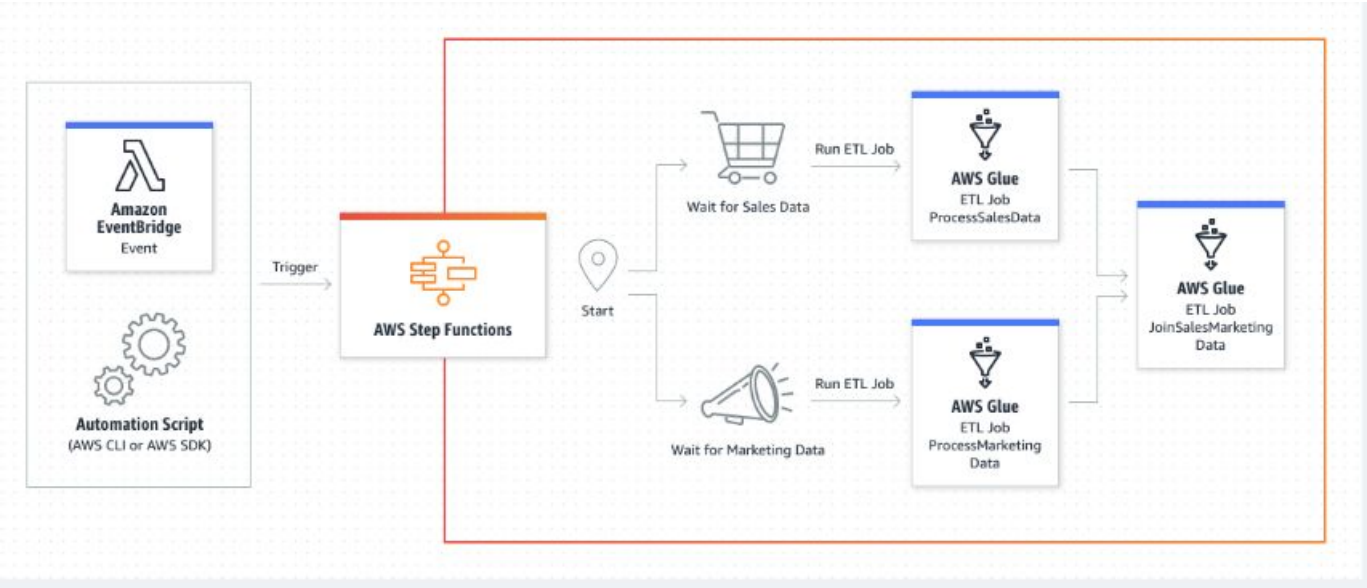

aws

AWS Step Function

Grow Data Skills

- **Overview**: AWS Step Functions is a serverless orchestration service that makes it easy to sequence AWS services into business-critical applications. Through visual workflows, you can design and run complex business processes, automations, and data pipelines.

- **How It Works**: Step Functions lets you coordinate multiple AWS services into serverless workflows so you can build and update apps quickly. You define your **workflows** as **state machines**, with each state representing a step in your application that can do work (Task states), decide which path to take (Choice states), stop with an error (Fail states), and more.



AWS Step Function

Grow Data Skills

- **Key Features:**
  - ***Visual Workflow Management:*** A graphical interface to visualize the components of your application as a series of steps.
  - ***Serverless Nature***: Automatically triggers and manages each step in your workflow, scaling with no need for provisioning infrastructure.
  - ***Reliability and Fault Tolerance:*** Built-in error handling, retry logic, and parallel execution paths ensure reliable application execution.
  - ***Integration with AWS Ecosystem:*** Seamlessly integrates with AWS services like Lambda, ECS, SNS, SQS, and DynamoDB, enabling comprehensive applications.

- **Components Of Step Function:**
  - ***State Machine:*** The core component that defines the workflow or orchestration logic in JSON format. It consists of states that represent the workflow steps, with each state performing a specific function within the orchestration.
  - ***States***: Building blocks of a state machine, each representing a single unit of work or a decision point. Types of states include:
    - *Task State*: Represents a single unit of work, such as invoking an AWS Lambda function, publishing to an SNS topic, or inserting an item into DynamoDB.
    - *Choice State*: Adds branching logic, allowing the workflow to choose different paths based on input.
    - *Wait State*: Delays the state machine from transitioning to the next state for a specified time.
    - *Parallel State*: Executes multiple branches of tasks in parallel, merging their results upon completion.
    - *Succeed/Fail States*: Indicate the successful or unsuccessful termination of the state machine.
    - *Pass State*: Passes its input to its output or injects fixed data, useful for parameter passing or static configuration.
    - *Map State*: Processes a list of input data by iterating through each item and applying a set of steps.

AWS Step Function

Grow Data Skills

- **Key Features:**
  - ***Visual Workflow Management:*** A graphical interface to visualize the components of your application as a series of steps.
  - ***Serverless Nature***: Automatically triggers and manages each step in your workflow, scaling with no need for provisioning infrastructure.
  - ***Reliability and Fault Tolerance:*** Built-in error handling, retry logic, and parallel execution paths ensure reliable application execution.
  - ***Integration with AWS Ecosystem:*** Seamlessly integrates with AWS services like Lambda, ECS, SNS, SQS, and DynamoDB, enabling comprehensive applications.

- **Components Of Step Function:**
  - ***State Machine:*** The core component that defines the workflow or orchestration logic in JSON format. It consists of states that represent the workflow steps, with each state performing a specific function within the orchestration.
  - ***States***: Building blocks of a state machine, each representing a single unit of work or a decision point. Types of states include:
    - *Task State*: Represents a single unit of work, such as invoking an AWS Lambda function, publishing to an SNS topic, or inserting an item into DynamoDB.
    - *Choice State*: Adds branching logic, allowing the workflow to choose different paths based on input.
    - *Wait State*: Delays the state machine from transitioning to the next state for a specified time.
    - *Parallel State*: Executes multiple branches of tasks in parallel, merging their results upon completion.
    - *Succeed/Fail States*: Indicate the successful or unsuccessful termination of the state machine.
    - *Pass State*: Passes its input to its output or injects fixed data, useful for parameter passing or static configuration.
    - *Map State*: Processes a list of input data by iterating through each item and applying a set of steps.