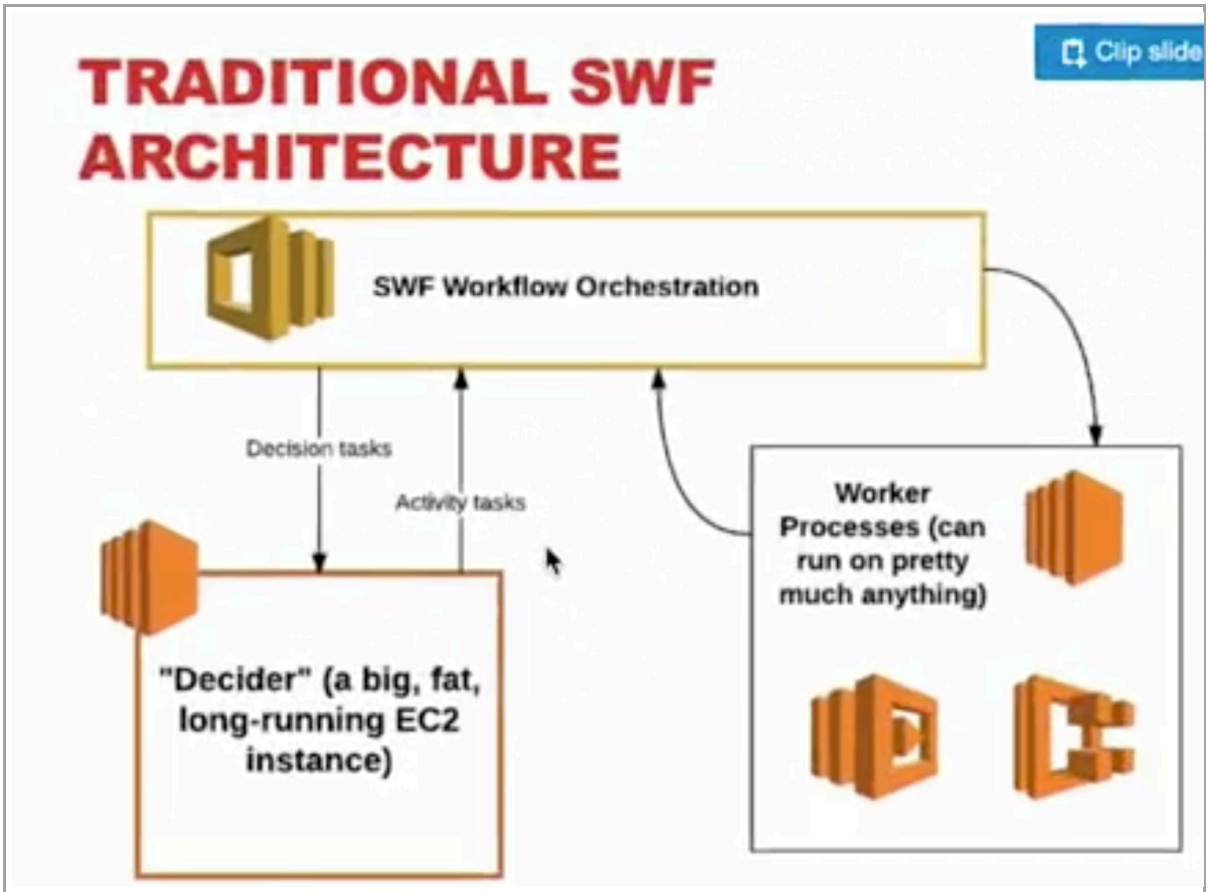
SWF - Simple Workflow

- Amazon SWF provides useful guarantees around task assignments.
- It ensures that a task is never duplicated and is assigned only once.
- Thus, even though you may have multiple workers for a particular activity type (or a number of instances of a decider), Amazon SWF will give a specific task to only one worker (or one decider instance).
- Additionally, Amazon SWF keeps at most one decision task outstanding at a time for a workflow execution.
- Thus, you can run multiple decider instances without worrying about two instances operating on the same execution simultaneously.
- These facilities enable you to coordinate your workflow without worrying about duplicate, lost, or conflicting tasks.

- By default, each workflow execution can run for a maximum of 1 year in Amazon SWF.
- Amazon SWF does not take any special action if a workflow execution is idle for an extended period of time.
- Idle executions are subject to the timeouts that you configure.
- For example, if you have set the maximum duration for an execution to be 1 day, then an idle execution will be timed out if it exceeds the 1 day limit.
- Idle executions are also subject to the Amazon SWF limit on how long an execution can run (1 year).

NOTE:
Alternate model is **'STEP FUNCTIONS'**



CONCEPTS

Concepts

- **Workflow**
  - A set of activities that carry out some objective, together with logic that coordinates the activities.
  - Workflows coordinate and manage the execution of activities that can be run asynchronously across multiple computing devices and that can feature both sequential and parallel processing.
  - Each workflow runs in an AWS resource called a *domain*, which controls the workflow's scope.
  - An AWS account can have multiple domains, each of which can contain multiple workflows, but workflows in different domains can't interact.
  - When you register an activity to a workflow, you provide information such as a name and version, and some timeout values based on how long you expect the activity to take.
- **Activity Task**
  - An **activity task** tells an activity worker to perform its function.
  - SWF stores tasks and assigns them to workers when they are ready, tracks their progress, and maintains their state, including details on their completion.
  - To coordinate tasks, you write a program that gets the latest state of each task from SWF and uses it to initiate subsequent tasks.
  - Activity tasks can run synchronously or asynchronously. They can be distributed across multiple computers, potentially in different geographic regions, or they can all run on the same computer.
  - Activity tasks for a running workflow execution appear on the *activity task list*, which is provided when you schedule an activity in the workflow.
  - If you don't specify a task list when scheduling an activity task, the task is automatically placed on the default task list.

- **Lambda task**
  - Executes a Lambda function instead of a traditional SWF activity.
- **Decision task**
  - A Decision task tells a decider that the state of the workflow execution has changed so that the decider can determine the next activity that needs to be performed. The decision task contains the current workflow history.
  - SWF assigns each decision task to exactly one decider and allows only one decision task at a time to be active in a workflow execution.
- **Workflow Starter**
  - Any application that can initiate workflow executions.
- **Activity Worker**
  - An **activity worker** is a program that receives activity tasks, performs them, and provides results back.
  - Implement workers to perform tasks. These workers can run either on cloud infrastructure, or on your own premises.
  - Different activity workers can be written in different programming languages and run on different operating systems.
  - Assigning particular tasks to particular activity workers is called *task routing*. Task routing is optional.
- **Decider**
  - A software program that contains the coordination logic in a workflow.
  - It schedules activity tasks, provides input data to the activity workers, processes events that arrive while the workflow is in progress, and ultimately ends the workflow whe the objective has been completed.
  - Both activity workers and the decider receive their tasks by polling the SWF service.
- **Workflow Execution History**
  - The workflow execution history is composed of events, where an event represents a significant change in the state of the workflow execution.
  - SWF informs the decider of the state of the workflow by including, with each decision task, a copy of the current workflow execution history.
- **Polling**
  - Deciders and activity workers communicate with SWF using *long polling*.

Workflow Execution

1. Write activity workers that implement the processing steps in your workflow.
2. Write a decider to implement the coordination logic of your workflow.
3. Register your activities and workflow with Amazon SWF.
4. Start your activity workers and decider.
5. Start one or more executions of your workflow. Each execution runs independently and you can provide each with its own set of input data. When an execution is started, Amazon SWF schedules the initial decision task. In response, your decider begins generating decisions which initiate activity tasks. Execution continues until your decider makes a decision to close the execution.
6. Filter and view complete details of running as well as completed executions.

SWF provides service operations that are **accessible through HTTP requests.**

Endpoints

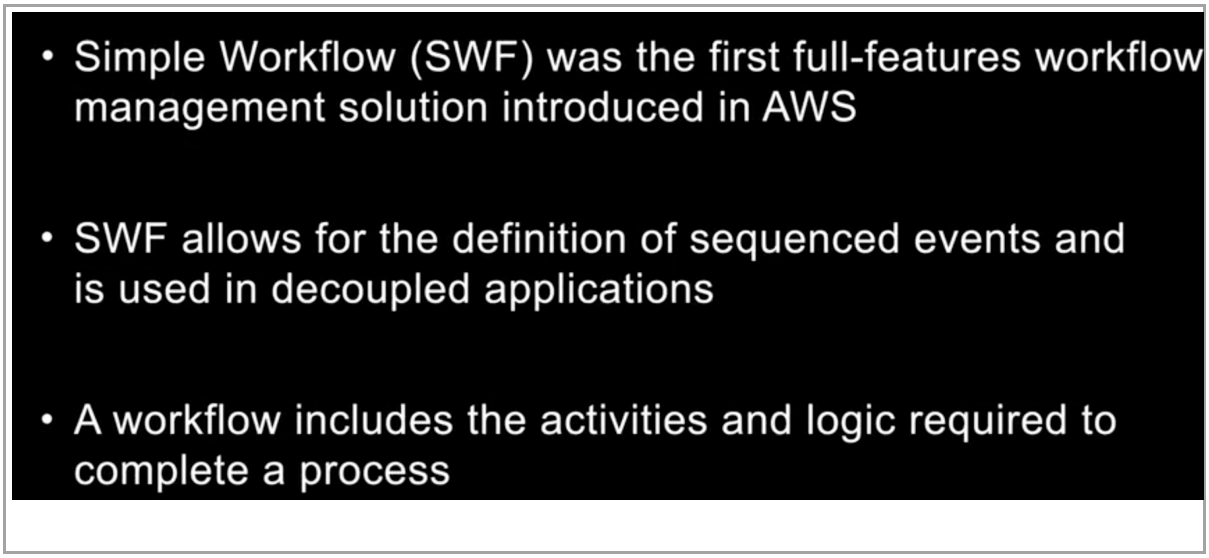
- To reduce latency and to store data in a location that meets your requirements, SWF provides endpoints in different regions.
- Each endpoint is completely independent. When you register an SWF domain, workflow or activity, it exists only within the region you registered it in.

Reference:

https://aws.amazon.com/swf/

Check out this Amazon SWF Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-simple-workflow-amazon-swf/



AWS - Step Functions

- **AWS Step Functions** provides serverless orchestration for modern applications.
- Orchestration centrally manages a workflow by breaking it into multiple steps, adding flow logic, and tracking the inputs and outputs between the steps.
- As your applications execute, Step Functions maintains application state, tracking exactly which workflow step your application is in, and stores an event log of data that is passed between application components.
- That means that if networks fail or components hang, your application can pick up right where it left off.

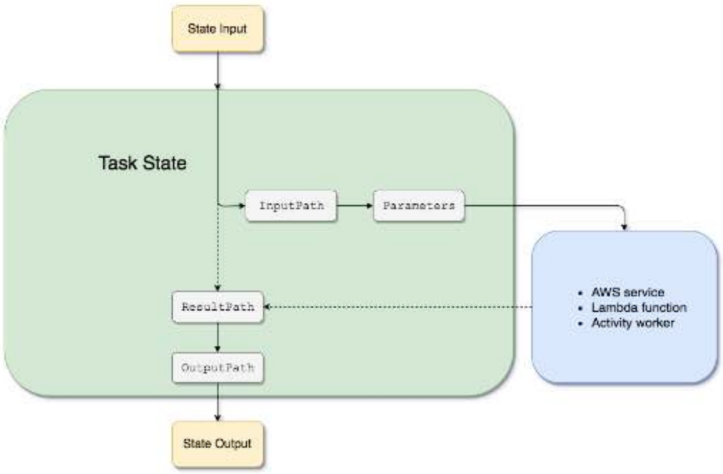
- Application development is faster and more intuitive with Step Functions, because you can define and manage the workflow of your application independently from its business logic.
- Making changes to one does not affect the other.
- You can easily update and modify workflows in one place, without having to struggle with managing, monitoring and maintaining multiple point-to-point integrations.
- Step Functions frees your functions and containers from excess code, so your applications are faster to write, more resilient, and easier to maintain.

- https://aws.amazon.com/step-functions/
- Replacements for SWF

Concepts

- Step Functions is based on the concepts of **tasks** and **state machines**.
  - A task performs work by using an activity or an AWS Lambda function, or by passing parameters to the API actions of other services.
  - A finite state machine can express an algorithm as a number of states, their relationships, and their input and output.
- You define state machines using the **JSON-based Amazon States Language**.
- A state is referred to by its *name*, which can be any string, but which must be unique within the scope of the entire state machine. An instance of a state exists until the end of its execution.
  - There are 6 types of states:
    - Task state – Do some work in your state machine. AWS Step Functions can invoke Lambda functions directly from a task state.
    - Choice state – Make a choice between branches of execution
    - Fail or Succeed state – Stop an execution with a failure or success
    - Pass state – Simply pass its input to its output or inject some fixed data
    - Wait state – Provide a delay for a certain amount of time or until a specified time/date
    - Parallel state – Begin parallel branches of execution

- **Activities** enable you to place a task in your state machine where the work is performed by an ***activity worker*** that can be hosted on Amazon EC2, Amazon ECS, or mobile devices.
- Activity tasks let you assign a specific step in your workflow to code running in an activity worker. Service tasks let you connect a step in your workflow to a supported AWS service.
- With **Transitions**, after executing a state, AWS Step Functions uses the value of the *Next* field to determine the next state to advance to. States can have multiple incoming transitions from other states.
- State machine data is represented by JSON text. It takes the following forms:
  - The initial input into a state machine
  - Data passed between states
  - The output from a state machine
- Individual states receive JSON as input and usually pass JSON as output to the next state.



- A **state machine execution** occurs when a state machine runs and performs its tasks. Each Step Functions state machine can have multiple simultaneous executions.
- **State machine updates** in AWS Step Functions are **eventually consistent**.
- By default, when a state reports an error, AWS Step Functions causes the execution to **fail entirely**.
  - Task and Parallel states can have a field named *Retry* and *Catch* to retry an execution or to have a fallback state.

Reference:

https://aws.amazon.com/step-functions/features/

Check out this AWS Step Functions Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-step-functions/

Amazon Simple Workflow (SWF) vs AWS Step Functions vs Amazon SQS:

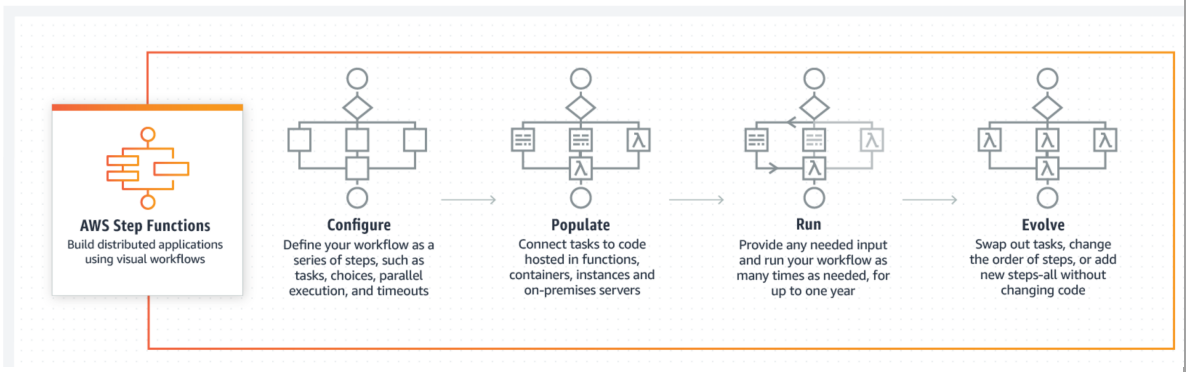
https://tutorialsdojo.com/aws-cheat-sheet-amazon-simple-workflow-swf-vs-aws-step-functions-vs-amazon-sqs/

Comparison of AWS Services Cheat Sheets:

https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/

- https://aws.amazon.com/getting-started/hands-on/create-a-serverless-workflow-step-functions-lambda/

How it works



- Step functions are positioned to replace SWF and AWS recommends using them instead of SWF workflows
- SWF utilizes state machines with a decider, activity tasks, and worker tasks
- A task is a single unit of work and choices provide branching logic
- Step functions support parallel processing