Defining Workflows to Orchestrate Functions

DISTRIBUTED COORDINATION · STEP FUNCTIONS · STATE MACHINES & ASL · STATE TYPES · STANDARD VS EXPRESS · API & QUOTAS

STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **What it is: AWS Step Functions** is a serverless **orchestration** service that combines Lambda functions and other AWS services to coordinate the components of distributed applications and microservices with **visual workflows**. It automatically triggers and tracks each step, **retries** when errors occur, and **logs the state of each step** so you can diagnose problems quickly.

02 **Workflow = state machine:** You define a workflow — a **state machine** — as a series of **states** and **transitions**, written in JSON with the **Amazon States Language (ASL)**. **Tasks do all the work**; every run starts with an input, and each state transforms that input and passes its output to the next state.

03 **How a run flows:** Execution begins at the top-level `StartAt` state (required). Each state declares its `Type` (required) and a `Next` field naming the state to run when it finishes — advancing is a **transition**. A run ends at a **terminal state**: a `Succeed`, a `Fail`, or any state with `End` set to `true` (or on a runtime error).

04 **Eight state types:** `Task` (a unit of work), `Choice` (branching), `Fail` and `Succeed` (stop with failure / success), `Pass` (forward or inject data, no work), `Wait` (delay), `Parallel` (concurrent branches), and `Map` (iterate steps over an array).

05 **Standard vs. Express: Standard** suits long-running, auditable work — up to **1 year**, **exactly-once**, billed **per state transition**. **Express** suits high-volume event processing — up to **5 minutes**, **at-least-once** (async) or **at-most-once** (sync), billed by the **number and duration of requests and memory** used. Both use the same ASL.

02 KEY TERMS

TERM	DEFINITION
AWS Step Functions	Serverless orchestration service that combines Lambda functions and other AWS services into visual workflows coordinating distributed applications and microservices
State machine (workflow)	An object with a set number of operating conditions where each state's output depends on the previous condition; defined in the Amazon States Language
State	An element of a state machine; a state can make decisions from its input, perform actions, and pass output to other states
Amazon States Language (ASL)	The JSON-based, structured language used to define state machines — the same for Standard and Express workflows
Task	The state type that does all the work — by running an activity, invoking a Lambda function, or calling another service's API
Transition	The move to the next state, determined by the current state's `Next` field after it finishes
Terminal state	A state that ends the run — a `Succeed`, a `Fail`, or any state with `End` set to `true`; a runtime error also stops the run
Activity	Program code (an activity worker) hosted outside Step Functions that waits for an operator to act or supply input, and polls Step Functions for tasks
Activity worker	Any application that can make an HTTP connection — on EC2, ECS, a mobile device, a Lambda function, anywhere — that polls for and completes an activity task
taskToken	A unique identifier Step Functions generates when a task is assigned to a worker; the worker returns it with `SendTaskSuccess` or `SendTaskFailure` to report the result
Choice Rule	An element of a Choice state's `Choices` array pairing a comparison (variable, comparison type, value) with a `Next` field naming the state to run on a match

TERM	DEFINITION
Workflow Studio	A low-code visual tool for building Step Functions state machines through a guided, interactive interface

03 THE EIGHT STATE TYPES

the core of every exam question

STATE	WHAT IT DOES
`Task`	Performs a single unit of work — via an activity, a Lambda function, or another service's API (`Resource` holds the ARN)
`Choice`	Adds branching logic; routes to a `Next` state based on a comparison, with an optional `Default`
`Fail`	Stops the run and marks it a failure; optional `Cause` and `Error` — no `Next`, no `End`
`Succeed`	Stops a state successfully; a terminal state — no `Next`, no `End`
`Pass`	Passes input straight to output without doing work; can inject fixed data (`Result`)
`Wait`	Delays continuation for a relative time (`Seconds`) or until an absolute timestamp
`Parallel`	Runs multiple `Branches` concurrently; waits for all to finish before its `Next`
`Map`	Dynamically iterates a set of steps over each entry of an array (`Iterator` required)

04 STANDARD VS. EXPRESS WORKFLOWS

same ASL, different runtime behavior

STANDARD

long-running, auditable

- Up to **1 year** per run · 2,000 invocations/s
- **Exactly-once** — repeats only if `Retry` is set in ASL
- Billed **per processed state transition**
- Run history via API up to **90 days**; supports activities

EXPRESS

high-volume event processing

- Up to **5 minutes** per run · 100,000 invocations/s
- **At-least-once** (async) / **at-most-once** (sync); may repeat
- Billed by **requests, their duration, and memory**
- Does **not** support Step Functions activities

05 HOW A RUN FLOWS

`01` Run starts at the `StartAt` state with an input → `02` The state runs its `Type` and transforms the input → `03` The `Next` field selects the following state — a **transition** → `04` Repeat until a terminal state: `Succeed`, `Fail`, or `End: true` → `05` The run returns a result (or stops on a runtime error)

06 STEP FUNCTIONS API OPERATIONS

objective: recall these

OPERATION	WHAT IT DOES
`Create`	Upload state machines defined in JSON; register activity workers
`StartExecution`	Start a run; return an ARN that identifies the run
`StopExecution`	Stop a run; return the date the state machine stopped
`List`	List all state machines, runs, and activities
`Describe`	Describe individual state machines, runs, and activities
`GetExecutionHistory`	Get a run's history as a list of events

07 EXAM TRAPS

where the knowledge check gets you

- × “Step Functions runs your code.” — It **orchestrates**; it does not execute your business logic itself. A `Task` state does the work by invoking a Lambda function, an activity worker, or another service's API.
- × “Succeed and Fail states still need a `Next` or `End` field.” — Both are **terminal** states: they have no `Next` field and do **not** require an `End` field.
- × “If one Parallel branch fails, only that branch stops.” — False. If any branch fails (unhandled error or a transition to `Fail`), the **entire Parallel state fails and all branches are stopped**.
- × “Standard and Express workflows are written in different languages.” — Both use the **same Amazon States Language**; only the runtime behavior, limits, and pricing differ.
- × “Express workflows are exactly-once.” — Exactly-once is **Standard**. Express is **at-least-once** (async) or **at-most-once** (sync), and an invocation **can run more than once**.
- × “Because invokers auto-retry, my function code can ignore duplicates.” — Retries mean the same event may arrive repeatedly, so your code must be **idempotent** — safe to process the same request several times, especially when writing to a database.
- × “A Standard workflow can run indefinitely.” — Runs are bounded: **Standard up to 1 year, Express up to 5 minutes**.
- × “Step Functions quotas are hard limits, identical for both workflow types.” — Quotas are **not hard** (they guard against a misconfigured state machine) and **may differ significantly between Standard and Express**.

08 SELF-QUIZ

answers are upside-down below

Q1 Which AWS service coordinates the components of distributed applications and microservices using visual workflows?

Q2 In Step Functions, what is another name for a workflow, and what are its individual elements called?

Q3 What JSON-based, structured language is used to define a state machine?

Q4 Which top-level field names the state where a state machine begins running?

Q5 Name the three ways a state can be a terminal state.

Q6 List the eight Step Functions state types.

Q7 Which state type does all the work, and by what three means can it perform it?

Q8 Which workflow type is billed per state transition, is exactly-once, and can run up to one year?

Q9 Which API action starts a state machine run and returns an ARN identifying that run?

Q10 Sample exam: a request is processed by comparing it to a limit and, if it exceeds the limit, routing it to a manager for approval. Which state type does this?

ANSWER KEY (rotate the page)

Q1 AWS Step Functions **Q2** a state machine; states **Q3** the Amazon States Language (ASL) **Q4** StartAt **Q5** a Succeed state, a Fail state, or any state with End set to true **Q6** Task, Choice, Fail, Succeed, Pass, Wait, Parallel, Map **Q7** Task — via an activity, a Lambda function, or by passing parameters to another service's API **Q8** Standard **Q9** StartExecution **Q10** Choice