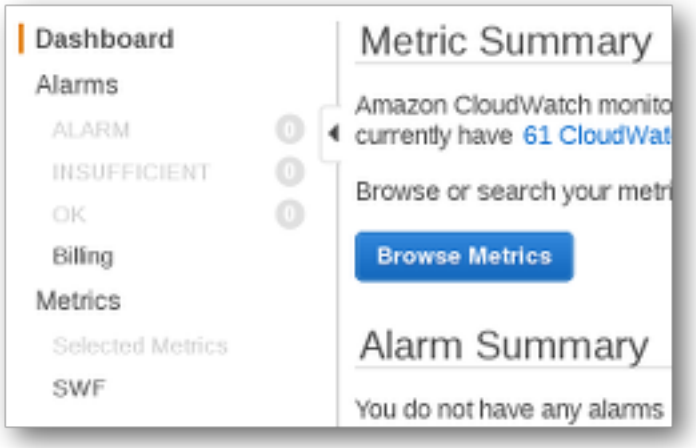
Topik

- Menampilkan Metrik
- Mengatur alarm

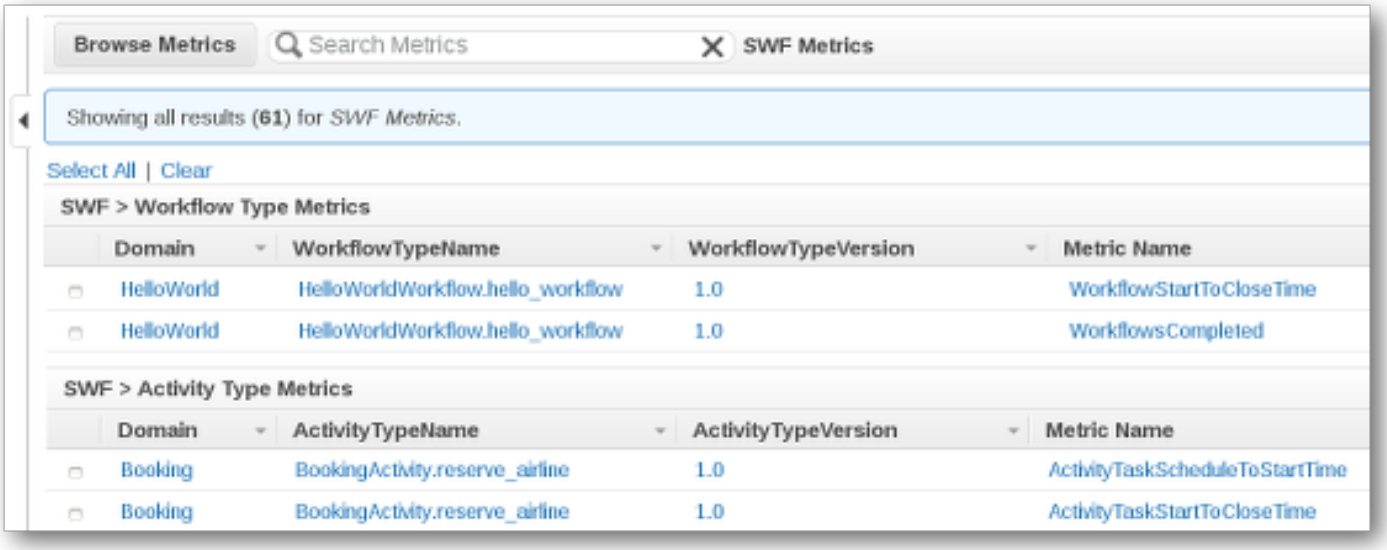
### Menampilkan Metrik

Untuk menampilkan metrik Anda untuk Amazon SWF

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, di bawah Metrics (Metrik), pilih SWF.




Jika Anda telah menjalankan eksekusi alur kerja baru-baru ini, Anda akan melihat dua daftar metrik yang disajikan: Metrik Tipe Alur Kerja dan Metrik Tipe Aktivitas.



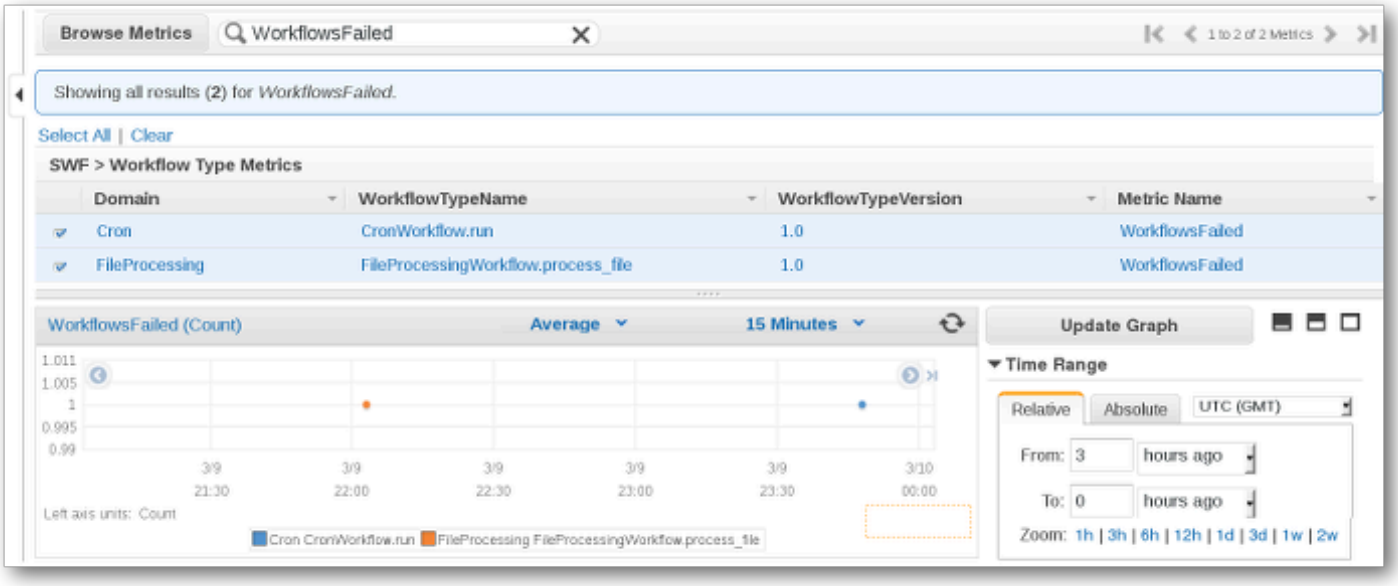

> **Note**
>
> Awalnya Anda mungkin hanya melihat Metrik Tipe Alur Kerja; Metrik Tipe Aktivitas disajikan dalam tampilan yang sama, namun Anda mungkin perlu menggulir ke bawah untuk melihatnya.

Hingga 50 metrik terbaru akan ditampilkan sekaligus, terbagi antara metrik alur kerja dan aktivitas.

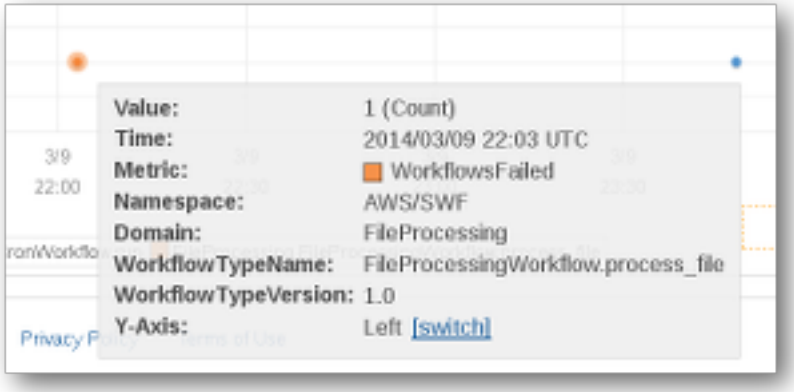
Anda dapat menggunakan judul interaktif di atas setiap kolom dalam daftar untuk mengurutkan metrik menggunakan salah satu dimensi yang disediakan. Untuk alur kerja, dimensinya adalah Domain,, WorkflowTypeNameWorkflowTypeVersion, dan Nama Metrik. Untuk aktivitas, dimensinya adalah Domain, ActivityTypeName, ActivityTypeVersion, dan Nama Metrik.

Berbagai jenis metrik dijelaskan di Metrik Amazon SWF untuk CloudWatch.

Anda dapat melihat grafik metrik dengan memilih kotak di samping baris metrik dalam daftar, dan mengubah parameter grafik menggunakan kontrol Time Range (Rentang waktu) di sebelah kanan tampilan grafik.




Untuk detail tentang titik apa pun pada grafik, letakkan kursor Anda di atas titik grafik. Detail dimensi titik akan ditampilkan.



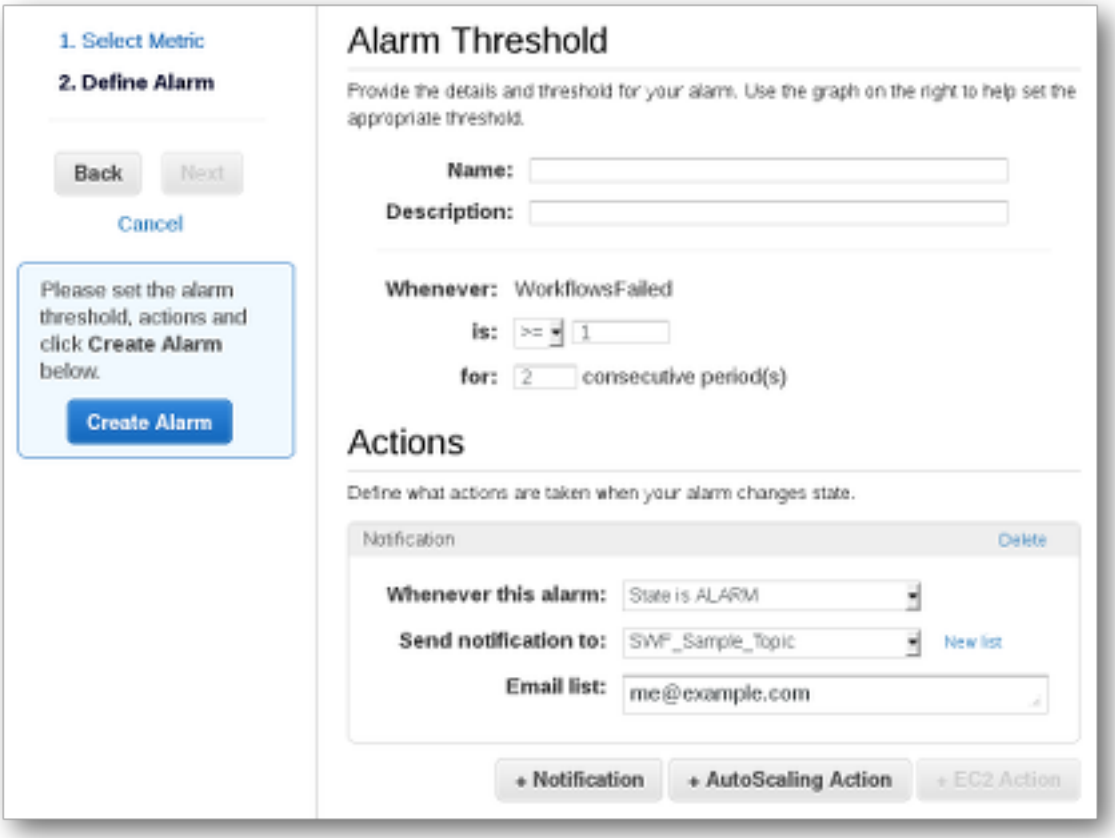
Untuk informasi selengkapnya tentang bekerja dengan CloudWatch metrik, lihat Melihat, Membuat Grafik, dan Menerbitkan Metrik di Panduan Pengguna Amazon CloudWatch .

## Mengatur alarm

Anda dapat menggunakan CloudWatch alarm untuk melakukan tindakan seperti memberi tahu Anda ketika ambang batas alarm tercapai. Misalnya, Anda dapat mengatur alarm untuk mengirim notifikasi ke topik SNS atau mengirim email ketika metrik `WorkflowsFailed` naik di atas ambang tertentu.

Mengatur alarm pada salah satu metrik

1. Pilih satu metrik dengan memilih kotaknya.
2. Di sebelah kanan grafik, di Tools (Alat), pilih Create Alarm (Buat Alarm).
3. Pada layar Define Alarm (Tentukan Alarm), masukkan nilai ambang batas alarm, parameter periode, dan tindakan yang akan diambil.




Untuk informasi selengkapnya tentang pengaturan dan penggunaan CloudWatch alarm, lihat Membuat CloudWatch Alarm Amazon di  CloudWatch Panduan Pengguna Amazon.

# Merekam panggilan API dengan AWS CloudTrail

Amazon Simple Workflow Service terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau. Layanan AWS CloudTrail menangkap semua panggilan API untuk Amazon SWF sebagai acara. Panggilan yang ditangkap mencakup panggilan dari konsol Amazon SWF dan panggilan kode ke operasi API Amazon SWF. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon SWF, 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. AWS Region 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. AWS Region 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.

## 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 seringkali merupakan aktivitas volume 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 sumber daya Amazon SWF 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 Mencatat peristiwa data dengan AWS Command Line Interface di Panduan AWS CloudTrail Pengguna.

Tabel berikut mencantumkan jenis sumber daya Amazon SWF yang dapat Anda log peristiwa data. Kolom tipe peristiwa Data 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.

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.

| Jenis peristiwa data | nilai resources.type | Data APIs masuk CloudTrail |
| --- | --- | --- |
| Domain SWF | `AWS::SWF::Domain` | Acara Alur Kerja<br>• CountClosedWorkflowExecutions<br>• CountOpenWorkflowExecutions<br>• DescribeWorkflowExecution<br>• ListClosedWorkflowExecutions<br>• ListOpenWorkflowExecutions<br>• GetWorkflowExecutionHistory<br>• RequestCancelWorkflowExecution<br>• SignalWorkflowExecution<br>• StartWorkflowExecution<br>• TerminateWorkflowExecution<br><br>Acara Tugas<br>• CountPendingActivityTasks<br>• PollForDecisionTask |

| Jenis peristiwa data | nilai resources.type | Data APIs masuk CloudTrail |
| --- | --- | --- |
| | | • PollForActivityTask<br>• RecordActivityTaskHeartbeat<br>• RespondActivityTaskCanceled<br>• RespondActivityTaskCompleted<br>• RespondActivityTaskFailed<br>• RespondDecisionTaskCompleted<br><br>Acara Keputusan<br><br>• CancelTimer<br>• CancelWorkflowExecution<br>• CompleteWorkflowExecution<br>• ContinueAsNewWorkflowExecution<br>• FailWorkflowExecution<br>• RecordMarker<br>• RequestCancelActivityTask<br>• RequestCancelExternalWorkflowExecution<br>• ScheduleActivityTask<br>• ScheduleLambdaFunction<br>• SignalExternalWorkflowExecution<br>• StartChildWorkflowExecution<br>• StartTimer |

CloudTrail peristiwa dan RespondDecisionTaskCompleted

RespondDecisionTaskCompletedTindakan mengambil daftar keputusan dalam payload permintaan. Panggilan yang telah selesai akan memancarkan peristiwa CloudTrail data N +1, satu untuk setiap keputusan ditambah satu untuk panggilan API itu sendiri. Peristiwa data dan peristiwa API semuanya akan memiliki id permintaan yang sama.

## 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 pesawat kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

Amazon Simple Workflow Service mencatat operasi bidang kontrol berikut CloudTrail sebagai peristiwa manajemen.

Acara Domain

- RegisterDomain
- DescribeDomain
- ListDomains
- DeprecateDomain
- UndeprecateDomain

Kegiatan Kegiatan

- RegisterActivityType
- DescribeActivityType
- ListActivityTypes
- DeprecateActivityType
- UndeprecateActivityType
- DeleteActivityType

WorkflowType Peristiwa

- RegisterWorkflowType
- DescribeWorkflowType
- ListWorkflowTypes
- DeprecateWorkflowType
- UndeprecateWorkflowType
- DeleteWorkflowType

Tag Acara

- TagResource
- UntagResource
- ListTagsforResource

## Contoh peristiwa

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 `CountClosedWorkflowExecutions` operasi.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "1234567890abcdef02345:admin",
      "arn": "arn:aws:sts::111122223333:assumed-role/Admin/admin",
      "accountId": "111122223333",
      "accessKeyId": "abcdef01234567890abc",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "1234567890abcdef02345",
          "arn": "arn:aws:iam::111122223333:role/Admin",
          "accountId": "111122223333",
          "userName": "Admin"
```

```
        },
        "attributes": {
          "creationDate": "2023-11-23T16:37:38Z",
          "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2023-11-23T17:52:46Z",
    "eventSource": "swf.amazonaws.com",
    "eventName": "CountClosedWorkflowExecutions",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "198.51.100.42",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.42",
    "requestParameters": {
      "domain": "nsg-domain",
      "closeTimeFilter": {
        "oldestDate": "Nov 23, 2023 5:52:46 PM",
        "latestDate": "Nov 23, 2023 5:52:46 PM"
      }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::SWF::Domain",
        "ARN": "arn:aws:swf:us-east-1:111122223333:/domain/nsg-domain"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
      "clientProvidedHostHeader": "swf.example.amazondomains.com"
    }
  }
```


Untuk informasi tentang konten CloudTrail rekaman, lihat konten CloudTrail rekaman di Panduan AWS CloudTrail Pengguna.

# EventBridge untuk perubahan status eksekusi Amazon SWF

Anda menggunakan Amazon EventBridge untuk menanggapi perubahan status atau peristiwa dalam AWS sumber daya. Ketika Amazon SWF memancarkan suatu acara, itu selalu masuk ke bus EventBridge acara default untuk akun Anda. Anda dapat membuat aturan untuk acara, mengaitkannya dengan bus acara default, dan menentukan tindakan target yang akan diambil saat EventBridge menerima acara yang cocok dengan aturan. Dengan cara ini, Anda dapat memantau alur kerja tanpa harus terus melakukan polling menggunakan API `GetWorkflowExecutionHistory`. Berdasarkan perubahan dalam eksekusi alur kerja, Anda dapat menggunakan EventBridge target untuk memanggil AWS Lambda fungsi, mempublikasikan pesan ke topik Amazon Simple Notification Service (Amazon SNS), dan banyak lagi.

Anda dapat melihat konten penuh dari kejadian perubahan status eksekusi menggunakan `DescribeWorkflowExecution`.

Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

## EventBridge acara

Tipe kejadian riwayat berisi perubahan state eksekusi. Bagian `detail` dari setiap kejadian berisi setidaknya parameter berikut:

- `eventId`: ID acara yang ditunjukkan oleh GetWorkflowExecutionHistory.
- `workflowExecutionDetail`: state alur kerja saat kejadian dipancarkan.
- `eventType`: tipe kejadian riwayat, salah satu dari berikut ini:
  - `ActivityTaskCanceled`
  - `ActivityTaskFailed`
  - `ActivityTaskTimedOut`
  - `WorkflowExecutionCanceled`
  - `WorkflowExecutionCompleted`
  - `WorkflowExecutionFailed`
  - `WorkflowExecutionStarted`
  - `WorkflowExecutionTerminated`
  - `WorkflowExecutionTimedOut`
  - `WorkflowExecutionContinuedAsNew`
  - `CancelTimerFailed`

- CancelWorkflowExecutionFailed
- ChildWorkflowExecutionFailed
- ChildWorkflowExecutionTimedOut
- CompleteWorkflowExecutionFailed
- ContinueAsNewWorkflowExecutionFailed
- DecisionTaskTimedOut
- FailWorkflowExecutionFailed
- RecordMarkerFailed
- RequestCancelActivityTaskFailed
- RequestCancelExternalWorkflowExecutionFailed
- ScheduleActivityTaskFailed
- SignalExternalWorkflowExecutionFailed
- StartActivityTaskFailed
- StartChildWorkflowExecutionFailed
- StartTimerFailed
- TimerCanceled
- LambdaFunctionFailed
- LambdaFunctionTimedOut
- StartLambdaFunctionFailed
- ScheduleLambdaFunctionFailed

## Contoh kejadian Amazon SWF

Berikut ini adalah contoh Amazon SWF mengirim acara ke: EventBridge

Topik

Dalam setiap kasus, bagian `detail` dalam data kejadian menyediakan informasi yang sama sebagai API `DescribeWorkflowExecution`. Bidang `executionStatus` menunjukkan status eksekusi pada saat kejadian dikirim, baik `OPEN` atau `CLOSED`.

Eksekusi dimulai


```
{
  "version": "0",
  "id": "444444444444",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444444444444",
  "time": "2020-05-08T15:57:38Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444444444444:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 1,
    "eventType": "WorkflowExecutionStarted",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "123456789012",
          "runId": "AKIAIOSFODNN7EXAMPLE"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588953458484,
        "closeTimestamp": null,
        "executionStatus": "OPEN",
        "closeStatus": null,
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
```

```
          "name": "444444444444"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444444444444:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 1,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": null,
    }
  }
}
```

## Eksekusi selesai

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T15:57:39Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 35,
    "eventType": "WorkflowExecutionCompleted",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
```

```
        },
        "startTimestamp": 1588953458820,
        "closeTimestamp": 1588953459448,
        "executionStatus": "CLOSED",
        "closeStatus": "COMPLETED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588953459402,
    }
  }
}
```

## Eksekusi gagal

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T15:57:38Z",
  "region": "us-east-1",
  "resources": [
```

```
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 11,
    "eventType": "WorkflowExecutionFailed",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588953158481,
        "closeTimestamp": 1588953458560,
        "executionStatus": "CLOSED",
        "closeStatus": "FAILED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": null,
    }
  }
```

```
}
```

### Batas waktu eksekusi habis

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-05T17:26:30Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 6,
    "eventType": "WorkflowExecutionTimedOut",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588698073748,
        "closeTimestamp": 1588699590745,
        "executionStatus": "CLOSED",
        "closeStatus": "TIMED_OUT",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
```

```
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 1,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588699585802,
    }
  }
}
```

Eksekusi diakhiri

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T22:37:26Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/canary"
  ],
  "detail": {
    "eventId": 48,
    "eventType": "WorkflowExecutionTerminated",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "1111-1111-1111",
          "version": "1.3"
        },
        "startTimestamp": 1588977445279,
```

```
        "closeTimestamp": 1588977446062,
        "executionStatus": "CLOSED",
        "closeStatus": "TERMINATED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "120",
        "taskList": {
          "name": "1111-1111-1111-2222-2222-2222"
        },
        "taskPriority": null,
        "childPolicy": "TERMINATE",
        "lambdaRole": null
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 1,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588977445882,
    }
  }
}
```

## Menggunakan Notifikasi Pengguna AWS dengan Amazon Simple Workflow Service

Anda dapat menggunakan Notifikasi Pengguna AWSuntuk mengatur saluran pengiriman untuk mendapatkan pemberitahuan tentang peristiwa Amazon Simple Workflow Service. Anda akan menerima notifikasi saat ada sebuah peristiwa yang cocok dengan sebuah aturan yang Anda tentukan. Anda dapat menerima pemberitahuan untuk acara melalui beberapa saluran, termasuk email, Pengembang Amazon Q dalam pemberitahuan obrolan aplikasi obrolan, atau pemberitahuan AWS Console Mobile Applicationpush. Anda juga dapat melihat notifikasi di Pusat Pemberitahuan Konsol. Notifikasi Pengguna mendukung agregasi, yang dapat mengurangi jumlah pemberitahuan yang Anda terima selama acara tertentu.

# Validasi Kepatuhan untuk Amazon Simple Workflow Service

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Simple Workflow Service sebagai bagian dari beberapa program kepatuhan AWS . Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan . 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 Laporan Pengunduhan AWS Artifak .

Tanggung jawab kepatuhan Anda saat menggunakan Amazon SWF ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan Panduan Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk Whitepaper Keamanan dan Kepatuhan HIPAA — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi yang sesuai dengan HIPAA. AWS
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub CSPM— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

# Ketahanan di Amazon Simple Workflow Service

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung

dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan 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 AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, Amazon SWF menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

## Keamanan Infrastruktur di Amazon Simple Workflow Service

Sebagai layanan terkelola, 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 melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Anda dapat menghubungi operasi API ini dari lokasi jaringan mana pun, tetapi Amazon SWF mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan Amazon SWF untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau spesifik. VPCs Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon SWF tertentu hanya dari VPC tertentu dalam jaringan. AWS

# Analisis Konfigurasi dan Kelemahan di Amazon Simple Workflow Service

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat model tanggung jawab AWS bersama.

# Menggunakan Layanan AWS CLI Alur Kerja Sederhana Amazon

Banyak fitur dari Amazon Simple Workflow Service yang dapat diakses dari AWS CLI. AWS CLI Ini menyediakan alternatif untuk menggunakan Amazon SWF dengan Konsol Manajemen AWS atau dalam beberapa kasus, untuk pemrograman dengan Amazon SWF API dan. AWS Flow Framework

Misalnya, Anda dapat menggunakan AWS CLI untuk mendaftarkan jenis alur kerja baru:

```
aws swf register-workflow-type --domain MyDomain --name "MySimpleWorkflow" --workflow-
version "v1"
```

Anda juga dapat mencantumkan tipe alur kerja terdaftar Anda:

```
aws swf list-workflow-types --domain MyDomain --registration-status REGISTERED
```

Berikut adalah contoh output default di JSON:

```
{
    "typeInfos": [
        {
            "status": "REGISTERED",
            "creationDate": 1377471607.752,
            "workflowType": {
                "version": "v1",
                "name": "MySimpleWorkflow"
            }
        },
        {
            "status": "REGISTERED",
            "creationDate": 1371454149.598,
            "description": "MyDomain subscribe workflow",
            "workflowType": {
                "version": "v3",
                "name": "subscribe"
            }
        }
    ]
}
```

Perintah Amazon SWF AWS CLI menyediakan kemampuan untuk memulai dan mengelola eksekusi alur kerja, polling untuk tugas aktivitas, merekam detak jantung tugas, dan banyak lagi! Untuk daftar lengkap perintah Amazon SWF, dengan penjelasan argumen dan contoh yang tersedia yang menunjukkan penggunaannya, lihat perintah Amazon SWF di Referensi Perintah AWS CLI .

AWS CLI Perintah mengikuti API Amazon SWF dengan cermat, sehingga Anda dapat menggunakannya AWS CLI untuk mempelajari tentang API SWF Amazon yang mendasarinya. Anda juga dapat menggunakan pengetahuan API yang ada untuk melakukan prototipe kode atau melakukan tindakan Amazon SWF pada baris perintah.

Untuk mempelajari selengkapnya AWS CLI, lihat Panduan AWS Command Line Interface Pengguna.

# Bekerja dengan Amazon SWF APIs

Selain menggunakan AWS SDKs yang dijelaskan dalamKembangkan dengan AWS SDKs, Anda dapat menggunakan HTTP API secara langsung.

Untuk menggunakan API, Anda mengirim permintaan HTTP ke Endpoint SWF sesuai dengan wilayah yang ingin Anda gunakan untuk domain, alur kerja, dan aktivitas Anda. Untuk informasi selengkapnya tentang membuat permintaan HTTP untuk Amazon SWF, lihat Membuat Permintaan HTTP ke Amazon SWF.

Bagian ini menyediakan informasi dasar tentang penggunaan HTTP API untuk mengembangkan alur kerja Anda dengan Amazon SWF. Fitur yang lebih canggih, seperti menggunakan timer, login dengan CloudTrail dan menandai alur kerja Anda disediakan di bagian,. Konsep alur kerja dasar di Amazon SWF

Topik



## Membuat Permintaan HTTP ke Amazon SWF

Jika Anda tidak menggunakan salah satunya AWS SDKs, Anda dapat melakukan operasi Amazon Simple Workflow Service (Amazon SWF) melalui HTTP menggunakan metode permintaan POST.

Metode POST mengharuskan Anda menentukan operasi di header permintaan dan memberikan data untuk operasi dalam format JSON dalam isi permintaan.

## Konten Header HTTP

Amazon SWF membutuhkan informasi berikut di header permintaan HTTP:

- `host` Endpoint Amazon SWF.
- `x-amz-date`Anda harus memberikan cap waktu baik di `Date` header HTTP atau AWS `x-amz-date header` (beberapa perpustakaan klien HTTP tidak mengizinkan Anda mengatur `Date` header). Saat header `x-amz-date` ada, sistem mengabaikan header `Date` saat mengautentikasi permintaan.

  Tanggal harus ditentukan dalam salah satu dari tiga format berikut, seperti yang ditentukan dalam HTTP/1.1 RFC:

  - Min, 06 Nov 1994 08:49:37 GMT (RFC 822, diperbarui oleh RFC 1123)
  - Min, 06-Nov-94 08:49:37 GMT (RFC 850, diusangkan oleh RFC 1036)
  - Sun 6 Nov 08:49:37 1994 (format asctime() ANSI C)
- `x-amzn-authorization` Permintaan ditandatangani dalam format:

  ```
  AWS3 AWSAccessKeyId=####,Algorithm=HmacSHA256, [,SignedHeaders=Header1;Header2;...]
  Signature=S(StringToSign)
  ```

  `AWS3`— Ini adalah tag AWS khusus implementasi yang menunjukkan versi otentikasi yang digunakan untuk menandatangani permintaan (saat ini, untuk Amazon SWF nilai ini selalu). `AWS3`

  `AWSAccessKeyId`— ID Kunci AWS Akses Anda.

  `Algorithm`— Algoritma yang digunakan untuk membuat nilai HMAC-SHA dari string-to-sign, seperti atau. `HmacSHA256` `HmacSHA1`

  `Signature`— Base64 (Algoritma ( StringToSign, SigningKey )). Untuk detail selengkapnya, lihat Menghitung Tanda Tangan HMAC-SHA untuk Amazon SWF

  `SignedHeaders`— (Opsional) Jika ada, harus berisi daftar semua Header HTTP yang digunakan dalam perhitungan HttpHeaders Canonicalized. Satu karakter titik koma (;) (karakter ASCII 59) harus digunakan sebagai pembatas untuk nilai daftar.
- `x-amz-target` – Layanan tujuan dari permintaan dan operasi untuk data, dalam format

`com.amazonaws.swf.service.model.SimpleWorkflowService. + <action>`

Sebagai contoh, `com.amazonaws.swf.service.model.SimpleWorkflowService.RegisterDomain`

- `content-type` – Jenis kebutuhan untuk menentukan JSON dan set karakter, sebagai `application/json; charset=UTF-8`

Berikut ini adalah contoh header untuk permintaan HTTP dalam pembuatan domain.

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1
Host: swf.us-east-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.25) Gecko/20111212
 Firefox/3.6.25 ( .NET CLR 3.5.30729; .NET4.0E)
Accept: application/json, text/javascript, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
X-Amz-Date: Fri, 13 Jan 2012 18:42:12 GMT
X-Amz-Target: com.amazonaws.swf.service.model.SimpleWorkflowService.RegisterDomain
Content-Encoding: amz-1.0
X-Amzn-Authorization: AWS3
 AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-
Date;X-Amz-Target;Content-Encoding,Signature=tzjkF55lxAxPhzp/BRGFYQRQRq6CqrM254dTDE/
EncI=
Referer: http://swf.us-east-1.amazonaws.com/explorer/index.html
Content-Length: 91
Pragma: no-cache
Cache-Control: no-cache

{"name": "867530902",
 "description": "music",
 "workflowExecutionRetentionPeriodInDays": "60"}
```

Berikut adalah contoh respons HTTP yang sepadan.

```
HTTP/1.1 200 OK
Content-Length: 0
```

```
Content-Type: application/json
x-amzn-RequestId: 4ec4ac3f-3e16-11e1-9b11-7182192d0b57
```

## Konten Isi HTTP

Isi permintaan HTTP berisi data untuk operasi yang ditentukan dalam header permintaan HTTP. Gunakan format data JSON untuk menyampaikan nilai-nilai data dan struktur data, secara bersamaan. Elemen dapat bersarang dalam elemen lain menggunakan notasi braket. Misalnya, hal berikut menunjukkan permintaan untuk mencantumkan semua eksekusi alur kerja yang dimulai antara dua titik yang ditentukan dalam waktu—menggunakan notasi Waktu Unix.

```
{
 "domain": "867530901",
 "startTimeFilter":
 {
   "oldestDate": 1325376070,
  "latestDate": 1356998399
 },
 "tagFilter":
 {
   "tag": "music purchase"
 }
}
```

## Sampel Permintaan dan Respon JSON Amazon SWF

Contoh berikut menunjukkan permintaan ke Amazon SWF untuk deskripsi domain yang kita buat sebelumnya. Kemudian contoh menunjukkan respon Amazon SWF.

### Permintaan POST HTTP

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1
Host: swf.us-east-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.25) Gecko/20111212
 Firefox/3.6.25 ( .NET CLR 3.5.30729; .NET4.0E)
Accept: application/json, text/javascript, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

```
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
X-Amz-Date: Sun, 15 Jan 2012 03:13:33 GMT
X-Amz-Target: com.amazonaws.swf.service.model.SimpleWorkflowService.DescribeDomain
Content-Encoding: amz-1.0
X-Amzn-Authorization: AWS3
 AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-
Date;X-Amz-Target;Content-
Encoding,Signature=IFJtq3M366CHqMlTpyqYqd9z0ChCoKDC5SCJBsLifu4=
Referer: http://swf.us-east-1.amazonaws.com/explorer/index.html
Content-Length: 21
Pragma: no-cache
Cache-Control: no-cache

{"name": "867530901"}
```

### Respons Amazon SWF

```
HTTP/1.1 200 OK
Content-Length: 137
Content-Type: application/json
x-amzn-RequestId: e86a6779-3f26-11e1-9a27-0760db01a4a8

{"configuration":
  {"workflowExecutionRetentionPeriodInDays": "60"},
 "domainInfo":
  {"description": "music",
   "name": "867530901",
   "status": "REGISTERED"}
}
```

Perhatikan protokol (`HTTP/1.1`) diikuti dengan kode status (`200`). Sebuah nilai kode `200` menunjukkan operasi yang berhasil.

Amazon SWF tidak membuat serial nilai null. Jika parser JSON Anda diatur untuk membuat serial nilai null untuk permintaan, Amazon SWF mengabaikannya.

## Menghitung Tanda Tangan HMAC-SHA untuk Amazon SWF

Setiap permintaan ke Amazon SWF harus dikonfirmasi. AWS SDKs Secara otomatis menandatangani permintaan Anda dan mengelola otentikasi berbasis token Anda. Namun, jika Anda ingin menulis permintaan `POST` HTTP Anda sendiri, Anda perlu membuat nilai `x-amzn-`

`authorization` untuk konten `POST Header` HTTP sebagai bagian dari autentikasi permintaan Anda.

Untuk informasi selengkapnya tentang format header, lihat Konten Header HTTP. Untuk AWS SDK for Java implementasi penandatanganan AWS Versi 3, lihat AWSSignerkelas.java.

## Membuat Tanda tangan Permintaan

Sebelum Anda membuat tanda tangan permintaan HMAC-SHA , Anda harus mendapatkan kredensial AWS (Access Key ID dan Kunci Rahasia).

> **Important**
>
> Anda dapat menggunakan salah satu SHA1 atau SHA256 untuk menandatangani permintaan Anda. Namun, pastikan bahwa Anda menggunakan metode yang sama selama proses penandatanganan. Metode yang Anda pilih harus sesuai dengan nilai nama `Algorithm` di header HTTP.

Untuk membuat tanda tangan permintaan

1. Buat bentuk kanosis header permintaan HTTP. Bentuk kanosis header HTTP meliputi:

   - `host`
   - Setiap elemen header dimulai dengan `x-amz-`

   Untuk informasi selengkapnya tentang header yang disertakan, lihat Konten Header HTTP.

   a. Untuk setiap pasangan nama nilai header, ubah nama header (tetapi bukan nilai header) menjadi huruf kecil.

   b. Membangun peta nama header dengan nilai header dipisahkan koma.

   ```
   x-amz-example: value1
   x-amz-example: value2  =>  x-amz-example:value1,value2
   ```

   Untuk informasi selengkapnya, lihat Bagian 4.2 dari RFC 2616.

   c. Untuk setiap pasangan nama nilai header, ubah pasangan nama nilai menjadi string dalam format `headerName:headerValue`. Trim spasi apa pun dari awal dan akhir `headerName` dan `headerValue`, tanpa spasi di setiap sisi titik dua.

```
x-amz-example1:value1,value2
x-amz-example2:value3
```

d. Masukkan baris baru (`U+000A`) setelah setiap string dikonversi, termasuk string terakhir.

e. Urutkan koleksi string yang dikonversi secara abjad, dengan nama header.

2. Buat string-to-sign nilai yang mencakup item berikut:

- Baris 1: Metode HTTP (`POST`), diikuti oleh baris baru.
- Baris 2: Permintaan URI (`/`), diikuti oleh baris baru.
- Baris 3: String kosong diikuti oleh baris baru.

> **Note**
>
> Biasanya, string kueri muncul di sini, tapi Amazon SWF tidak menggunakan string kueri.

- Baris 4–n: String yang mewakili header permintaan kanonikalisasi yang Anda dikomputasi pada Langkah 1, diikuti oleh baris baru. Baris baru ini menciptakan baris kosong antara header dan isi permintaan HTTP. Untuk informasi selengkapnya, lihat RFC 2616.
- Isi permintaan, tidak diikuti oleh baris baru.

3. Hitung SHA256 atau SHA1 intisari nilainya. string-to-sign Gunakan metode SHA yang sama di seluruh proses.

4. Compute dan Base64-encode HMAC-SHA menggunakan intisari SHA256 atau SHA1 intisari (tergantung pada metode yang Anda gunakan) dari nilai yang dihasilkan dari langkah sebelumnya dan kunci akses rahasia sementara dari Security Token Service menggunakan tindakan API. AWS `GetSessionToken`

> **Note**
>
> Amazon SWF mengharapkan tanda yang sama (=) pada akhir nilai HMAC-SHA dikodekan Base64. Jika rutinitas penkodean Base64 Anda tidak menyertakan tanda sama yang ditambahkan, tambahkan satu ke akhir nilai.

Untuk informasi selengkapnya tentang penggunaan kredenal keamanan sementara dengan Amazon SWF dan layanan AWS lainnya, AWS lihat Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

5. Tempatkan nilai yang dihasilkan sebagai nilai untuk nama `Signature` di header `x-amzn-authorization` dari permintaan HTTP ke Amazon SWF.
6. Amazon SWF memverifikasi permintaan dan melakukan operasi tertentu.

# Daftar Amazon SWF Actions by Category

Bagian ini berisi daftar topik referensi untuk tindakan Amazon SWF di Antarmuka Program Aplikasi (API) dari Amazon SWF. Daftar ini berdasarkan functional category (kategori fungsional).

Untuk daftar tindakan berdasarkan abjad, lihat Referensi API Amazon Simple Workflow Service.

Topik

- Tindakan Terkait dengan Aktivitas
- Tindakan Terkait dengan Pengambil Keputusan
- Tindakan Terkait dengan Eksekusi Alur Kerja
- Tindakan Terkait dengan Administrasi
- Tindakan Visibilitas

## Tindakan Terkait dengan Aktivitas

Aktivitas pekerja menggunakan `PollForActivityTask` untuk mendapatkan tugas aktivitas baru. Setelah menerima tugas aktivitas dari Amazon SWF, pekerja melakukan tugas dan merespons dengan `RespondActivityTaskCompleted` jika berhasil atau `RespondActivityTaskFailed` jika tidak berhasil.

Berikut ini adalah tindakan yang dilakukan oleh pekerja aktivitas.

- `PollForActivityTask`
- `RespondActivityTaskCompleted`
- `RespondActivityTaskFailed`
- `RespondActivityTaskCanceled`

- RecordActivityTaskHeartbeat

## Tindakan Terkait dengan Pengambil Keputusan

Pengambil keputusan menggunakan `PollForDecisionTask` untuk mendapatkan tugas keputusan. Setelah menerima tugas keputusan dari Amazon SWF, pengambil keputusan memeriksa riwayat eksekusi alur kerja dan memutuskan apa yang harus dilakukan selanjutnya. `RespondDecisionTaskCompleted` digunakan untuk menyelesaikan tugas keputusan dan memberikan nilai nol atau lebih untuk keputusan berikutnya.

Berikut ini adalah tindakan-tindakan yang dilakukan oleh para pengambil keputusan.

- PollForDecisionTask
- RespondDecisionTaskCompleted

## Tindakan Terkait dengan Eksekusi Alur Kerja

Tindakan berikut beroperasi pada eksekusi alur kerja.

- RequestCancelWorkflowExecution
- StartWorkflowExecution
- SignalWorkflowExecution
- TerminateWorkflowExecution

## Tindakan Terkait dengan Administrasi

Meskipun Anda dapat melakukan tugas administratif dari konsol Amazon SWF, Anda dapat menggunakan tindakan di bagian ini untuk mengotomatiskan fungsi atau membangun alat administratif Anda sendiri.

### Manajemen Aktivitas

- RegisterActivityType
- DeprecateActivityType
- UndeprecateActivityType
- DeleteActivityType

### Manajemen Alur Kerja

- RegisterWorkflowType
- DeprecateWorkflowType
- UndeprecateWorkflowType
- DeleteWorkflowType

### Manajemen Domain

Tindakan ini memungkinkan Anda untuk mendaftar dan menghentikan domain Amazon SWF.

- RegisterDomain
- DeprecateDomain
- UndeprecateDomain

Untuk informasi lebih lanjut dan contoh tindakan manajemen domain ini, lihat Mendaftarkan Domain dengan Amazon SWF.

### Manajemen Eksekusi Alur Kerja

- RequestCancelWorkflowExecution
- TerminateWorkflowExecution

## Tindakan Visibilitas

Meskipun Anda dapat melakukan tindakan visibilitas dari konsol Amazon SWF, Anda dapat menggunakan tindakan di bagian ini untuk membuat konsol atau alat administratif Anda sendiri.

### Visibilitas Aktivitas

- ListActivityTypes
- DescribeActivityType

### Visibilitas Alur Kerja

- ListWorkflowTypes

- DescribeWorkflowType

### Visibilitas Eksekusi Alur Kerja

- DescribeWorkflowExecution
- ListOpenWorkflowExecutions
- ListClosedWorkflowExecutions
- CountOpenWorkflowExecutions
- CountClosedWorkflowExecutions
- GetWorkflowExecutionHistory

### Visibilitas Domain

- ListDomains
- DescribeDomain

### Visibilitas Daftar Tugas

- CountPendingActivityTasks
- CountPendingDecisionTasks

## Mendaftarkan Domain dengan Amazon SWF

Tipe alur kerja dan aktivitas Anda serta eksekusi alur kerja itu sendiri semua tercakup dalam domain. Domain mengisolasi serangkaian tipe, eksekusi, dan daftar tugas dari lainnya dalam akun yang sama.

Anda dapat mendaftarkan domain dengan menggunakan Konsol Manajemen AWS atau dengan menggunakan `RegisterDomain` tindakan di Amazon SWF API. Contoh berikut menggunakan API.

```
https://swf.us-east-1.amazonaws.com
RegisterDomain
{
  "name" : "867530901",
  "description" : "music",
  "workflowExecutionRetentionPeriodInDays" : "60"
```

```
}
```

Parameter ditentukan dalam format JavaScript Object Notation (JSON). Di sini, periode retensi diatur ke 60 hari. Selama periode retensi, semua informasi tentang eksekusi alur kerja tersedia melalui operasi visibilitas menggunakan API SWF Amazon Konsol Manajemen AWS atau Amazon.

Setelah mendaftarkan domain, Anda harus mendaftarkan tipe alur kerja dan tipe aktivitas yang digunakan oleh alur kerja. Anda harus mendaftarkan domain terlebih dahulu karena nama domain terdaftar adalah bagian dari informasi yang diperlukan untuk mendaftarkan tipe alur kerja dan aktivitas.

### Lihat Juga

`RegisterDomain` dalam Referensi API Amazon Simple Workflow Service

## Menyetel nilai batas waktu di Amazon SWF

Topik

- Kuota pada Nilai Batas Waktu
- Batas Waktu Tugas Eksekusi Alur Kerja dan Keputusan
- Batas Waktu Aktivitas
- Lihat juga

### Kuota pada Nilai Batas Waktu

Nilai batas waktu selalu dinyatakan dalam hitungan detik, dan dapat diatur ke jumlah detik berapapun hingga satu tahun (31536000 detik)—batas eksekusi maksimum untuk alur kerja atau aktivitas apa pun. Nilai khusus `NONE` digunakan untuk menetapkan parameter batas waktu untuk “no timeout”, atau tak terbatas, tetapi batas maksimum satu tahun masih berlaku.

### Batas Waktu Tugas Eksekusi Alur Kerja dan Keputusan

Anda dapat menetapkan nilai batas waktu untuk tugas Alur Kerja dan Keputusan saat mendaftarkan tipe alur kerja. Sebagai contoh:

```
https://swf.us-east-1.amazonaws.com
RegisterWorkflowType
{
```

```
  "domain": "867530901",
  "name": "customerOrderWorkflow",
  "version": "1.0",
  "description": "Handle customer orders",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultExecutionStartToCloseTimeout": "3600",
  "defaultTaskList": { "name": "mainTaskList" },
  "defaultChildPolicy": "TERMINATE"
}
```

Pendaftaran tipe alur kerja ini menetapkan `defaultTaskStartToCloseTimeout` menjadi 600 detik (10 menit), dan `defaultExecutionStartToCloseTimeout` menjadi 3600 detik (1 jam).

Untuk informasi lebih lanjut tentang pendaftaran tipe alur kerja, lihat Mendaftarkan Tipe Alur Kerja dengan Amazon SWF, dan `RegisterWorkflowType` di Referensi API Amazon Simple Workflow Service.

Anda dapat mengambil alih nilai yang ditetapkan untuk `defaultExecutionStartToCloseTimeout` dengan menentukan `executionStartToCloseTimeout` i.

## Batas Waktu Aktivitas

Anda dapat mengatur nilai batas waktu untuk tugas aktivitas saat mendaftarkan tipe aktivitas. Sebagai contoh:

```
https://swf.us-east-1.amazonaws.com
RegisterActivityType
{
  "domain": "867530901",
  "name": "activityVerify",
  "version": "1.0",
  "description": "Verify the customer credit",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultTaskHeartbeatTimeout": "120",
  "defaultTaskList": { "name": "mainTaskList" },
  "defaultTaskScheduleToStartTimeout": "1800",
  "defaultTaskScheduleToCloseTimeout": "5400"
}
```

Pendaftaran tipe aktivitas ini menetapkan `defaultTaskStartToCloseTimeout` menjadi 600 detik (10 menit), `defaultTaskHeartbeatTimeout` menjadi 120 detik (2

menit), `defaultTaskScheduleToStartTimeout` menjadi 1800 detik (30 menit) dan `defaultTaskScheduleToCloseTimeout` menjadi 5400 detik (1,5 jam).

Untuk informasi lebih lanjut tentang pendaftaran tipe aktivitas, lihat Mendaftarkan Tipe Aktivitas dengan Amazon SWF, dan `RegisterActivityType` di Referensi API Amazon Simple Workflow Service.

Anda dapat mengambil alih nilai yang ditetapkan untuk `defaultTaskStartToCloseTimeout` dengan menentukan `taskStartToCloseTimeout` saat penjadwalan tugas aktivitas.

### Lihat juga

Tipe Batas Waktu Amazon SWF

## Mendaftarkan Tipe Alur Kerja dengan Amazon SWF

Contoh yang dibahas dalam bagian ini adalah mendaftarkan tipe alur kerja menggunakan API Amazon SWF. Nama dan versi yang Anda tentukan selama pendaftaran membentuk pengidentifikasi unik untuk tipe alur kerja. Domain yang ditentukan harus sudah terdaftar menggunakan tindakan API `RegisterDomain`.

Parameter batas waktu dalam contoh berikut adalah nilai durasi yang ditentukan dalam detik. Untuk parameter `defaultTaskStartToCloseTimeout`, Anda dapat menggunakan penentu durasi `NONE` untuk menunjukkan tidak ada batas waktu. Namun, Anda tidak dapat menentukan nilai `NONE` untuk `defaultExecutionStartToCloseTimeout`; ada batas maksimum satu tahun pada waktu yand dapat dijalankan eksekusi alur kerja. Melebihi batas ini selalu menyebabkan eksekusi alur kerja kehabisan waktu. Jika Anda menentukan nilai untuk `defaultExecutionStartToCloseTimeout` lebih besar dari satu tahun, pendaftaran akan gagal.

```
https://swf.us-east-1.amazonaws.com
RegisterWorkflowType
{
  "domain" : "867530901",
  "name" : "customerOrderWorkflow",
  "version" : "1.0",
  "description" : "Handle customer orders",
  "defaultTaskStartToCloseTimeout" : "600",
  "defaultExecutionStartToCloseTimeout" : "3600",
  "defaultTaskList" : { "name": "mainTaskList" },
  "defaultChildPolicy" : "TERMINATE"
```

```
}
```

## Lihat Juga

RegisterWorkflowType dalam Referensi API Amazon Simple Workflow Service

# Mendaftarkan Tipe Aktivitas dengan Amazon SWF

Contoh berikut mendaftarkan tipe aktivitas dengan menggunakan API Amazon SWF. Nama dan versi yang Anda tentukan selama pendaftaran membentuk pengidentifikasi unik untuk jenis aktivitas di dalam domain. Domain yang ditentukan harus sudah terdaftar menggunakan tindakan `RegisterDomain`.

Parameter batas waktu dalam contoh ini adalah nilai durasi yang ditentukan dalam detik. Anda dapat menggunakan penentu durasi `NONE` untuk menunjukkan tidak ada batas waktu.

```
https://swf.us-east-1.amazonaws.com
RegisterActivityType
{
  "domain" : "867530901",
  "name" : "activityVerify",
  "version" : "1.0",
  "description" : "Verify the customer credit",
  "defaultTaskStartToCloseTimeout" : "600",
  "defaultTaskHeartbeatTimeout" : "120",
  "defaultTaskList" : { "name" : "mainTaskList" },
  "defaultTaskScheduleToStartTimeout" : "1800",
  "defaultTaskScheduleToCloseTimeout" : "5400"
}
```

## Lihat Juga

RegisterActivityType dalam Referensi API Amazon Simple Workflow Service

# AWS Lambda tugas di Amazon SWF

Topik

- Tentang AWS Lambda
- Manfaat dan pembatasan dalam menggunakan tugas Lambda

- Menggunakan tugas Lambda dalam alur kerja Anda

## Tentang AWS Lambda

AWS Lambda adalah layanan komputasi yang dikelola sepenuhnya yang menjalankan kode Anda sebagai respons terhadap peristiwa yang dihasilkan oleh kode khusus atau dari berbagai AWS layanan seperti Amazon S3, DynamoDB, Amazon Kinesis, Amazon SNS, dan Amazon Cognito. Untuk informasi selengkapnya tentang Lambda, lihat Panduan Developer AWS Lambda.

Amazon Simple Workflow Service menyediakan tugas Lambda sehingga Anda dapat menjalankan fungsi Lambda untuk menggantikan, atau digunakan bersama kegiatan Amazon SWF tradisional.

> **Important**
>
> AWS Akun Anda akan dikenakan biaya untuk eksekusi (permintaan) Lambda yang dijalankan oleh Amazon SWF atas nama Anda. Untuk detail tentang harga Lambda, lihat https://aws.amazon.com/lambda/ harga/.

## Manfaat dan pembatasan dalam menggunakan tugas Lambda

Ada sejumlah manfaat menggunakan tugas Lambda dibandingkan menjalankan kegiatan Amazon SWF tradisional:

- Tugas Lambda tidak perlu didaftarkan atau diversikan seperti tipe aktivitas Amazon SWF.
- Anda dapat menggunakan fungsi Lambda yang ada yang telah Anda tetapkan dalam alur kerja Anda.
- Fungsi Lambda dipanggil langsung oleh Amazon SWF; Anda tidak perlu menerapkan program pekerja untuk mengeksekusi fungsi seperti yang harus Anda lakukan pada kegiatan tradisional.
- Lambda menyediakan metrik dan pencatatan untuk melacak dan menganalisis eksekusi fungsi Anda.

Ada juga sejumlah batasan mengenai tugas Lambda yang harus Anda sadari:

- Tugas Lambda hanya dapat dijalankan di AWS wilayah yang memberikan dukungan untuk Lambda. Lihat Wilayah dan Titik Akhir Lambda di Referensi Umum Amazon Web Services untuk detail tentang wilayah yang saat ini didukung untuk Lambda.

- Tugas Lambda saat ini hanya didukung oleh API HTTP SWF dasar dan di untuk Java. AWS Flow Framework Saat ini tidak ada dukungan untuk tugas Lambda di Ruby AWS Flow Framework .

## Menggunakan tugas Lambda dalam alur kerja Anda

Untuk menggunakan tugas Lambda dalam alur kerja Amazon SWF Anda, Anda akan perlu:

1. Mengatur IAM role agar memberikan izin bagi Amazon SWF untuk memanggil fungsi Lambda.
2. Melampirkan IAM role ke alur kerja Anda.
3. Memanggil fungsi Lambda Anda selama eksekusi alur kerja.

### Mengatur IAM role

Sebelum Anda dapat memanggil fungsi Lambda dari Amazon SWF Anda harus menyediakan IAM role yang menyediakan akses ke Lambda dari Amazon SWF. Anda dapat:

- pilih peran yang telah ditentukan sebelumnya, AWSLambdaPeran, untuk memberikan izin alur kerja Anda untuk menjalankan fungsi Lambda apa pun yang terkait dengan akun Anda.
- tentukan kebijakan Anda sendiri dan peran terkait untuk memberikan izin alur kerja untuk menjalankan fungsi Lambda tertentu, yang ditentukan oleh Nama Sumber Daya Amazon (). ARNs

#### Batasi izin pada peran IAM

Anda dapat membatasi izin pada peran IAM yang Anda berikan ke Amazon SWF dengan menggunakan kunci konteks `SourceAccount` dan dalam `SourceArn` kebijakan kepercayaan sumber daya Anda. Kunci ini membatasi penggunaan kebijakan IAM sehingga hanya digunakan dari eksekusi Amazon Simple Workflow Service yang termasuk dalam ARN domain yang ditentukan. Jika Anda menggunakan kedua kunci konteks kondisi global, `aws:SourceAccount` nilai dan akun yang direferensikan dalam `aws:SourceArn` nilai harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Dalam contoh berikut, kunci `SourceArn` konteks membatasi peran layanan IAM agar hanya digunakan dalam eksekusi Amazon Simple Workflow Service yang termasuk `someDomain` dalam akun,. `123456789012`

- Pernyataan 1

Prinsipal: `"Service": "swf.amazonaws.com"`

Tindakan: `sts:AssumeRole`

```
"Condition": {
   "ArnLike": {
     "aws:SourceArn": "arn:aws:swf:*:123456789012:/domain/someDomain"
   }
}
```

Dalam contoh berikut, kunci `SourceAccount` konteks membatasi peran layanan IAM agar hanya digunakan dalam eksekusi Amazon Simple Workflow Service di akun,. `123456789012`

```
"Condition": {
   "StringLike": {
     "aws:SourceAccount": "123456789012"
   }
}
```

Menyediakan akses bagi Amazon SWF untuk memanggil setiap peran Lambda

Anda dapat menggunakan peran yang telah ditentukan sebelumnya, AWSLambdaPeran, untuk memberikan alur kerja Amazon SWF Anda kemampuan untuk menjalankan fungsi Lambda apa pun yang terkait dengan akun Anda.

Untuk menggunakan AWSLambda Peran untuk memberi Amazon SWF akses untuk menjalankan fungsi Lambda

1. Buka Amazon IAM console (Konsol Amazon IAM).
2. Pilih Roles (Peran), kemudian Create New Role (Buat Peran Baru).
3. Beri nama bagi peran Anda, misalnya `swf-lambda` dan pilih Next Step (Langkah Selanjutnya).
4. Di bawah AWS Service Roles (Peran Layanan), pilih Amazon SWF, dan pilih Next Step (Langkah Berikutnya).
5. Pada layar Lampirkan Kebijakan, pilih AWSLambdaPeran dari daftar.
6. Pilih Next Step (Langkah Selanjutnya) dan kemudian Create Role (Buat Baru) setelah Anda meninjau peran tersebut.

### Menentukan IAM role untuk menyediakan akses guna memanggil fungsi Lambda tertentu

Jika Anda ingin menyediakan akses untuk memanggil fungsi Lambda tertentu dari alur kerja Anda, Anda perlu menentukan kebijakan IAM Anda sendiri.

Untuk membuat kebijakan IAM untuk menyediakan akses ke fungsi Lambda tertentu

1. Buka Amazon IAM console (Konsol Amazon IAM).
2. Pilih Policies (Kebijakan), kemudian Create Policy (Buat Kebijakan).
3. Pilih Salin Kebijakan AWS Terkelola dan pilih AWSLambdaPeran dari daftar. Sebuah kebijakan akan dibuat untuk Anda. Anda memiliki opsi untuk mengedit nama dan deskripsi kebijakan untuk menyesuaikan kebutuhan Anda.
4. Di bidang Sumber Daya di Dokumen Kebijakan, tambahkan ARN dari satu atau beberapa fungsi Lambda Anda. Misalnya:

   - Sumber daya: `arn:aws:lambda:us-east-1:111122223333:function:hello_lambda_function`

   > **Note**
   >
   > Untuk keterangan lengkap tentang cara menentukan sumber daya di IAM role, lihat Gambaran Umum Kebijakan IAM di Menggunakan IAM.

5. Pilih Create Policy (Buat Kebijakan) untuk menyelesaikan pembuatan kebijakan Anda.

Anda kemudian dapat memilih kebijakan ini saat membuat IAM role baru, dan menggunakan peran tersebut untuk memberikan akses panggilan ke alur kerja Amazon SWF Anda. Prosedur ini sangat mirip dengan membuat peran dengan kebijakan Peranan. sebagai gantinya, pilih kebijakan Anda sendiri saat membuat AWSLambdaperan.

Untuk membuat peran Amazon SWF menggunakan kebijakan Lambda Anda

1. Buka Amazon IAM console (Konsol Amazon IAM).
2. Pilih Roles (Peran), kemudian Create New Role (Buat Peran Baru).
3. Beri nama bagi peran Anda, misalnya `swf-lambda-function` dan pilih Next Step (Langkah Selanjutnya).

4. Di bawah AWS Service Roles (Peran Layanan), pilih Amazon SWF, dan pilih Next Step (Langkah Berikutnya).
5. Pada layar Attach Policy (Lampirkan Kebijakan), pilih kebijakan khusus fungsi Lambda Anda dari daftar.
6. Pilih Next Step (Langkah Selanjutnya) dan kemudian Create Role (Buat Baru) setelah Anda meninjau peran tersebut.

## Lampirkan IAM role ke alur kerja Anda

Setelah Anda menentukan IAM role Anda, Anda harus melampirkannya ke alur kerja yang akan menggunakan IAM role untuk memanggil fungsi Lambda yang dapat diakses oleh Amazon SWF.

Ada dua tempat di mana Anda dapat melampirkan peran ke alur kerja Anda:

- Selama pendaftaran tipe alur kerja. Peran ini kemudian dapat digunakan sebagai peran Lambda default untuk setiap eksekusi tipe alur kerja tersebut.
- Saat memulai eksekusi alur kerja. Peran ini hanya akan digunakan selama eksekusi alur kerja ini (dan di seluruh eksekusi).

Untuk menyediakan peran Lambda default bagi tipe alur kerja

- Saat memanggil RegisterWorkflowType, atur defaultLambdaRole bidang ke ARN dari peran yang Anda tentukan.

Untuk menyediakan peran Lambda yang akan digunakan selama eksekusi alur kerja

- Saat memanggil StartWorkflowExecution, atur bidang LambdArole ke ARN dari peran yang Anda tentukan.

> **Note**
>
> jika akun memanggil RegisterWorkflowType atau StartWorkflowExecution tidak memiliki izin untuk menggunakan peran yang diberikan, maka panggilan akan gagal dengan OperationNotPermittedFault.

## Panggil fungsi Lambda Anda dari alur kerja Amazon SWF

Anda dapat menggunakan tipe ScheduleLambdaFunctionDecisionAttributes data untuk mengidentifikasi fungsi Lambda yang akan dipanggil selama eksekusi alur kerja.

Selama panggilan ke RespondDecisionTaskCompleted, berikan daftar ScheduleLambdaFunctionDecisionAttributes keputusan Anda. Misalnya:

```
{
  "decisions": [{
    "ScheduleLambdaFunctionDecisionAttributes": {
      "id": "lambdaTaskId",
      "name": "myLambdaFunctionName",
      "input": "inputToLambdaFunction",
      "startToCloseTimeout": "30"
    },
  }],
}
```

Atur parameter berikut:

- id dengan pengidentifikasi untuk tugas Lambda. id ini harus berupa string dari karakter 1-256 dan tidak boleh berisi karakter : (titik dua), / (garis miring), | (bar vertikal), atau karakter kontrol (\u0000 - \u001f and \u007f - \u009f), maupun `arn` string literal.
- nama dengan nama fungsi Lambda Anda. Alur kerja Amazon SWF Anda harus disediakan dengan IAM role yang memberikan akses bagi alur kerja untuk memanggil fungsi Lambda. Nama yang diberikan harus mengikuti batasan untuk FunctionName parameter seperti dalam tindakan Lambda Invoke.
- input dengan input data opsional untuk fungsi. Jika disetel, ini harus mengikuti batasan untuk ClientContext parameter seperti dalam tindakan Lambda Invoke.
- startToCloseBatas waktu dengan periode maksimum opsional, dalam detik, yang dapat diambil fungsi untuk dijalankan sebelum tugas gagal dengan pengecualian batas waktu. Nilai NONE dapat digunakan untuk menentukan durasi tidak terbatas.

Untuk informasi selengkapnya, lihat Menerapkan AWS Lambda Tugas

# Mengembangkan Pekerja Aktivitas di Amazon SWF

Pekerja aktivitas menyediakan satu atau lebih implementasi tipe aktivitas. Pekerja aktivitas berkomunikasi dengan Amazon SWF untuk menerima tugas aktivitas dan melaksanakannya. Anda dapat memiliki armada beberapa pekerja aktivitas yang melakukan tugas dari tipe aktivitas yang sama.

Amazon SWF membuat tugas aktivitas yang tersedia untuk pekerja aktivitas ketika decider menjadwal tugas aktivitas. Ketika decider menjadwal tugas aktivitas, ia menyediakan data (yang Anda tentukan) yang dibutuhkan pekerja aktivitas untuk melakukan tugas aktivitas. Amazon SWF memasukkan data ini ke dalam tugas aktivitas sebelum mengirimkannya ke pekerja aktivitas.

Pekerja aktivitas dikelola oleh Anda. Mereka dapat ditulis dalam bahasa apa pun. Pekerja dapat dijalankan di mana saja, selama dapat berkomunikasi dengan Amazon SWF melalui API. Karena Amazon SWF menyediakan semua informasi yang dibutuhkan untuk melakukan tugas aktivitas, semua pekerja aktivitas dapat menjadi stateless. Keadaan stateless mengaktifkan alur kerja Anda menjadi sangat dapat diskalakan; untuk menangani peningkatan kebutuhan kapasitas, cukup tambahkan lebih banyak pekerja aktivitas.

Bagian ini menjelaskan cara mengimplementasi pekerja aktivitas. Para pekerja aktivitas harus berulang kali melakukan hal berikut.

1. Polling Amazon SWF untuk tugas aktivitas.
2. Memulai melakukan tugas.
3. Melaporkan detak jantung ke Amazon SWF secara berkala jika tugas berumur panjang.
4. Melaporkan bahwa tugas telah selesai atau gagal dan mengembalikan hasil ke Amazon SWF.

Topik

## Polling untuk Tugas Aktivitas

Untuk melakukan tugas aktivitas, setiap pekerja aktivitas harus melakulan polling Amazon SWF dengan memanggil tindakan `PollForActivityTask` secara berkala.

Pada contoh berikut, pekerja aktivitas `ChargeCreditCardWorker01` melakukan polling untuk tugas pada daftar tugas, `ChargeCreditCard-v0.1`. Jika tidak ada tugas aktivitas yang tersedia, setelah 60 detik, Amazon SWF mengirimkan kembali respons kosong. Sebuah respons kosong adalah struktur `Task` di mana nilai `taskToken` merupakan string kosong.

```
https://swf.us-east-1.amazonaws.com
PollForActivityTask
{
  "domain" : "867530901",
  "taskList" : { "name": "ChargeCreditCard-v0.1" },
  "identity" : "ChargeCreditCardWorker01"
}
```

Jika tugas aktivitas telah tersedia, Amazon SWF mengembalikannya ke pekerja aktivitas. Tugas berisi data yang ditentukan decider saat menjadwalkan aktivitas.

Setelah pekerja aktivitas menerima tugas aktivitas, ia siap untuk melakukan pekerjaan. Bagian berikutnya memberikan informasi tentang melakukan tugas aktivitas.

## Melakukan Tugas Aktivitas

Setelah menerima tugas aktivitas, pekerja aktivitas siap untuk melakukannya.

Untuk melakukan tugas aktivitas

1. Programkan pekerja aktivitas Anda untuk meinterpretasikan konten di bidang input tugas. Bidang ini berisi data yang ditentukan oleh decider ketika tugas dijadwalkan.
2. Programkan pekerja aktivitas untuk mulai memproses data dan mengeksekusi logika Anda.

Bagian selanjutnya menjelaskan cara memprogram pekerja aktivitas Anda untuk memberikan pembaruan state ke Amazon SWF untuk kegiatan yang berjalan lama.

## Pelaporan Detak Jantung Aktivitas Tugas

Jika batas waktu detak jantung terdaftar dengan jenis aktivitas, maka pekerja aktivitas harus mencatat detak jantung sebelum batas waktu detak jantung terlampaui. Jika tugas aktivitas tidak memberikan detak jantung dalam batas waktu, tugas kehabisan waktu, Amazon SWF menutupnya dan menjadwalkan tugas keputusan baru untuk menginformasikan decider dari batas waktu. Decider kemudian dapat menjadwal ulang tugas kegiatan atau mengambil tindakan lain.

Jika, setelah waktu habis, pekerja aktivitas mencoba untuk mengontak Amazon SWF, seperti dengan memanggil `RespondActivityTaskCompleted`, Amazon SWF akan mengembalikan kesalahan `UnknownResource`.

Bagian ini menjelaskan cara menyediakan detak jantung aktivitas.

Untuk mencatat detak jantung tugas aktivitas, programkan pekerja aktivitas Anda untuk memanggil tindakan `RecordActivityTaskHeartbeat`. Tindakan ini juga menyediakan bidang string yang dapat Anda gunakan untuk menyimpan data bentuk bebas untuk mengukur kemajuan dengan cara apa pun yang sesuai untuk aplikasi Anda.

Dalam contoh ini, pekerja aktivitas melaporkan detak jantung ke Amazon SWF dan menggunakan bidang detail untuk melaporkan bahwa tugas aktivitas 40 persen selesai. Untuk melaporkan detak jantung, pekerja aktivitas harus menentukan token tugas aktivitas.

```
https://swf.us-east-1.amazonaws.com
RecordActivityTaskHeartbeat
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "details" : "40"
}
```

Tindakan ini tidak dengan sendirinya membuat kejadian dalam riwayat eksekusi alur kerja; namun, jika waktu tugas habis, riwayat eksekusi alur kerja akan berisi kejadian `ActivityTaskTimedOut` yang berisi informasi dari detak jantung terakhir yang dihasilkan oleh pekerja aktivitas.

## Menyelesaikan atau Kegagalan Tugas Aktivitas

Setelah menjalankan tugas, pekerja aktivitas harus melaporkan apakah tugas kegiatan selesai atau gagal.

## Menyelesaikan Tugas Aktivitas

Untuk menyelesaikan tugas aktivitas, programkan pekerja aktivitas untuk memanggil tindakan `RespondActivityTaskCompleted` setelah berhasil menyelesaikan tugas aktivitas, yang menentukan token tugas.

Dalam contoh ini, pekerja aktivitas menunjukkan bahwa tugas berhasil diselesaikan.

```
https://swf.us-east-1.amazonaws.com
RespondActivityTaskCompleted
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "results": "40"
}
```

Ketika aktivitas selesai, Amazon SWF menjadwalkan tugas keputusan baru untuk eksekusi alur kerja yang terkait dengan aktivitas.

Programkan pekerja aktivitas untuk melakukan polling tugas kegiatan lain setelah menyelesaikan tugas yang ada. Hal ini menciptakan putaran di mana pekerja aktivitas terus melakukan polling dan menyelesaikan tugas.

Jika aktivitas tidak merespons dalam StartToCloseTimeoutperiode tersebut, atau jika ScheduleToCloseTimeouttelah terlampaui, Amazon SWF akan menghentikan tugas aktivitas dan menjadwalkan tugas keputusan. Hal ini memungkinkan decider untuk mengambil tindakan yang tepat, seperti penjadwalan ulang tugas.

Misalnya, jika EC2 instance Amazon menjalankan tugas aktivitas dan instance gagal sebelum tugas selesai, decider akan menerima peristiwa batas waktu dalam riwayat eksekusi alur kerja. Jika tugas aktivitas menggunakan detak jantung, penentu akan menerima peristiwa saat tugas gagal memberikan detak jantung berikutnya setelah instans Amazon EC2 gagal. Jika tidak, decider akhirnya menerima kejadian ketika tugas aktivitas gagal untuk menyelesaikan sebelum mencapai salah satu nilai batas waktu keseluruhannya. Kemudian terserah kepada decider untuk menetapkan kembali tugas atau mengambil beberapa tindakan lain.

## Kegagalan Tugas Aktivitas

Jika pekerja aktivitas tidak dapat melakukan tugas aktivitas untuk beberapa alasan, tetapi masih dapat berkomunikasi dengan Amazon SWF, Anda dapat memprogramnya untuk mengagalkan tugas.

Untuk memprogram pekerja aktivitas dalam pengagalan tugas aktivitas, programkan pekerja aktivitas untuk memanggil tindakan `RespondActivityTaskFailed` yang menentukan token tugas.

```
https://swf.us-east-1.amazonaws.com
RespondActivityTaskFailed
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "reason" : "CC-Invalid",
  "details" : "Credit Card Number Checksum Failed"
}
```

Sebagai developer, Anda menentukan nilai-nilai yang disimpan dalam bidang alasan dan detail. Ini adalah bentuk bebas string; Anda dapat menggunakan konvensi kode kesalahan yang melayani aplikasi Anda. Amazon SWF tidak memproses nilai-nilai ini. Namun, Amazon SWF mungkin menampilkan nilai-nilai ini di konsol tersebut.

Ketika tugas aktivitas gagal, Amazon SWF menjadwal tugas keputusan untuk eksekusi alur kerja dengan tugas aktivitas yang terkait untuk menginformasikan decider kegagalan. Programkan decider Anda untuk menangani aktivitas yang gagal, seperti dengan menjadwal ulang aktivitas atau mengagalkan eksekusi alur kerja, tergantung pada sifat kegagalan.

## Peluncuran Pekerja Aktivitas

Untuk meluncurkan pekerja aktivitas, paketkan logika Anda menjadi dapat dieksekusi yang dapat digunakan pada platform pekerja aktivitas Anda. Misalnya, Anda mungkin memaketkan kode aktivitas Anda sebagai dapat dieksekuso Java yang dapat Anda jalankan di server Linux dan Windows.

Setelah diluncurkan, pekerja Anda mulai melakukan polling untuk tugas. Hingga decider menjadwal tugas aktivitas, namun, polling ini kehabisan waktu tanpa tugas dan pekerja Anda hanya terus melakukan polling.

Karena polling merupakan permintaan keluar, pekerja aktivitas dapat berjalan pada jaringan apa pun yang memiliki akses ke endpoint Amazon SWF.

Anda dapat meluncurkan sebanyak mungkin pekerja aktivitas yang Anda inginkan. Sebagai decider jadwal tugas aktivitas, Amazon SWF mendistribusikan tugas aktivitas kepada pekerja aktivitas polling secara otomatis.

# Mengembangkan penentu di Amazon SWF

Decider adalah implementasi dari logika koordinasi tipe alur kerja Anda yang berjalan selama eksekusi alur kerja Anda. Anda dapat menjalankan beberapa decider untuk satu tpe alur kerja.

Karena state eksekusi untuk eksekusi alur kerja disimpan dalam riwayat alur kerja, decider dapat menjadi stateless. Amazon SWF mempertahankan riwayat eksekusi alur kerja dan memberikannya ke decider dengan setiap tugas keputusan. Hal ini memungkinkan Anda untuk secara dinamis menambahkan dan menghapus decider jika diperlukan, sehingga pemrosesan alur kerja Anda sangat terukur. Seiring bertambahnya beban pada sistem, Anda cukup menambahkan lebih banyak decider untuk menangani peningkatan kapasitas. Perhatikan, bagaimanapun, hanya ada satu tugas keputusan yang terbuka setiap saat untuk eksekusi alur kerja tertentu.

Setiap kali perubahan state terjadi untuk eksekusi alur kerja, Amazon SWF menjadwal tugas keputusan. Setiap kali decider menerima tugas keputusan, decider melakukan hal berikut:

- Menginterpretasi riwayat eksekusi alur kerja yang disediakan dengan tugas keputusan
- Menerapkan logika koordinasi berdasarkan riwayat eksekusi alur kerja dan membuat keputusan tentang apa yang harus dilakukan selanjutnya. Setiap keputusan diwakili oleh struktur Keputusan
- Menyelesaikan tugas keputusan dan memberikan daftar keputusan untuk Amazon SWF.

Bagian ini menjelaskan cara mengembangkan decider, yang melibatkan:

- Pemrograman decider Anda untuk polling tugas keputusan
- Pemrograman decider Anda untuk menginterpretasi riwayat eksekusi alur kerja dan membuat keputusan
- Pemrograman decider Anda untuk meresponsi tugas keputusan.

Contoh dalam bagian ini menunjukkan cara memprogram decider untuk alur kerja contoh perdagangan elektronik.

Anda dapat mengimplementasikan decider dalam bahasa apapun yang Anda suka dan menjalankannya di mana saja, selama decider dapat berkomunikasi dengan Amazon SWF melalui API layanannya.

Topik

## Menentukan Logika Koordinasi

Hal pertama yang harus dilakukan ketika mengembangkan decider adalah menentukan logika koordinasi. Dalam contoh perdagangan elektronik, logika koordinasi yang menjadwalkan setiap aktivitas setelah aktivitas sebelumnya selesai mungkin terlihat mirip dengan berikut ini:

```
IF lastEvent = "StartWorkflowInstance"
 addToDecisions ScheduleVerifyOrderActivity

ELSIF lastEvent = "CompleteVerifyOrderActivity"
 addToDecisions ScheduleChargeCreditCardActivity

ELSIF lastEvent = "CompleteChargeCreditCardActivity"
 addToDecisions ScheduleCompleteShipOrderActivity

ELSIF lastEvent = "CompleteShipOrderActivity"
 addToDecisions ScheduleRecordOrderCompletion

ELSIF lastEvent = "CompleteRecordOrderCompletion"
 addToDecisions CloseWorkflow

ENDIF
```

Decider menerapkan logika koordinasi untuk riwayat eksekusi alur kerja dan membuat daftar keputusan ketika menyelesaikan tugas keputusan menggunakan tindakan `RespondDecisionTaskCompleted`.

## Polling untuk Tugas Keputusan

Setiap decider melakukan polling untuk tugas keputusan. Tugas keputusan berisi informasi yang digunakan decider untuk menghasilkan keputusan seperti tugas kegiatan penjadwalan. Untuk polling tugas keputusan, decider menggunakan tindakan `PollForDecisionTask`.

Dalam contoh ini, decider melakukan polling tugas keputusan, menentukan daftar tugas `customerOrderWorkflow-0.1`.

```
https://swf.us-east-1.amazonaws.com
PollForDecisionTask
{
  "domain": "867530901",
  "taskList": {"name": "customerOrderWorkflow-v0.1"},
  "identity": "Decider01",
  "maximumPageSize": 50,
  "reverseOrder": true
}
```

Jika tugas keputusan tersedia dari daftar tugas yang ditentukan, Amazon SWF segera mengembalikannya. Jika tidak ada tugas keputusan tersedia, Amazon SWF memegang koneksi terbuka hingga 60 detik, dan mengembalikan tugas segera setelah tersedia. Jika tidak ada tugas yang tersedia, Amazon SWF mengembalikan respons kosong. Respons kosong adalah struktur `Task` di mana nilai `taskToken` merupakan string kosong. Pastikan untuk memprogram decider Anda untuk polling tugas lain jika menerima respons kosong.

Jika tugas keputusan tersedia, Amazon SWF mengembalikan respons yang berisi tugas keputusan serta tampilan riwayat eksekusi alur kerja yang diberi nomor halaman.

Dalam contoh ini, tipe kejadian terbaru menunjukkan eksekusi alur kerja telah dimulai dan elemen input berisi informasi yang diperlukan untuk melakukan tugas pertama.

```
{
  "events": [
    {
      "decisionTaskStartedEventAttributes": {
        "identity": "Decider01",
        "scheduledEventId": 2
      },
      "eventId": 3,
      "eventTimestamp": 1326593394.566,
      "eventType": "DecisionTaskStarted"
    }, {
      "decisionTaskScheduledEventAttributes": {
        "startToCloseTimeout": "600",
        "taskList": { "name": "specialTaskList" }
      },
      "eventId": 2,
```

```
      "eventTimestamp": 1326592619.474,
      "eventType": "DecisionTaskScheduled"
    }, {
      "eventId": 1,
      "eventTimestamp": 1326592619.474,
      "eventType": "WorkflowExecutionStarted",
      "workflowExecutionStartedEventAttributes": {
        "childPolicy" : "TERMINATE",
        "executionStartToCloseTimeout" : "3600",
        "input" : "data-used-decider-for-first-task",
        "parentInitiatedEventId": 0,
        "tagList" : ["music purchase", "digital", "ricoh-the-dog"],
        "taskList": { "name": "specialTaskList" },
        "taskStartToCloseTimeout": "600",
        "workflowType": {
          "name": "customerOrderWorkflow",
          "version": "1.0"
        }
      }
    }
  ],
  ...
}
```

Setelah menerima riwayat eksekusi alur kerja, decider menginterpretasi riwayat dan membuat keputusan berdasarkan logika koordinasi.

Karena jumlah kejadian riwayat untuk eksekusi satu alur kerja mungkin besar, hasil yang dikembalikan mungkin dipisah di sejumlah halaman. Untuk mengambil halaman berikutnya, lakukan panggilan tambahan untuk `PollForDecisionTask` menggunakan yang nextPageTokendikembalikan oleh panggilan awal. Perhatikan bahwa Anda tidak menelepon `GetWorkflowExecutionHistory` dengan ini nextPageToken. Sebagai gantinya, panggil `PollForDecisionTask` lagi.

## Menerapkan Logika Koordinasi

Setelah decider menerima tugas keputusan, programkan untuk menginterpretasi riwayat eksekusi alur kerja untuk menentukan apa yang telah terjadi sejauh ini. Berdasarkan hal tersebut, decider harus menghasilkan daftar keputusan.

Dalam contoh perdagangan elektronik, kita hanya peduli dengan kejadian terakhir dalam riwayat alur kerja, jadi kita mendefinisikan logika berikut.

```
IF lastEvent = "StartWorkflowInstance"
 addToDecisions ScheduleVerifyOrderActivity

ELSIF lastEvent = "CompleteVerifyOrderActivity"
 addToDecisions ScheduleChargeCreditCardActivity

ELSIF lastEvent = "CompleteChargeCreditCardActivity"
 addToDecisions ScheduleCompleteShipOrderActivity

ELSIF lastEvent = "CompleteShipOrderActivity"
 addToDecisions ScheduleRecordOrderCompletion

ELSIF lastEvent = "CompleteRecordOrderCompletion"
 addToDecisions CloseWorkflow

ENDIF
```

Jika lastEvent adalah `CompleteVerifyOrderActivity`, Anda akan menambahkan kegiatan `ScheduleChargeCreditCardActivity` ke daftar keputusan.

Setelah decider menentukan keputusan yang akan diambil, decider dapat merespons Amazon SWF dengan keputusan yang tepat.

## Merespons dengan Keputusan

Setelah menginterpreasikan riwayat alur kerja dan menghasilkan daftar keputusan, decider siap untuk merespons kembali ke Amazon SWF dengan keputusan tersebut.

Programkan decider Anda untuk mengekstraksi data yang dibutuhkan dari riwayat eksekusi alur kerja, lalu buat keputusan yang menentukan tindakan berikutnya sesuai untuk alur kerja. Decider mentransmisikan keputusan ini kembali ke Amazon SWF menggunakan tindakan `RespondDecisionTaskCompleted`. Lihat Referensi API mazon Simple Workflow Service untuk daftar tipe keputusan yang tersedia.

Dalam contoh perdagangan elektronik, ketika decider merespons dengan serangkaian keputusan yang dihasilkan, itu juga termasuk input kartu kredit dari riwayat eksekusi alur kerja. Pekerja aktivitas kemudian memiliki informasi yang dibutuhkan untuk melakukan tugas aktivitas.

Ketika semua aktivitas dalam eksekusi alur kerja selesai, decider menutup eksekusi alur kerja.

```
https://swf.us-east-1.amazonaws.com
```

```
RespondDecisionTaskCompleted
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions" : [
    {
      "decisionType" :"ScheduleActivityTask",
      "scheduleActivityTaskDecisionAttributes" : {
        "control" :"OPTIONAL_DATA_FOR_DECIDER",
        "activityType" : {
          "name" :"ScheduleChargeCreditCardActivity",
          "version" :"1.1"
        },
        "activityId" :"3e2e6e55-e7c4-beef-feed-aa815722b7be",
        "scheduleToCloseTimeout" :"360",
        "taskList" : { "name" :"CC_TASKS" },
        "scheduleToStartTimeout" :"60",
        "startToCloseTimeout" :"300",
        "heartbeatTimeout" :"60",
        "input" : "4321-0001-0002-1234: 0212 : 234"
      }
    }
  ]
}
```

## Menutup Eksekusi Alur Kerja

Ketika decider menentukan bahwa proses bisnis selesai, yaitu, tidak ada lagi kegiatan untuk dilakukan, decider menghasilkan keputusan untuk menutup eksekusi alur kerja.

Untuk menutup eksekusi alur kerja, programkan decider Anda untuk menginterpretasikan kejadian dalam riwayat alur kerja untuk menentukan apa yang telah terjadi dalam eksekusi sejauh ini dan melihat apakah eksekusi alur kerja harus ditutup.

Jika alur kerja telah berhasil diselesaikan, kemudian tutup eksekusi alur kerja dengan memanggil `RespondDecisionTaskCompleted` dengan keputusan `CompleteWorkflowExecution`. Atau, Anda bisa gagal dalam eksekusi yang salah menggunakan keputusan `FailWorkflowExecution`.

Dalam contoh perdagangan elektronik, decider meninjau riwayat dan menambahkan keputusan berdasarkan logika koordinasi untuk menutup eksekusi alur kerja ke daftar keputusannya, dan memulai tindakan `RespondDecisionTaskCompleted` dengan keputusan alur kerja yang tertutup.

> **Note**
>
> Ada beberapa kasus di mana menutup alur kerja eksekusi mengalami kegagalan. Misalnya, jika sinyal diterima saat decider menutup eksekusi alur kerja, keputusan tertutup akan gagal. Untuk menangani kemungkinan ini, pastikan bahwa decider terus melakukan polling untuk tugas-tugas keputusan. Dalam kasus ini, pastikan juga bahwa decider yang menerima tugas keputusan berikutnya merespons kejadian, sinyal yang mencegah eksekusi dari penutupan.

Anda mungkin juga mendukung pembatalan eksekusi alur kerja. Hal ini sangat berguna untuk alur kerja yang berjalan lama. Untuk mendukung pembatalan, decider Anda harus menangani kejadian `WorkflowExecutionCancelRequested` dalam riwayat. Kejadian ini menunjukkan bahwa pembatalan eksekusi telah diminta. Decider Anda harus melakukan tindakan pembersihan yang tepat, seperti membatalkan tugas aktivitas yang sedang berlangsung, dan menutup alur kerja yang memanggil tindakan `RespondDecisionTaskCompleted` dengan keputusan `CancelWorkflowExecution`.

Contoh berikut memanggil `RespondDecisionTaskCompleted` untuk menentukan bahwa eksekusi alur kerja saat ini dibatalkan.

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions" : [
    {
      "decisionType":"CancelWorkflowExecution",
      "CancelWorkflowExecutionAttributes":{
        "Details": "Customer canceled order"
      }
    }
  ]
}
```

Amazon SWF memeriksa untuk memastikan bahwa keputusan penutupan atau pembatalan eksekusi alur kerja adalah keputusan terakhir yang dikirim oleh decider. Artinya, tidak valid untuk memiliki serangkaian keputusan di mana ada keputusan setelah salah satu menutup alur kerja.

## Meluncurkan Decider

Setelah menyelesaikan pengembangan decider, Anda siap untuk meluncurkan satu atau lebih decider.

Untuk meluncurkan decider, paketkan logika koordinasi Anda menjadi dapat dieksekusi yang dapat digunakan pada platform decider Anda. Misalnya, Anda mungkin memaketkan kode decider Anda sebagai dapat dieksekusi Java yang dapat Anda jalankan pada komputer Linux dan Windows.

Setelah diluncurkan, decider Anda harus mulai nelakukan polling Amazon SWF untuk tugas-tugas. Sampai Anda memulai eksekusi alur kerja dan Amazon SWF menjadwal tugas keputusan, polling ini akan kehabisan waktu dan mendapat respons kosong. Respons kosong adalah struktur `Task` di mana nilai `taskToken` merupakan string kosong. Keputusan Anda harus terus melakukan polling.

Amazon SWF memastikan bahwa hanya satu tugas keputusan yang dapat aktif untuk eksekusi alur kerja setiap saat. Hal ini mencegah masalah seperti keputusan yang bertentangan. Selain itu, Amazon SWF memastikan bahwa satu tugas keputusan ditugaskan untuk decider tunggal, terlepas dari jumlah decider yang berjalan.

Jika terjadi sesuatu yang menghasilkan tugas keputusan saat decider sedang memproses tugas keputusan lain, Amazon SWF mengantrekan tugas baru sampai tugas saat selesai. Setelah tugas saat ini selesai, Amazon SWF membuat tugas keputusan baru yang tersedia. Juga, tugas keputusan yang di-batch dalam artian, jika beberapa aktivitas telah selesai sementara decider sedang memproses tugas keputusan, Amazon SWF hanya akan membuat satu tugas keputusan baru untuk memperhitungkan beberapa penyelesaian tugas. Namun, setiap penyelesaian tugas akan menerima kejadian individu dalam riwayat eksekusi alur kerja.

Karena polling merupakan permintaan keluar, decider dapat berjalan di jaringan apa pun yang memiliki akses ke endpoint Amazon SWF.

Agar eksekusi alur kerja untuk berlanjut, satu atau lebih decider harus berjalan. Anda dapat meluncurkan sebanyak decider yang Anda inginkan. Amazon SWF mendukung beberapa decider yang melakukan polling pada daftar tugas yang sama.

# Memulai alur kerja di Amazon SWF

Anda dapat memulai eksekusi alur kerja dari tipe alur kerja terdaftar dari aplikasi apa pun menggunakan tindakan `StartWorkflowExecution`. Ketika memulai eksekusi, Anda mengaitkan pengidentifikasi, bernama `workflowId`, dengan tindakan tersebut. Parameter `workflowId` dapat

berupa string yang sesuai untuk aplikasi Anda, seperti nomor pesanan dalam aplikasi pemrosesan pesanan. Anda tidak dapat menggunakan `workflowId` yang sama untuk beberapa eksekusi alur kerja terbuka dalam domain yang sama. Misalnya, jika Anda memulai dua eksekusi alur kerja dengan `workflowId` `Customer Order 01`, eksekusi alur kerja kedua tidak akan dimulai dan permintaan akan gagal. Namun, Anda dapat menggunakan kembali `workflowId` dari eksekusi tertutup. Amazon SWF juga mengaitkan sistem unik yang dihasilkan pengidentifikasi, bernama `runId`, dengan setiap eksekusi alur kerja.

Setelah tipe alur kerja dan aktivitas terdaftar, mulai alur kerja dengan memanggil tindakan `StartWorkflowExecution`. Nilai dari parameter `input` dapat berupa string apa pun yang ditentukan oleh aplikasi yang memulai alur kerja. Parameter `executionStartToCloseTimeout` adalah lama waktu dalam detik yang dapat dihabiskan eksekusi alur kerja dari memulai hingga menutup. Melebihi batas ini menyebabkan eksekusi alur kerja kehabisan waktu. Tidak seperti beberapa parameter batas waktu lainnya di Amazon SWF, Anda tidak dapat menentukan nilai `NONE` untuk batas waktu ini; ada batas maksimum satu tahun pada waktu yang dapat dijalankan eksekusi alur kerja. Demikian pula, lamanya waktu dalam hitungan detik yang dapat diambil oleh tugas keputusan yang terkait dengan eksekusi alur kerja ini sebelum waktu habis. taskStartToCloseTimeout

```
https://swf.us-east-1.amazonaws.com
StartWorkflowExecution
{
  "domain" : "867530901",
  "workflowId" : "20110927-T-1",
  "workflowType" : {
    "name" : "customerOrderWorkflow", "version" : "1.1"
  },
  "taskList" : { "name" : "specialTaskList" },
  "input" : "arbitrary-string-that-is-meaningful-to-the-workflow",
  "executionStartToCloseTimeout" : "1800",
  "tagList" : [ "music purchase", "digital", "ricoh-the-dog" ],
  "taskStartToCloseTimeout" : "1800",
  "childPolicy" : "TERMINATE"
}
```

Jika tindakan `StartWorkflowExecution` berhasil, Amazon SWF mengembalikan `runId` untuk eksekusi alur kerja. Parameter `runId` untuk eksekusi alur kerja adalah unik dalam wilayah tertentu. Simpan `runId` jika nanti Anda perlu menentukan eksekusi alur kerja ini dalam panggilan ke Amazon SWF. Misalnya, Anda akan menggunakan `runId` jika nanti Anda perlu mengirim sinyal ke eksekusi alur kerja.

```
{"runId": "9ba33198-4b18-4792-9c15-7181fb3a8852"}
```

# Menetapkan prioritas tugas di Amazon SWF

Secara default, tugas pada daftar tugas dikirimkan berdasarkan waktu kedatangan: tugas yang dijadwalkan pertama biasanya dijalankan pertama, sejauh mungkin. Dengan mengatur prioritas tugas opsional, Anda dapat memberikan prioritas bagi tugas tertentu: Amazon SWF akan mencoba untuk mengirimkan tugas dengan prioritas yang lebih tinggi pada daftar tugas terlebih dahulu sebelum tugas yang dengan prioritas lebih rendah.

> **Note**
>
> Tugas yang dijadwalkan pertama biasanya dijalankan terlebih dahulu, tetapi hal ini tidak dapat dipastikan.

Anda dapat mengatur prioritas tugas untuk alur kerja dan aktivitas. Prioritas tugas alur kerja tidak memengaruhi prioritas tugas aktivitas apa pun yang dijadwalkan, juga tidak memengaruhi alur kerja anak yang dimulai oleh alur kerja tersebut. Prioritas default untuk aktivitas atau alur kerja diatur (baik oleh Anda atau oleh Amazon SWF) selama pendaftaran, dan prioritas tugas terdaftar selalu digunakan kecuali diabil alih saat menjadwalkan aktivitas atau memulai eksekusi alur kerja.

Nilai prioritas tugas dapat berkisar antara “-2147483648” hingga “2147483647”, dengan angka yang lebih tinggi menunjukkan prioritas yang lebih tinggi. Jika Anda tidak mengatur prioritas tugas untuk aktivitas atau alur kerja, prioritas akan diberikan prioritas nol ("0").

Topik



## Mengatur Prioritas Tugas untuk Alur Kerja

Anda dapat mengatur prioritas tugas untuk alur kerja saat Anda mendaftarkannya atau memulainya. Prioritas tugas yang diatur saat tipe alur kerja terdaftar digunakan sebagai default untuk setiap eksekusi tipe alur kerja tersebut, kecuali jika diambil alih saat memulai eksekusi alur kerja.

Untuk mendaftarkan jenis alur kerja dengan prioritas tugas default, setel defaultTaskPriorityopsi saat menggunakan RegisterWorkflowTypetindakan:

```
{
  "domain": "867530901",
  "name": "expeditedOrderWorkflow",
  "version": "1.0",
  "description": "Expedited customer orders workflow",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultExecutionStartToCloseTimeout": "3600",
  "defaultTaskList": {"name": "mainTaskList"},
  "defaultTaskPriority": "10",
  "defaultChildPolicy": "TERMINATE"
}
```

Anda dapat mengganti prioritas tugas terdaftar tipe alur kerja saat memulai eksekusi alur kerja dengan: StartWorkflowExecution

```
{
  "childPolicy": "TERMINATE",
  "domain": "867530901",
  "executionStartToCloseTimeout": "1800",
  "input": "arbitrary-string-that-is-meaningful-to-the-workflow",
  "tagList": ["music purchase", "digital", "ricoh-the-dog"],
  "taskList": {"name": "specialTaskList"},
  "taskPriority": "-20",
  "taskStartToCloseTimeout": "600",
  "workflowId": "20110927-T-1",
  "workflowType": {"name": "customerOrderWorkflow", "version": "1.0"},
}
```

Anda juga dapat mengganti prioritas tugas terdaftar saat memulai alur kerja anak atau saat melanjutkan alur kerja sebagai baru, seperti saat menanggapi keputusan dengan. RespondDecisionTaskCompleted

Untuk menetapkan prioritas tugas alur kerja anak, sediakan nilai di `startChildWorkflowExecutionDecisionAttributes`:

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
```

```
    {
      "decisionType": "StartChildWorkflowExecution",
      "startChildWorkflowExecutionDecisionAttributes": {
        "childPolicy": "TERMINATE",
        "control": "digital music",
        "executionStartToCloseTimeout": "900",
        "input": "201412-Smith-011x",
        "taskList": {"name": "specialTaskList"},
        "taskPriority": "5",
        "taskStartToCloseTimeout": "600",
        "workflowId": "verification-workflow",
        "workflowType": {
          "name": "MyChildWorkflow",
          "version": "1.0"
        }
      }
    }
  ]
}
```

Saat melanjutkan alur kerja sebagai alur kerja baru, atur prioritas tugas di `continueAsNewWorkflowExecutionDecisionAttributes`:

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
    {
      "decisionType": "ContinueAsNewWorkflowExecution",
      "continueAsNewWorkflowExecutionDecisionAttributes": {
        "childPolicy": "TERMINATE",
        "executionStartToCloseTimeout": "1800",
        "input": "5634-0056-4367-0923,12/12,437",
        "taskList": {"name": "specialTaskList"},
        "taskStartToCloseTimeout": "600",
        "taskPriority": "100",
        "workflowTypeVersion": "1.0"
      }
    }
  ]
}
```

# Mengatur Prioritas Tugas untuk Aktivitas

Anda dapat mengatur prioritas tugas untuk suatu aktivitas baik ketika mendaftarkan maupun ketika menjadwalkan aktivitas tersebut. Prioritas tugas yang diatur ketika mendaftarkan tipe aktivitas digunakan sebagai prioritas default ketika aktivitas dijalankan, kecuali diambil alih ketika menjadwalkan aktivitas.

Untuk menetapkan prioritas tugas saat mendaftarkan jenis aktivitas, setel defaultTaskPriorityopsi saat menggunakan RegisterActivityTypetindakan:

```
{
  "defaultTaskHeartbeatTimeout": "120",
  "defaultTaskList": {"name": "mainTaskList"},
  "defaultTaskPriority": "10",
  "defaultTaskScheduleToCloseTimeout": "900",
  "defaultTaskScheduleToStartTimeout": "300",
  "defaultTaskStartToCloseTimeout": "600",
  "description": "Verify the customer credit card",
  "domain": "867530901",
  "name": "activityVerify",
  "version": "1.0"
}
```

Untuk menjadwalkan tugas dengan prioritas tugas, gunakan opsi TaskPriority saat menjadwalkan aktivitas dengan tindakan: RespondDecisionTaskCompleted

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
    {
      "decisionType": "ScheduleActivityTask",
      "scheduleActivityTaskDecisionAttributes": {
        "activityId": "verify-account",
        "activityType": {
            "name": "activityVerify",
            "version": "1.0"
        },
        "control": "digital music",
        "input": "abab-101",
        "taskList": {"name": "mainTaskList"},
        "taskPriority": "15"
      }
```

```
    }
  ]
}
```

## Tindakan yang Mengembalikan Informasi Tugas Prioritas

Anda bisa mendapatkan informasi tentang mengatur prioritas tugas (atau mengatur prioritas tugas default) dari tindakan Amazon SWF berikut:

- DescribeActivityTypemengembalikan defaultTaskPriorityjenis aktivitas di `configuration` bagian respon.
- DescribeWorkflowExecutionmengembalikan TaskPriority dari eksekusi alur kerja di bagian respon`executionConfiguration`.
- DescribeWorkflowTypemengembalikan defaultTaskPriorityjenis alur kerja di `configuration` bagian respon.
- GetWorkflowExecutionHistorydan PollForDecisionTaskmemberikan informasi prioritas tugas di `activityTaskScheduledEventAttributesdecisionTaskScheduledEventAttributes`,`workfl` dan `workflowExecutionStartedEventAttributes` bagian respons.

# Menangani kesalahan di Amazon SWF

Ada sejumlah jenis kesalahan yang dapat terjadi selama eksekusi alur kerja.

Topik

- Validasi Kesalahan
- Kesalahan dalam Memenuhi Tindakan atau Keputusan
- Timeout
- Kesalahan dimunculkan oleh kode pengguna
- Kesalahan yang terkait dengan penutupan eksekusi alur kerja

## Validasi Kesalahan

Validasi kesalahan terjadi ketika permintaan ke Amazon SWF gagal karena tidak dibentuk dengan benar atau berisi data yang tidak valid. Dalam konteks ini, permintaan bisa menjadi tindakan seperti `DescribeDomain` atau bisa menjadi keputusan seperti `StartTimer`. Jika permintaan merupakan

sebuah tindakan, Amazon SWF mengembalikan kode kesalahan dalam respons. Periksa kode kesalahan ini karena dapat memberikan informasi tentang aspek permintaan apa yang menyebabkan kegagalan. Misalnya, satu atau lebih dari argumen yang diteruskan dengan permintaan mungkin tidak valid. Untuk daftar kode kesalahan umum, buka topik untuk tindakan di Referensi API Amazon Simple Workflow Service.

Jika permintaan yang gagal adalah keputusan, sebuah kejadian yang sesuai akan tercantum dalam riwayat eksekusi alur kerja. Misalnya, jika keputusan `StartTimer` gagal, Anda akan melihat acara `StartTimerFailed` dalam riwayat. Decider harus memeriksa kejadian ini ketika menerima riwayat dalam merespons `PollForDecisionTask` atau `GetWorkflowExecutionHistory`. Di bawah ini adalah daftar kemungkinan kejadian kegagalan keputusan yang dapat terjadi ketika keputusan tidak dibentuk dengan benar atau berisi data yang tidak valid.

## Kesalahan dalam Memenuhi Tindakan atau Keputusan

Bahkan jika permintaan terbentuk dengan benar, kesalahan dapat terjadi ketika Amazon SWF mencoba untuk melaksanakan permintaan. Dalam kasus ini, salah satu kejadian berikut dalam riwayat akan menunjukkan bahwa terjadi kesalahan. Lihatlah bidang acara `reason` untuk menentukan penyebab kegagalan.

- `CancelTimerFailed`
- `RequestCancelActivityTaskFailed`
- `RequestCancelExternalWorkflowExecutionFailed`
- `ScheduleActivityTaskFailed`
- `SignalExternalWorkflowExecutionFailed`
- `StartChildWorkflowExecutionFailed`
- `StartTimerFailed`

## Timeout

Decider, pekerja aktivitas, dan eksekusi alur kerja semua beroperasi dalam batasan periode batas waktu. Dalam jenis kesalahan ini, waktu tugas atau alur kerja anak habis. Sebuah kejadian akan muncul dalam riwayat yang menggambarkan batas waktu. Decider harus menangani kejadian ini dengan, misalnya, menjadwal ulang tugas atau memulai ulang alur kerja anak. Untuk informasi selengkapnya tentang batas waktu, lihat Tipe Batas Waktu Amazon SWF

- `ActivityTaskTimedOut`

- `ChildWorkflowExecutionTimedOut`
- `DecisionTaskTimedOut`
- `WorkflowExecutionTimedOut`

## Kesalahan dimunculkan oleh kode pengguna

Contoh dari jenis syarat kesalahan adalah kegagalan tugas aktivitas dan kegagalan alur kerja anak. Seperti kesalahan batas waktu, Amazon SWF menambahkan kejadian yang sesuai dengan riwayat eksekusi alur kerja. Decider harus menangani kejadian ini, mungkin dengan menjadwal ulang tugas atau memulai ulang alur kerja anak.

- `ActivityTaskFailed`
- `ChildWorkflowExecutionFailed`

## Kesalahan yang terkait dengan penutupan eksekusi alur kerja

Decider juga dapat melihat kejadian berikut jika mereka mencoba untuk menutup alur kerja yang memiliki tugas keputusan yang tertunda.

- `FailWorkflowExecutionFailed`
- `CompleteWorkFlowExecutionFailed`
- `ContinueAsNewWorkflowExecutionFailed`
- `CancelWorkflowExecutionFailed`

Untuk informasi selengkapnya tentang salah satu kejadian yang terdaftar di atas, lihat Riwayat Kejadian dalam Referensi API Amazon SWF.

# Kuota Amazon SWF

Amazon SWF menempatkan kuota pada ukuran parameter alur kerja tertentu, seperti pada jumlah domain per akun dan pada ukuran riwayat eksekusi alur kerja. Kuota ini dirancang untuk mencegah alur kerja yang salah mengkonsumsi semua sumber daya sistem, tetapi bukan batas maksimal. Jika Anda menemukan bahwa aplikasi Anda sering melebihi kuota ini, Anda dapat meminta peningkatan service quotas.

Daftar Isi



## Kuota Akun Umum untuk Amazon SWF

- Domain terdaftar maksimum – 100

  Kuota ini mencakup domain terdaftar dan tidak lagi digunakan.
- Alur kerja maksimum dan tipe aktivitas – 10.000 setiap domain

  Kuota ini termasuk tipe terdaftar dan tidak lagi digunakan.
- Kuota panggilan API – Di luar lonjakan yang jarang terjadi, aplikasi dapat di-throttling jika mereka membuat sejumlah besar panggilan API dalam waktu yang sangat singkat.
- Ukuran permintaan maksimum – 1 MB per permintaan

  Ini adalah ukuran data total per permintaan API Amazon SWF, termasuk header permintaan dan semua data permintaan terkait lainnya.
- Respons terpotong untuk Hitungan APIs - Menunjukkan bahwa kuota internal telah tercapai dan bahwa responsnya bukan hitungan penuh.

Beberapa kueri akan mencapai kuota 1 MB secara internal yang disebutkan di atas sebelum mengembalikan respons penuh. Berikut ini dapat mengembalikan respons terpotong bukan jumlah penuh.

- CountClosedWorkflowExecutions
- CountOpenWorkflowExecutions
- CountPendingActivityTasks
- CountPendingDecisionTasks

Untuk tiap bagian, jika respons `truncated` diatur ke BETUL, jumlah kurang dari jumlah penuh. Kuota internal ini tidak dapat ditingkatkan.

- Jumlah maksimum tag – 50 tag per sumber daya.

  Mencoba untuk menambahkan tag di luar 50 akan menghasilkan kesalahan 400, `TooManyTagsFault`.

## Kuota pada Eksekusi Alur Kerja

- Eksekusi alur kerja terbuka maksimal – 100.000 per domain

  Jumlah ini mencakup eksekusi alur kerja anak.
- Waktu eksekusi alur kerja maksimum - 1 tahun. Ini adalah kuota keras yang tidak dapat diubah.
- Ukuran riwayat eksekusi alur kerja maksimum - 25.000 acara. Ini adalah kuota keras yang tidak dapat diubah.

  Praktik terbaik adalah menyusun setiap alur kerja sehingga riwayatnya tidak tumbuh melampaui 10.000 kejadian. Karena decider harus mengambil riwayat alur kerja, riwayat yang lebih kecil memungkinkan decider untuk menyelesaikan lebih cepat. Jika menggunakan Flow Framework, Anda dapat menggunakan ContinueAsNew untuk melanjutkan alur kerja dengan riwayat baru.
- Eksekusi alur kerja anak terbuka maksimal – 1.000 per eksekusi alur kerja

Jika kasus penggunaan mengharuskan Anda untuk melampaui kuota ini, Anda dapat menggunakan fitur Amazon SWF yang menyediakan untuk melanjutkan eksekusi dan struktur aplikasi menggunakan eksekusi alur kerja anak. Jika Anda menemukan bahwa Anda masih membutuhkan peningkatan kuota, lihatMeminta peningkatan kuota.

# Kuota tentang Eksekusi tugas

- Poller maksimum per daftar tugas – 1.000 per daftar tugas

  Anda dapat memiliki maksimal 1.000 poller yang sekaligus melakukan polling daftar tugas tertentu. Jika Anda melampau 1.000, Anda menerima `LimitExceededException`.

  > **Note**
  >
  > Meskipun maksimumnya adalah 1.000, Anda mungkin menemukan kesalahan `LimitExceededException` jauh sebelum kuota ini. Kesalahan ini tidak berarti tugas Anda tertunda. Sebaliknya, itu berarti Anda memiliki jumlah maksimum poller menganggur pada daftar tugas. Amazon SWF menetapkan batas ini untuk menghemat sumber daya di sisi klien dan server. Menetapkan batas mencegah jumlah poller yang berlebihan menunggu secara tidak perlu. Anda dapat mengurangi `LimitExceededException` kesalahan dengan menggunakan beberapa daftar tugas untuk mendistribusikan polling.

- Tugas maksimum terjadwal per detik – 2.000 per daftar tugas

  Anda dapat menjadwalkan maksimum 2.000 tugas per detik pada daftar tugas tertentu. Jika Anda melebihi 2.000, keputusan `ScheduleActivityTask` akan gagal dengan kesalahan`ACTIVITY_CREATION_RATE_EXCEEDED`.

  > **Note**
  >
  > Sementara maksimumnya adalah 2.000, Anda mungkin menemukan kesalahan `ACTIVITY_CREATION_RATE_EXCEEDED` jauh sebelum kuota ini. Untuk mengurangi kesalahan ini, gunakan beberapa daftar tugas untuk mendistribusikan beban.

- Waktu eksekusi tugas maksimum – 1 tahun (dibatasi oleh waktu eksekusi alur kerja maksimum)

  Anda dapat mengonfigurasi batas waktu aktivitas untuk menyebabkan kejadian batas waktu terjadi jika tahap tertentu dari eksekusi tugas aktivitas membutuhkan waktu terlalu lama.

- Waktu maksimum SWF akan menyimpan tugas dalam antrean – 1 tahun (dibatasi oleh kuota waktu eksekusi alur kerja)

  Anda dapat mengonfigurasi default batas waktu aktivitas selama pendaftaran aktivitas yang akan menyebabkan kejadian batas waktu terjadi jika tahap tertentu dari eksekusi tugas aktivitas

membutuhkan waktu terlalu lama. Anda juga dapat mengganti batas waktu aktivitas default ketika Anda menjadwalkan tugas aktivitas dalam kode decider Anda.

- Tugas aktivitas terbuka maksimum – 1.000 per eksekusi alur kerja.

  Kuota ini mencakup tugas aktivitas yang telah dijadwalkan dan diproses oleh pekerja.
- Timer pembukaan maksimum – 1.000 per eksekusi alur kerja
- Ukuran input/result data maksimum - 32.768 karakter

  Kuota ini memengaruhi data hasil eksekusi aktivitas atau alur kerja, input data saat menjadwalkan tugas aktivitas atau eksekusi alur kerja, dan input yang dikirim dengan sinyal eksekusi alur kerja.
- Keputusan maksimum dalam respons tugas keputusan – bervariasi

  Karena kuota 1 MB pada ukuran permintaan API maksimal, jumlah keputusan yang dikembalikan dalam satu panggilan untuk `RespondDecisionTaskCompleted` akan dibatasi sesuai dengan ukuran data yang digunakan oleh setiap keputusan, termasuk ukuran data input yang disediakan untuk tugas aktivitas terjadwal atau eksekusi alur kerja.

# Kuota throttling Amazon SWF

Selain service quotas yang dijelaskan sebelumnya, panggilan API Amazon SWF dan kejadian keputusan tertentu di-throttling untuk memelihara bandwidth layanan, menggunakan skema bucket token. Jika tingkat permintaan Anda secara konsisten melebihi tingkat yang tercantum di sini, Anda dapat meminta peningkatan kuota throttle.

Kuota pembatasan dan keputusan sama di semua wilayah.

## Melambatkan kuota untuk semua Wilayah

Kuota berikut berlaku di tingkat akun individu. Anda juga dapat meminta kenaikan kuota berikut. Untuk informasi tentang melakukan ini, lihatMeminta peningkatan kuota.

| Nama API | Ukuran bucket | Tingkat isi ulang per detik |
|---|---|---|
| `CountClosedWorkflowExecutions` | 2000 | 6 |
| `CountOpenWorkflowExecutions` | 2000 | 6 |

| Nama API | Ukuran bucket | Tingkat isi ulang per detik |
|---|---|---|
| CountPendingActivityTasks | 200 | 6 |
| CountPendingDecisionTasks | 200 | 6 |
| DeleteActivityType | 200 | 6 |
| DeleteWorkflowType | 200 | 6 |
| DeprecateActivityType | 200 | 6 |
| DeprecateDomain | 100 | 6 |
| DeprecateWorkflowType | 200 | 6 |
| DescribeActivityType | 2000 | 6 |
| DescribeDomain | 200 | 6 |
| DescribeWorkflowExecution | 2000 | 6 |
| DescribeWorkflowType | 2000 | 6 |
| GetWorkflowExecutionHistory | 2000 | 60 |
| ListActivityTypes | 200 | 6 |
| ListClosedWorkflowExecutions | 200 | 6 |
| ListDomains | 100 | 6 |
| ListOpenWorkflowExecutions | 200 | 48 |
| ListTagsForResource | 50 | 30 |
| ListWorkflowTypes | 200 | 6 |
| PollForActivityTask | 2000 | 200 |
| PollForDecisionTask | 2000 | 200 |

| Nama API | Ukuran bucket | Tingkat isi ulang per detik |
|---|---|---|
| RecordActivityTaskHeartbeat | 2000 | 160 |
| RegisterActivityType | 200 | 60 |
| RegisterDomain | 100 | 6 |
| RegisterWorkflowType | 200 | 60 |
| RequestCancelWorkflowExecution | 2000 | 30 |
| RespondActivityTaskCanceled | 2000 | 200 |
| RespondActivityTaskCompleted | 2000 | 200 |
| RespondActivityTaskFailed | 2000 | 200 |
| RespondDecisionTaskCompleted | 2000 | 200 |
| SignalWorkflowExecution | 2000 | 30 |
| StartWorkflowExecution | 2000 | 200 |
| TagResource | 50 | 30 |
| TerminateWorkflowExecution | 2000 | 60 |
| UndeprecateActivityType | 200 | 6 |
| UndeprecateDomain | 100 | 6 |
| UndeprecateWorkflowType | 200 | 6 |
| UntagResource | 50 | 30 |

## Kuota keputusan untuk semua Wilayah

Kuota berikut berlaku di tingkat akun individu. Anda juga dapat meminta kenaikan kuota berikut. Untuk informasi tentang melakukan ini, lihatMeminta peningkatan kuota.

| Nama API | Ukuran bucket | Tingkat isi ulang per detik |
|---|---|---|
| `RequestCancelExternalWorkflowExecution` | 1200 | 120 |
| `ScheduleActivityTask` | 1000 | 200 |
| `SignalExternalWorkflowExecution` | 1200 | 120 |
| `StartChildWorkflowExecution` | 500 | 12 |
| `StartTimer` | 2000 | 200 |

## Kuota tingkat alur kerja

Kuota berikut berlaku di tingkat alur kerja dan tidak dapat ditingkatkan.

| Nama API | Ukuran bucket | Tingkat isi ulang per detik |
|---|---|---|
| `GetWorkflowExecutionHistory` | 400 | 200 |
| `SignalWorkflowExecution` | 1000 | 1000 |
| `RecordActivityTaskHeartbeat` | 1000 | 1000 |
| `RequestCancelWorkflowExecution` | 200 | 200 |

# Meminta peningkatan kuota

Untuk informasi selengkapnya, lihat kuota layanan AWS di Referensi Umum AWS.

# Sumber daya tambahan dan info referensi untuk Amazon SWF

Bab ini menyediakan sumber daya tambahan dan informasi referensi yang berguna ketika mengembangkan alur kerja dengan Amazon SWF.

Topik



## Tipe Batas Waktu Amazon SWF

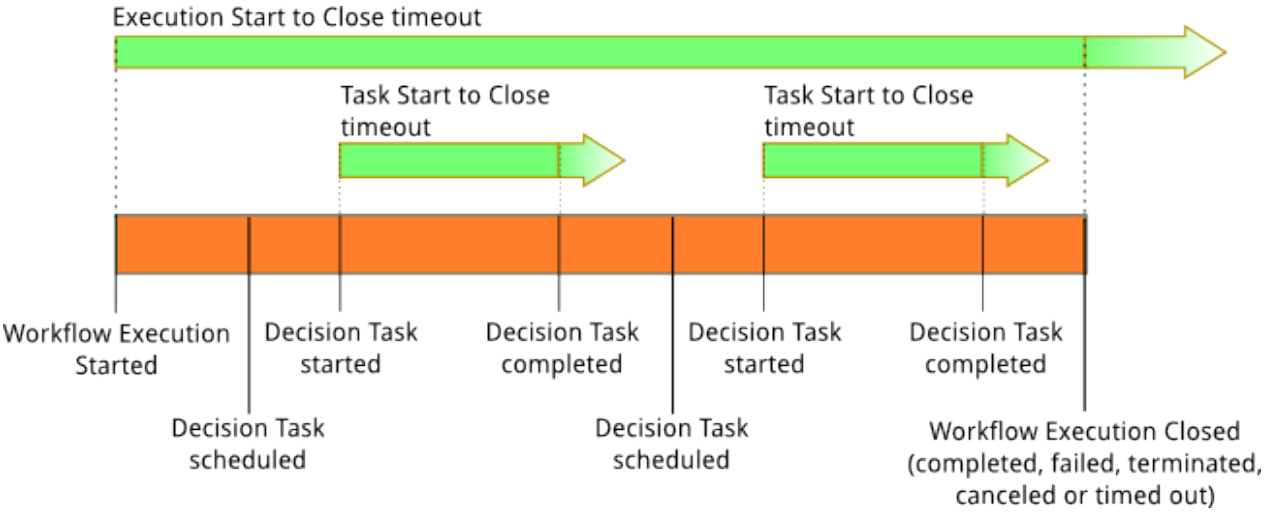
Untuk memastikan eksekusi alur kerja berjalan dengan benar, Anda dapat mengatur berbagai jenis batas waktu dengan Amazon SWF. Beberapa batas waktu menentukan berapa lama alur kerja dapat berjalan secara keseluruhan. Batas waktu lainnya menentukan berapa lama tugas aktivitas dapat berjalan sebelum ditugaskan ke pekerja dan berapa lama waktu yang bisa mereka gunakan untuk menyelesaikan dari waktu yang dijadwalkan. Semua batas waktu di API Amazon SWF ditentukan dalam hitungan detik. Amazon SWF juga mendukung string `NONE` sebagai nilai batas waktu, yang menunjukkan tidak ada batas waktu.

Untuk batas waktu yang berkaitan dengan tugas keputusan dan tugas aktivitas, Amazon SWF menambahkan sebuah kejadian ke riwayat eksekusi alur kerja. Atribut acara memberikan informasi tentang jenis batas waktu apa yang terjadi dan tugas keputusan atau aktivitas tugas yang terpengaruh. Amazon SWF juga menjadwalkan tugas keputusan. Ketika penentu menerima tugas keputusan baru, ia akan melihat peristiwa batas waktu dalam sejarah dan mengambil tindakan yang tepat dengan memanggil tindakan. RespondDecisionTaskCompleted

Tugas dianggap terbuka dari saat yang dijadwalkan sampai ditutup. Oleh karena itu tugas dilaporkan sebagai terbuka sementara pekerja sedang memproses tugas. Tugas ditutup ketika seorang pekerja melaporkannya sebagai selesai, dibatalkan, atau gagal. Sebuah tugas juga dapat ditutup oleh Amazon SWF sebagai hasil dari batas waktu.

# Batas Waktu dalam Alur Kerja dan Tugas Keputusan

Diagram berikut menunjukkan bagaimana alur kerja dan batas waktu keputusan terkait dengan masa pakai alur kerja:



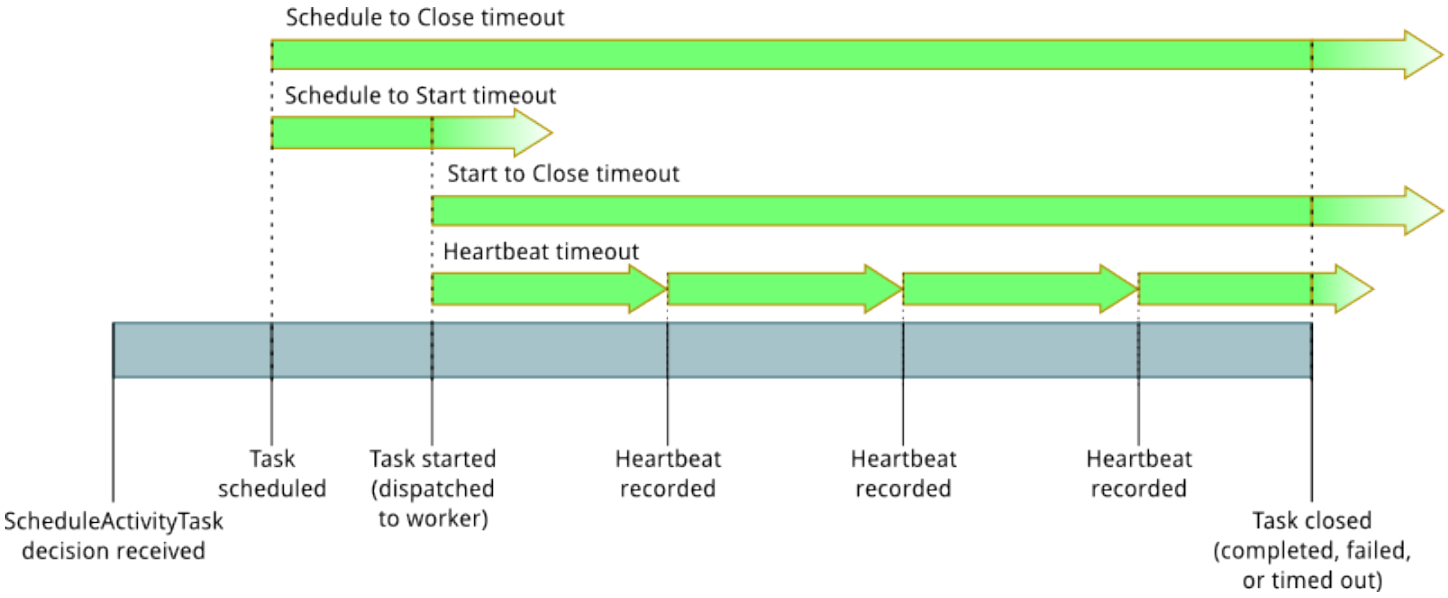

Ada dua tipe batas waktu timeout yang relevan dengan alur kerja dan tugas keputusan:

- Alur Kerja Start to Close (**`timeoutType: START_TO_CLOSE`**) – Batas Waktu ini menentukan waktu maksimum yang dapat diambil untuk menyelesaikan eksekusi alur kerja. Batas ini diatur sebagai default selama pendaftaran alur kerja, tetapi dapat diambil alih dengan nilai yang berbeda ketika alur kerja dimulai. Jika batas waktu ini terlampaui, Amazon SWF menutup eksekusi alur kerja dan menambahkan WorkflowExecutionTimedOutjenis peristiwa ke riwayat eksekusi alur kerja. Selain `timeoutType`, atribut kejadian menentukan `childPolicy` yang berlaku untuk eksekusi alur kerja ini. Kebijakan anak menentukan bagaimana eksekusi alur kerja anak ditangani jika eksekusi alur kerja induk mencapai batas waktu atau berakhir. Contohnya, jika `childPolicy` diatur menjadi TERMINATE (AKHIRI), maka eksekusi alur kerja anak akan diakhiri. Setelah eksekusi alur kerja mencapai batas waktu, Anda tidak dapat mengambil tindakan apa pun selain panggilan visibilitas.
- Tugas Keputusan Start to Close (**`timeoutType: START_TO_CLOSE`**) – Batas waktu ini menentukan waktu maksimum yang dapat digunakan decider yang sesuai untuk menyelesaikan tugas keputusan. Hal ini diatur selama pendaftaran tipe alur kerja. Jika batas waktu ini terlampaui, tugas ditandai sebagai kehabisan waktu dalam riwayat eksekusi alur kerja, dan Amazon SWF menambahkan jenis peristiwa ke riwayat alur kerja. DecisionTaskTimedOut Atribut event akan menyertakan IDs untuk peristiwa yang sesuai dengan saat tugas keputusan ini dijadwalkan (`scheduledEventId`) dan saat dimulai (`startedEventId`). Selain menambahkan kejadian tersebut, Amazon SWF juga menjadwalkan tugas keputusan baru untuk memberitahu

decider bahwa tugas keputusan ini mencapai batas waktu. Setelah batas waktu ini terjadi, upaya untuk menyelesaikan tugas keputusan yang mencapai batas waktu menggunakan `RespondDecisionTaskCompleted` akan gagal.

## Batas Waktu dalam Tugas Aktivitas

Diagram berikut menunjukkan bagaimana batas waktu terkait dengan masa pakai tugas kegiatan:




Ada empat tipe batas waktu yang relevan dengan tugas aktivitas:

- Tugas Aktivitas Start to Close (**`timeoutType: START_TO_CLOSE`**) – Batas waktu ini menentukan waktu maksimum yang dapat diambil seorang pekerja aktivitas untuk memproses tugas setelah pekerja menerima tugas. Upaya untuk menutup tugas aktivitas yang habis waktu menggunakan RespondActivityTaskCanceled, RespondActivityTaskCompleted, dan RespondActivityTaskFailedakan gagal.
- Tugas Aktivitas Heartbeat (**`timeoutType: HEARTBEAT`**) – Batas waktu ini menentukan waktu maksimum untuk menjalankan tugas sebelum menyediakan kemajuannya melalui tindakan `RecordActivityTaskHeartbeat`.
- Tugas Aktivitas Schedule to Start (**`timeoutType: SCHEDULE_TO_START`**) - Batas waktu ini menentukan berapa lama Amazon SWF menunggu sebelum mencapai batas waktu tugas aktivitas jika tidak ada pekerja yang tersedia untuk melakukan tugas. Setelah batas waktu tercapai, tugas yang kedaluwarsa tidak akan ditugaskan ke pekerja lain.
- Tugas Aktivitas Schedule to Close (**`timeoutType: SCHEDULE_TO_CLOSE`**) - Batas waktu ini menentukan berapa lama waktu yang bisa digunakan sejak waktu yang dijadwalkan hingga tugas

selesai. Sebagai praktik terbaik, nilai ini tidak boleh lebih besar dari jumlah schedule-to-start batas waktu tugas dan batas start-to-close waktu tugas.

> **Note**
>
> Setiap tipe batas waktu memiliki nilai default, yang umumnya diatur ke `NONE` (tak terbatas). Namun, waktu maksimum untuk setiap eksekusi aktivitas terbatas pada satu tahun.

Anda menetapkan nilai default untuk nilai ini selama pendaftaran tipe aktivitas, tetapi Anda dapat mengambil alih dengan nilai baru ketika Anda menjadwalkan tugas aktivitas. Ketika salah satu batas waktu ini terjadi, Amazon SWF akan menambahkan ActivityTaskTimedOutjenis peristiwa ke riwayat alur kerja. Atribut `timeoutType` nilai dari kejadian ini akan menentukan batas waktu mana yang akan terjadi. Untuk setiap batas waktu, nilai `timeoutType` ditampilkan dalam tanda kurung. Atribut event juga akan menyertakan IDs untuk peristiwa yang sesuai dengan saat tugas aktivitas dijadwalkan (`scheduledEventId`) dan saat dimulai (`startedEventId`). Selain menambahkan kejadian tersebut, Amazon SWF juga menjadwalkan tugas keputusan baru untuk memberitahu decider bahwa batas waktu terjadi.

## Titik Akhir Amazon Simple Workflow Service

Daftar Wilayah dan Titik Akhir Amazon SWF saat ini disediakan di Referensi Umum Amazon Web, bersama dengan titik akhir untuk layanan lainnya.

Domain Amazon SWF serta semua alur kerja dan aktivitas terkait harus berada dalam wilayah yang sama untuk berkomunikasi satu sama lain. Selain itu, setiap domain terdaftar, alur kerja, dan aktivitas dalam suatu wilayah tidak ada di wilayah lain. Misalnya, jika Anda membuat domain bernama "MySampleDomain" di us-east-1 dan di us-west-2, domain tersebut ada sebagai domain terpisah: tidak ada alur kerja, daftar tugas, aktivitas, atau data yang terkait dengan domain Anda yang dibagikan di seluruh wilayah.

Jika Anda menggunakan AWS sumber daya lain dalam alur kerja Anda, seperti EC2 instans Amazon, ini juga harus ada di wilayah yang sama dengan sumber daya Amazon SWF Anda. Satu-satunya pengecualian untuk ini adalah layanan yang mencakup wilayah, seperti Amazon S3 dan IAM. Anda dapat mengakses layanan ini dari alur kerja yang ada di wilayah mana pun yang mendukungnya.

# Dokumentasi Tambahan untuk Amazon Simple Workflow Service

Selain Panduan Developer ini, Anda juga dapat membaca informasi lain di dokumentasi berikut.

## Referensi API Amazon Simple Storage Service

Referensi API Amazon Simple Workflow Service menyediakan informasi detail tentang Amazon SWF HTTP API, termasuk tindakan, struktur permintaan dan respons serta kode kesalahan.

## AWS Flow Framework Dokumentasi

AWS Flow Framework adalah kerangka kerja pemrograman yang menyederhanakan proses penerapan aplikasi asinkron terdistribusi yang menggunakan Amazon SWF untuk mengelola alur kerja dan aktivitas mereka, sehingga Anda dapat berfokus pada penerapan logika alur kerja Anda.

Masing-masing AWS Flow Framework dirancang untuk bekerja secara idiomatis dalam bahasa yang dirancang, sehingga Anda dapat bekerja secara alami dengan bahasa pilihan Anda untuk mengimplementasikan alur kerja dengan semua manfaat Amazon SWF.

Ada AWS Framework Framework untuk Java. Panduan Pengembang AWS Flow Framework untuk Java memberikan informasi tentang cara mendapatkan, mengatur, dan menggunakan AWS Flow Framework untuk Java.

## AWS Dokumentasi SDK

Kit Pengembangan Perangkat AWS Lunak (SDKs) menyediakan akses ke Amazon SWF dalam berbagai bahasa pemrograman. SDKs Ikuti API HTTP dengan cermat, tetapi juga menyediakan antarmuka pemrograman khusus bahasa untuk beberapa fitur Amazon SWF. Anda dapat mengetahui informasi lebih lanjut tentang setiap SDK dengan mengunjungi tautan berikut.

> **Note**
>
> Hanya SDKs yang memiliki dukungan untuk Amazon SWF pada saat penulisan yang tercantum di sini. Untuk daftar lengkap yang tersedia AWS SDKs, kunjungi halaman Alat untuk Amazon Web Services.

Java

AWS SDK for Java Ini menyediakan Java API untuk layanan AWS infrastruktur.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman Dokumentasi AWS SDK for Java. Anda juga dapat langsung menuju bagian Amazon SWF di referensi SDK dengan mengikuti tautan berikut:

- `Class: AmazonSimpleWorkflowClient`
- `Class: AmazonSimpleWorkflowAsyncClient`
- `Interface: AmazonSimpleWorkflow`
- `Interface: AmazonSimpleWorkflowAsync`

JavaScript

AWS SDK for JavaScript Ini memungkinkan pengembang untuk membangun pustaka atau aplikasi yang menggunakan AWS layanan menggunakan sederhana dan easy-to-use API yang tersedia baik di browser atau di dalam aplikasi Node.js di server.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman Dokumentasi AWS SDK for JavaScript. Anda juga dapat langsung menuju bagian Amazon SWF di referensi SDK dengan mengikuti tautan ini:

- `Class: AWS.SimpleWorkflow`

.NET

AWS SDK for .NET Ini adalah paket tunggal yang dapat diunduh yang mencakup templat proyek Visual Studio, pustaka AWS .NET, sampel kode C #, dan dokumentasi. Ini AWS SDK for .NET memudahkan pengembang Windows untuk membangun aplikasi.NET untuk Amazon SWF dan layanan lainnya.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman Dokumentasi AWS SDK for .NET. Anda juga dapat langsung menuju bagian Amazon SWF di referensi SDK dengan mengikuti tautan berikut:

- `Namespace: Amazon.SimpleWorkflow`
- `Namespace: Amazon.SimpleWorkflow.Model`

PHP

AWS SDK for PHP Ini menyediakan antarmuka pemrograman PHP ke Amazon SWF.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman Dokumentasi AWS SDK for PHP. Anda juga dapat langsung menuju bagian Amazon SWF di referensi SDK dengan mengikuti tautan ini:

- `Class: SwfClient`

Python

AWS SDK for Python (Boto) Ini menyediakan antarmuka pemrograman Python ke Amazon SWF.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman boto: Antarmuka Python untuk Amazon Web Services. Anda juga dapat langsung menuju bagian Amazon SWF di dokumentasi dengan mengikuti tautan berikut:

- Amazon SWF Tutorial
- Amazon SWF Referensi

Ruby

AWS SDK for Ruby Ini menyediakan antarmuka pemrograman Ruby ke Amazon SWF.

Untuk menampilkan dokumentasi yang tersedia, lihat halaman Dokumentasi AWS SDK for Ruby. Anda juga dapat langsung menuju bagian Amazon SWF di referensi SDK dengan mengikuti tautan ini:

- Kelas: AWS::Simple Alur Kerja

## AWS CLI Dokumentasi

The AWS Command Line Interface (AWS CLI) adalah alat terpadu untuk mengelola AWS layanan Anda. Dengan hanya satu alat untuk mengunduh dan mengkonfigurasi, Anda dapat mengontrol beberapa AWS layanan dari baris perintah dan mengotomatiskannya melalui skrip.

Untuk informasi lebih lanjut tentang AWS CLI, lihat AWS Command Line Interfacehalaman.

Untuk gambaran umum perintah yang tersedia untuk Amazon SWF, lihat swf di Referensi Perintah AWS CLI .

# Sumber Daya Web untuk Amazon Simple Workflow Service

Ada sejumlah sumber daya Web yang dapat Anda gunakan untuk mempelajari lebih lanjut tentang Amazon SWF atau untuk mendapatkan bantuan tentang penggunaan layanan dan pengembangan alur kerja.

## Forum Amazon SWF

Forum Amazon SWF menyediakan tempat bagi Anda untuk berkomunikasi dengan developer Amazon SWF lainnya dan anggota tim pengembangan Amazon SWF di Amazon untuk mengajukan pertanyaan dan mendapatkan jawaban.

Anda dapat mengunjungi forum di: Forum: Amazon Simple Workflow Service.

## FAQ Amazon SWF

FAQ Amazon SWF memberikan jawaban atas pertanyaan yang sering diajukan tentang Amazon SWF, termasuk gambaran umum kasus penggunaan umum, perbedaan antara Amazon SWF dan layanan lainnya, dan sejumlah pertanyaan lain.

Anda dapat mengakses FAQ di sini: FAQ Amazon SWF.

## Video Amazon SWF

Saluran Amazon Web Services on YouTube menyediakan pelatihan video untuk semua Layanan Web Amazon, termasuk Amazon SWF. Untuk daftar lengkap video terkait Amazon SWF, gunakan kueri berikut: Alur Kerja Sederhana di Amazon Web Services

# Opsi migrasi untuk Ruby Flow

The AWS Flow Framework for Ruby tidak lagi dalam pengembangan aktif. Sementara kode yang ada akan terus bekerja tanpa batas waktu, tidak akan ada fitur atau versi baru. Topik ini akan mencakup opsi penggunaan dan migrasi untuk terus bekerja dengan Amazon SWF, dan informasi tentang cara melakukan migrasi ke Step Functions.

| Opsi | Deskripsi |
| --- | --- |
| Terus menggunakan Ruby Flow Framework | Untuk saat ini, Ruby Flow Framework akan terus bekerja. Jika Anda tidak melakukan apa-apa, kode Anda akan terus berfungsi seperti biasa. Berencana untuk bermigrasi dari AWS Flow Framework Ruby dalam waktu dekat. |
| Migrasi ke Java Flow Framework | Java Flow Framework tetap berada dalam pengembangan aktif dan akan terus menerima fitur serta update baru. |

| Opsi | Deskripsi |
| --- | --- |
| Migrasi ke Step Functions | Step Functions menyediakan cara untuk mengkoordinasikan komponen aplikasi terdistribusi menggunakan alur kerja visual yang dikendalikan oleh mesin state. |
| Gunakan API SWF secara langsung, tanpa Flow Framework | Anda dapat terus bekerja di Ruby dan menggunakan API SWF secara langsung alih-alh menggunakan Ruby Flow Framework. |

Keuntungan yang diberikan Flow Framework, baik Ruby maupun Java, adalah bahwa Framework ini memungkinkan Anda untuk berfokus pada logika alur kerja Anda. Kerangka kerja menangani banyak detail komunikasi dan koordinasi, dan beberapa kompleksitas yang diabstraksikan. Anda dapat terus memiliki tingkat abstraksi yang sama dengan bermigrasi ke Java Flow Framework, atau Anda dapat langsung berinteraksi dengan SDK Amazon SWF secara langsung.

## Lanjutkan untuk menggunakan Ruby Flow Framework

The AWS Flow Framework for Ruby akan terus berfungsi seperti sekarang dalam jangka pendek. Jika Anda memiliki alur kerja yang ditulis dalam AWS Flow Framework untuk Ruby, ini akan terus berfungsi. Tanpa pembaruan, dukungan, atau perbaikan keamanan, langkah terbaik adalah dengan memiliki rencana pasti untuk bermigrasi dari AWS Flow Framework untuk Ruby dalam waktu dekat.

## Migrasi ke Java Flow Framework

The AWS Flow Framework for Java akan tetap dalam pengembangan aktif. Secara konseptual, AWS Flow Framework untuk Java mirip AWS Flow Framework dengan Ruby: Anda masih dapat fokus pada logika alur kerja Anda, dan kerangka kerja akan membantu mengelola logika penentu Anda, dan akan membuat aspek lain dari Amazon SWF lebih mudah dikelola.

- AWS Flow Framework untuk Java
- AWS Flow Framework untuk Referensi API Java

## Migrasi ke Step Functions

AWS Step Functions menyediakan layanan yang mirip dengan Amazon SWF, tetapi di mana logika alur kerja Anda dikendalikan oleh mesin status. Step Functions memungkinkan Anda untuk

mengkoordinasikan komponen aplikasi dan microservices terdistribusi menggunakan alur kerja visual. Anda membangun aplikasi dari komponen individu yang masing-masing menjalankan fungsi diskrit, atau tugas, memungkinkan Anda untuk menskalakan dan mengubah aplikasi dengan cepat. Step Functions menyediakan cara yang dapat diandalkan untuk mengkoordinasikan komponen dan langkah melalui fungsi aplikasi Anda. Sebuah konsol grafis menyediakan cara untuk memvisualisasikan komponen aplikasi Anda sebagai serangkaian langkah. Konsol ini secara otomatis memicu dan melacak setiap langkah, dan mencoba lagi ketika terdapat kesalahan, sehingga aplikasi Anda melakukan eksekusi secara berurutan dan seperti yang diharapkan, setiap waktu. Step Functions mencatat status setiap langkah, jadi ketika terjadi kesalahan, Anda dapat mendiagnosa dan men-debug masalah dengan cepat.

Dalam Step Functions, Anda mengelola koordinasi tugas Anda menggunakan mesin state, ditulis dalam JSON deklaratif, yang ditentukan menggunakan Bahasa Amazon States. Dengan menggunakan mesin state, Anda tidak harus menulis dan memelihara program decider untuk mengontrol logika aplikasi Anda. Step Functions menyediakan pendekatan intuitif, produktif, dan tangkas untuk mengkoordinasi komponen aplikasi menggunakan alur kerja visual. Anda harus mempertimbangkan AWS Step Functions untuk menggunakan semua aplikasi baru Anda, dan Step Functions menyediakan platform yang sangat baik untuk bermigrasi ke alur kerja yang saat ini telah Anda terapkan di AWS Flow Framework for Ruby.

Untuk membantu memigrasikan tugas Anda ke Step Functions, sambil terus memanfaatkan kemampuan bahasa Ruby Anda, Step Functions menyediakan contoh pekerja aktivitas Ruby. Contoh ini menggunakan praktik terbaik untuk menerapkan pekerja aktivitas, dan dapat digunakan sebagai templat untuk memigrasikan logika tugas Anda ke Step Functions. Untuk informasi lebih lanjut, lihat topik Contoh Pekerja Aktivitas di Ruby di Panduan Developer AWS Step Functions.

> **Note**
>
> Bagi banyak pelanggan, bermigrasi ke Step Functions dari AWS Flow Framework for Ruby adalah pilihan terbaik. Namun, jika Anda mengharuskan sinyal campur tangan dalam proses Anda, atau jika Anda perlu meluncurkan proses turunan yang mengembalikan hasil ke induk, pertimbangkan untuk menggunakan Amazon SWF API secara langsung, atau bermigrasi ke AWS Flow Framework for Java.

Untuk informasi lebih lanjut tentang AWS Step Functions, lihat:

- AWS Step Functions Panduan Pengembang

- AWS Step Functions Referensi API
- AWS Step Functions Referensi Baris Perintah

## Gunakan API Amazon SWF secara langsung

Sementara AWS Flow Framework untuk Ruby mengelola beberapa kompleksitas Amazon SWF, Anda juga dapat menggunakan Amazon SWF API secara langsung. Menggunakan API secara langsung memungkinkan Anda untuk membangun alur kerja di mana Anda memiliki kontrol penuh atas pelaksanaan tugas dan pengkoordinasiannya, tanpa khawatir tentang kompleksitas mendasar seperti melacak kemajuan mereka dan mempertahankan keadaan mereka.

- Panduan Pengembang Layanan Alur Kerja Sederhana Amazon
- Referensi API Layanan Alur Kerja Sederhana Amazon

# Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak perilisan terakhir dari Panduan Developer Amazon Simple Workflow Service.

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Pembaruan khusus dokumentasi | Amazon SWF sekarang menyertakan bagian tentang Pemberitahuan AWS Pengguna, Layanan AWS yang bertindak sebagai lokasi pusat untuk AWS pemberita huan Anda di. Konsol Manajemen AWS Untuk informasi selengkapnya, lihat Menggunakan Notifikasi Pengguna AWS dengan Amazon Simple Workflow Service. | 4 Mei 2023 |
| Perbarui | Amazon SWF sekarang menyediakan pengalaman konsol baru untuk mengelola alur kerja SWF dan tindakan terkait eksekusi mereka. Untuk informasi selengkapnya, lihat tutorial konsol Amazon SWF. | 12 September 2022 |
| Perbarui | Memperbarui Kuota tentang Eksekusi tugas bagian untuk menyertakan`Maximum tasks scheduled per second`, dan Metrik Amazon SWF untuk CloudWatch halaman untuk menyertakan informasi tentang menggunak an nama sumber daya non-ASCII dengan. CloudWatch | 12 Mei 2021 |
| Fitur baru | Amazon Simple Workflow Service sekarang mendukung Amazon EventBridge. Untuk informasi selengkapnya, lihat:<br>• EventBridge untuk Amazon SWF<br>• EventBridge Panduan Pengguna | 18 Desember 2020 |
| Fitur baru | Amazon Simple Workflow Service mendukung izin IAM menggunakan tanda. Untuk informasi lebih lanjut, lihat berikut ini.<br>• Tag di Amazon SWF | 20 Juni 2019 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| | • Kelola tag<br>• Menandai eksekusi alur kerja<br>• Kontrol akses ke domain dengan tag<br>• `TagResource`<br>• `UntagResource`<br>• `ListTagsForResource`<br>• `RegisterDomain` | |
| Fitur baru | Amazon Simple Workflow Service sekarang tersedia di wilayah Eropa (Stockholm). | 12 Desember 2018 |
| Perbarui | Memperbaiki topik Amazon Simple Workflow Service pada CloudTrail integrasi. Lihat Merekam panggilan API dengan AWS CloudTrail. | 7 Agustus 2018 |
| Perbarui | Menambahkan informasi tentang `PendingTasks` metrik baru untuk CloudWatch. Untuk informasi selengkapnya, lihat Metrik Amazon SWF. | 18 Juni 2018 |
| Pembaruan | Meningkatkan penyorotan sintaks dalam sampel kode. | 29 Maret 2018 |
| Pembaruan | Menambahkan topik yang menjelaskan opsi bagi pengguna Ruby Flow untuk bermigrasi dari platform tersebut. Untuk informasi selengkapnya, lihat Opsi migrasi untuk Ruby Flow. | 9 Maret 2018 |
| Pembaruan | Meningkatkan navigasi pada topik konsep lanjutan. Lihat Konsep alur kerja lanjutan di Amazon SWF. | 19 Februari 2018 |
| Perbarui | Dokumentasi CloudWatch metrik yang ditingkatkan dengan menambahkan informasi statistik yang valid. Lihat Metrik Amazon SWF untuk CloudWatch. | 4 Desember 2017 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Pembaruan | Mengubah TOC untuk meningkatkan struktur dokumen. Menambahkan informasi baru pada Metrik Kejadian API dan Keputusan. | 9 November 2017 |
| Pembaruan | Memperbarui bagian Kuota Amazon SWF untuk menyertakan batas throttling untuk semua wilayah. | 18 Oktober 2017 |
| Pembaruan | Merubah `task_list` ke `workflowId` di Memulai dengan Amazon SWF untuk menghindari kekeliruan dengan `activity_list` . | 25 Juli 2017 |
| Pembaruan | Bersihkan contoh kode di seluruh panduan ini. | 5 Juni 2017 |
| Pembaruan | Menyederhanakan dan meningkatkan organisasi dan isi panduan ini. | 19 Mei 2017 |
| Pembaruan | Pembaruan dan perbaikan tautan. | 16 Mei 2017 |
| Pembaruan | Pembaruan dan perbaikan tautan. | 1 Oktober 2016 |
| Dukungan tugas Lambda | Anda dapat menentukan tugas Lambda selain tugas Aktivitas tradisional di alur kerja Anda. Untuk informasi selengkapnya, lihat AWS Lambda tugas di Amazon SWF. | 21 Juli 2015 |
| Dukungan untuk mengatur prioritas tugas | Amazon SWF sekarang menyertakan dukungan untuk mengatur prioritas tugas pada daftar tugas, dan akan berusaha untuk memberi prioritas bagi tugas dengan prioritas yang lebih tinggi sebelum tugas dengan prioritas yang lebih rendah. Informasi tentang cara menetapkan prioritas tugas untuk alur kerja dan untuk aktivitas disediakan di Menetapkan prioritas tugas di Amazon SWF. | 17 Desember 2014 |
| Perbarui | Menambahkan topik baru yang menjelaskan cara mencatat panggilan API Amazon SWF menggunakan CloudTrail:. Merekam panggilan API dengan AWS CloudTrail | 8 Mei 2014 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Perbarui | Dua topik baru yang terkait dengan CloudWatch metrik untuk Amazon SWF telah ditambahkanMetrik Amazon SWF untuk CloudWatch:, yang menyediakan daftar dan deskripsi metrik yang didukung, dan, yang menyediak an informasi tentang cara melihat metrik Melihat Metrik Amazon SWF untuk menggunakan CloudWatch Konsol Manajemen AWS dan menyetel alarm dengan. Konsol Manajemen AWS | 28 April 2014 |
| Pembaruan | Menambahkan bagian baru: Sumber daya tambahan dan info referensi untuk Amazon SWF. Bagian ini menyediak an beberapa informasi referensi layanan dan memberikan informasi tentang dokumentasi, contoh, kode dan sumber daya web tambahan lainnya untuk developer Amazon SWF. | 19 Maret 2014 |
| Pembaruan | Menambahkan tutorial alur kerja. Lihat Memulai dengan Amazon SWF. | 25 Oktober 2013 |
| Pembaruan | Menambahkan Informasi dan contoh AWS CLI. | 26 Agustus 2013 |
| Pembaruan | Pembaruan dan perbaikan. | 1 Agustus 2013 |
| Pembaruan | Memperbarui dokumen untuk menjelaskan cara menggunakan IAM untuk kontrol akses. | 22 Februari 2013 |
| Perilisan awal | Ini adalah perilisan pertama dari Panduan Developer Amazon Simple Workflow Service. | 16 Oktober 2012 |

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.