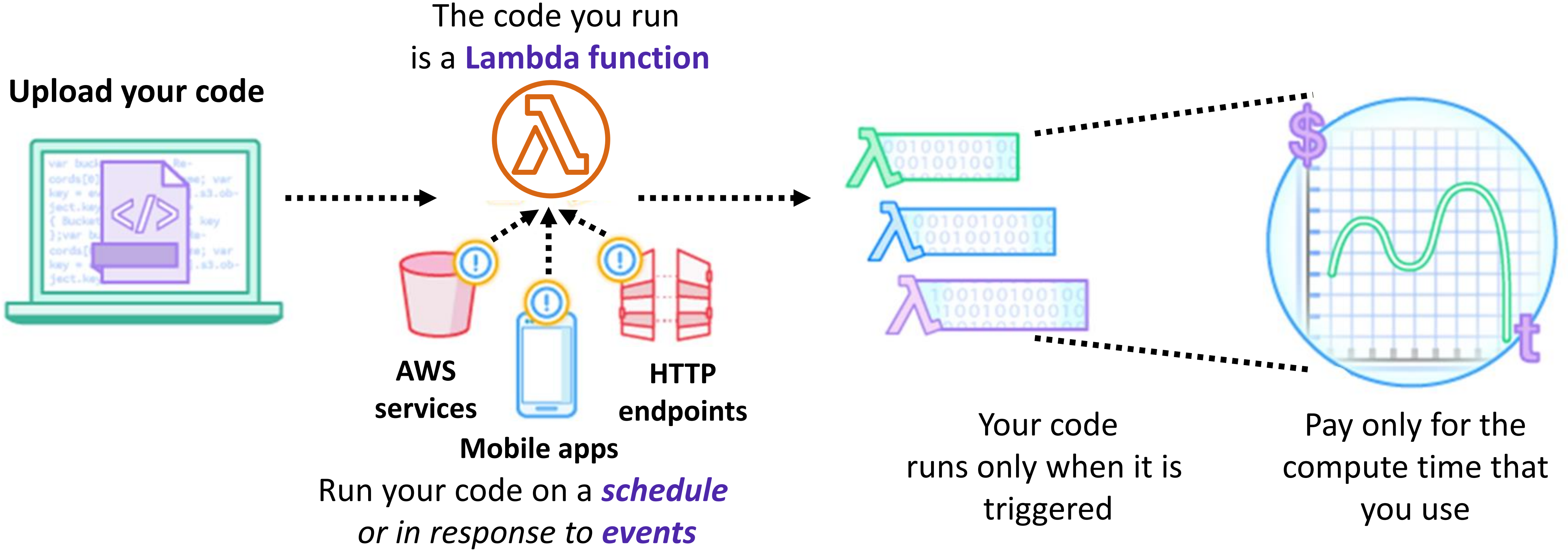
Serverless Computing

AWS Lambda: Run code without servers

AWS Lambda is a **serverless** compute service.



Upload your code
The code you run
is a **Lambda function**
AWS services
Mobile apps
HTTP endpoints
Run your code on a ***schedule***
or in response to ***events***
Your code runs only when it is triggered
Pay only for the compute time that you use

Benefits of Lambda

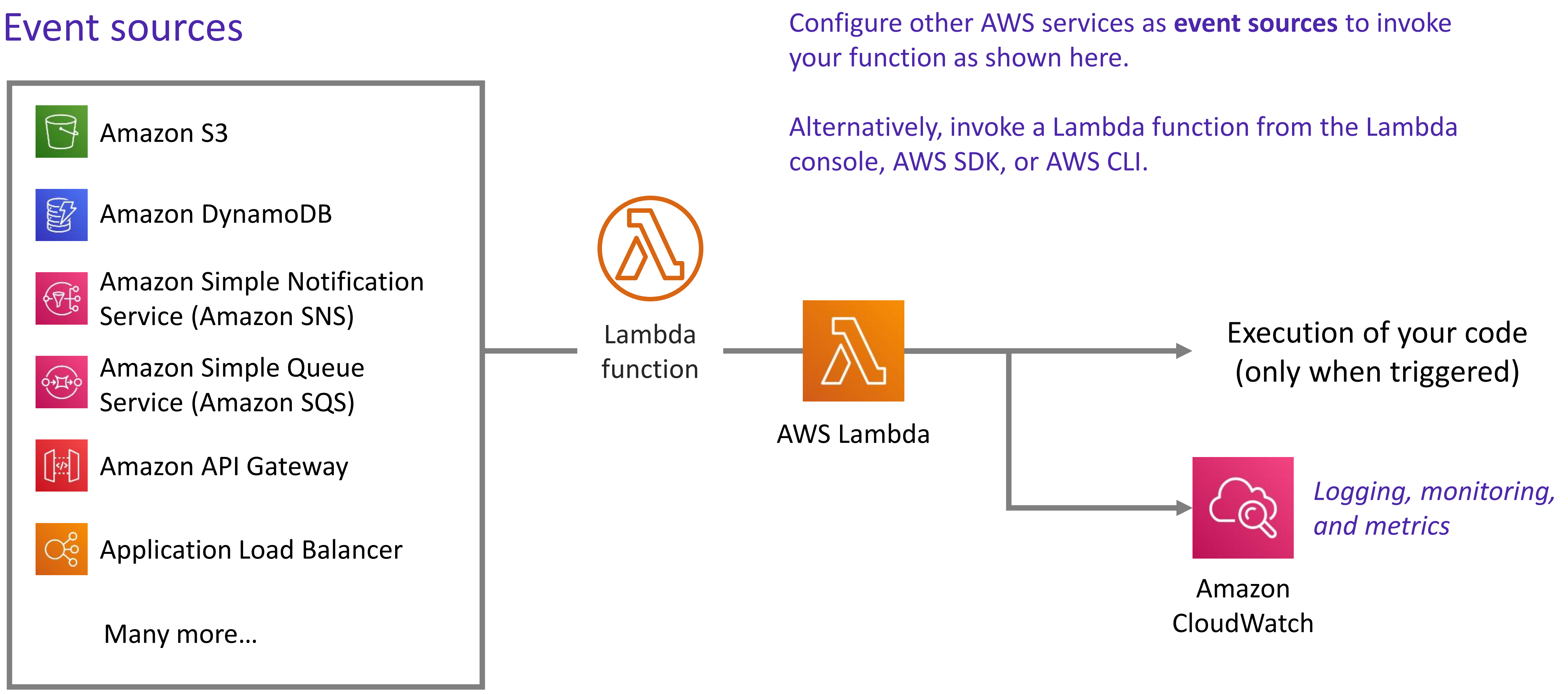
AWS Lambda

- It supports multiple programming languages
- Completely automated administration
- Built-in fault tolerance
- It supports the orchestration of multiple functions
- Pay-per-use pricing

AWS Lambda event sources

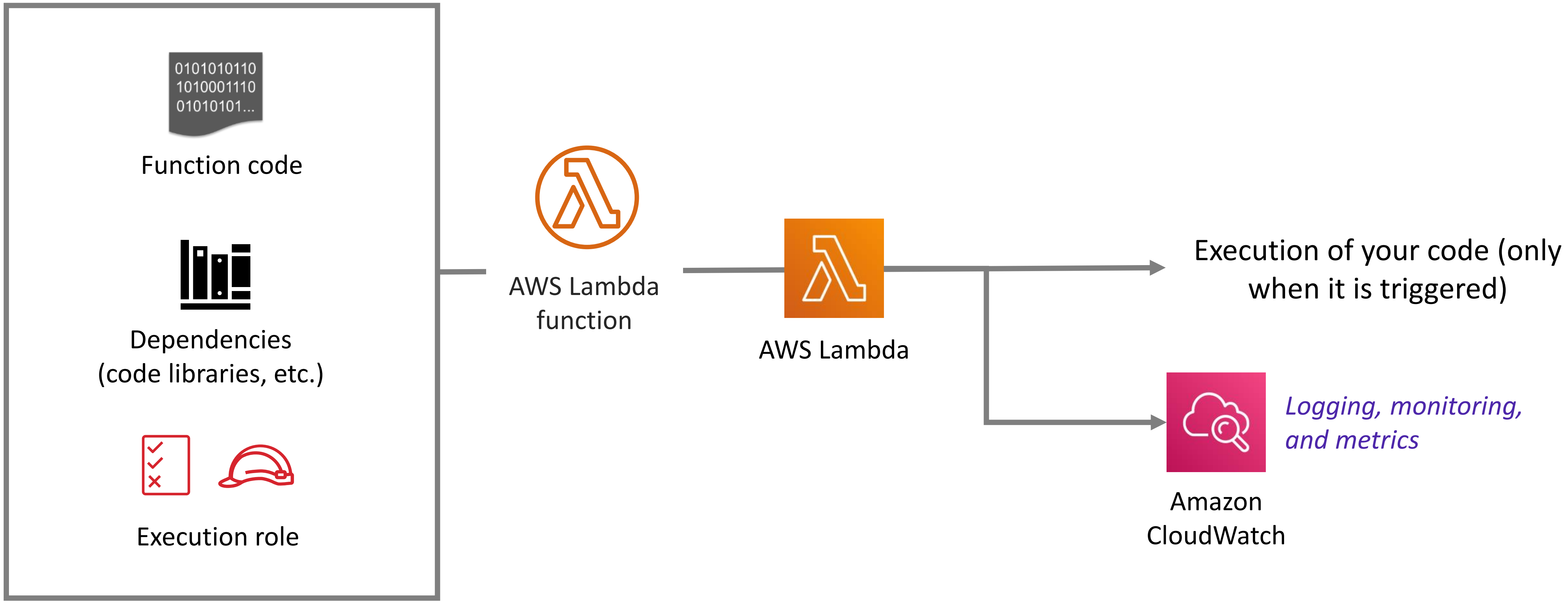
Configure other AWS services as **event sources** to invoke your function as shown here.

Alternatively, invoke a Lambda function from the Lambda console, AWS SDK, or AWS CLI.


Event sources
Amazon S3
Amazon DynamoDB
Amazon Simple Notification Service (Amazon SNS)
Amazon Simple Queue Service (Amazon SQS)
Amazon API Gateway
Application Load Balancer
Many more…
Lambda function
AWS Lambda
Execution of your code (only when triggered)
Logging, monitoring, and metrics
Amazon CloudWatch

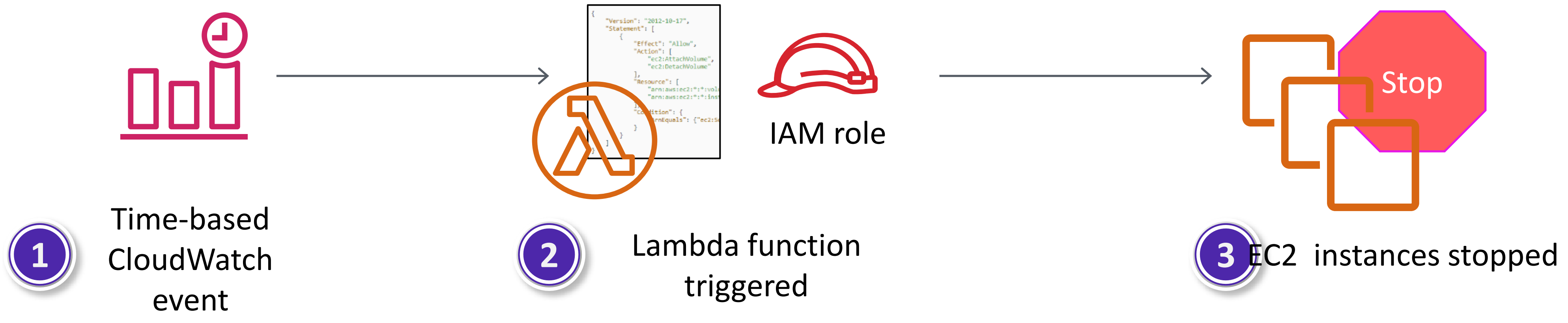
AWS Lambda function configuration

Lambda function configuration

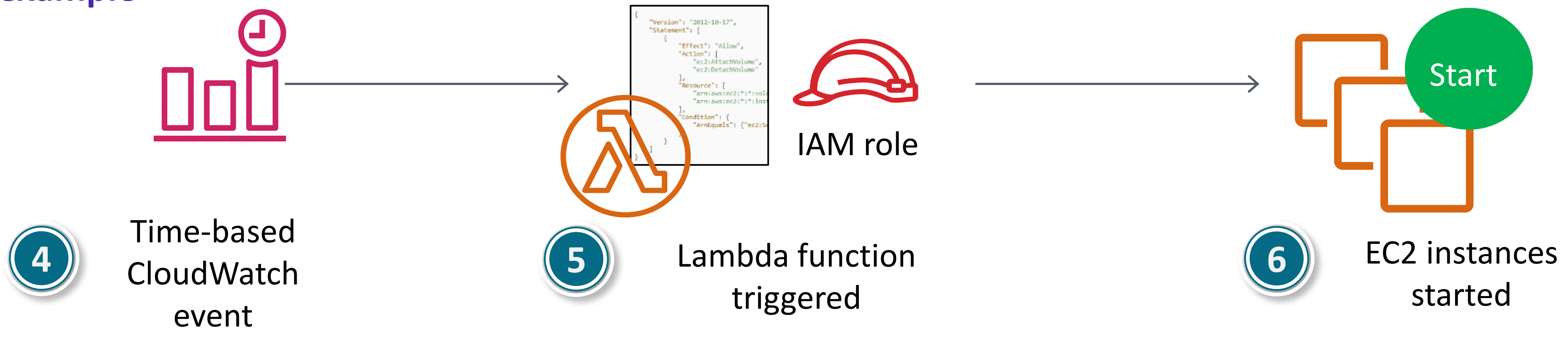


Schedule-based Lambda function example: Start and stop EC2 instances

Stop instances example

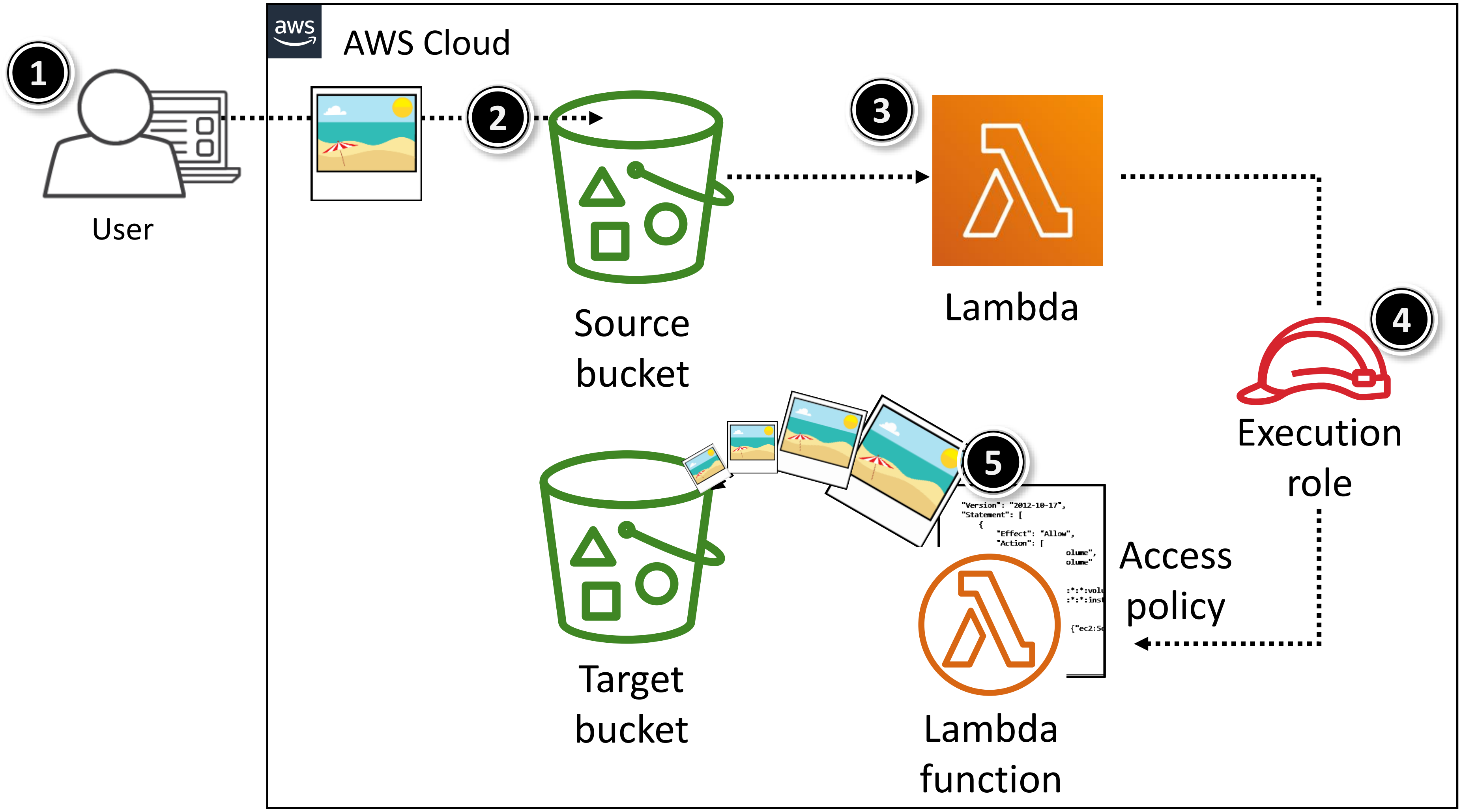


Start instances example



© 2019 Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Event-based Lambda function example: Create thumbnail images



AWS Lambda limits

Soft limits per Region:

- Concurrent executions = 1,000
- Function and layer storage = 75 GB

Hard limits for individual functions:

- Maximum function memory allocation = 3,008 MB
- Function timeout = 15 minutes
- Deployment package size = 250 MB unzipped, including layers

Additional limits also exist. Details are in the AWS Lambda Limits documentation.

© 2019 Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Function execution

All calls are limited to 15 minutes (900 seconds) of running time. The default timeout is three seconds but can be set to any value between one and 900 seconds.

Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 100 milliseconds.

Function scheduling

- AWS Lambda can execute your Lambda function on a regular schedule.

- You can specify:
  - A fixed rate (for example, execute the Lambda function every hour or 15 minutes).
  - A cron expression.

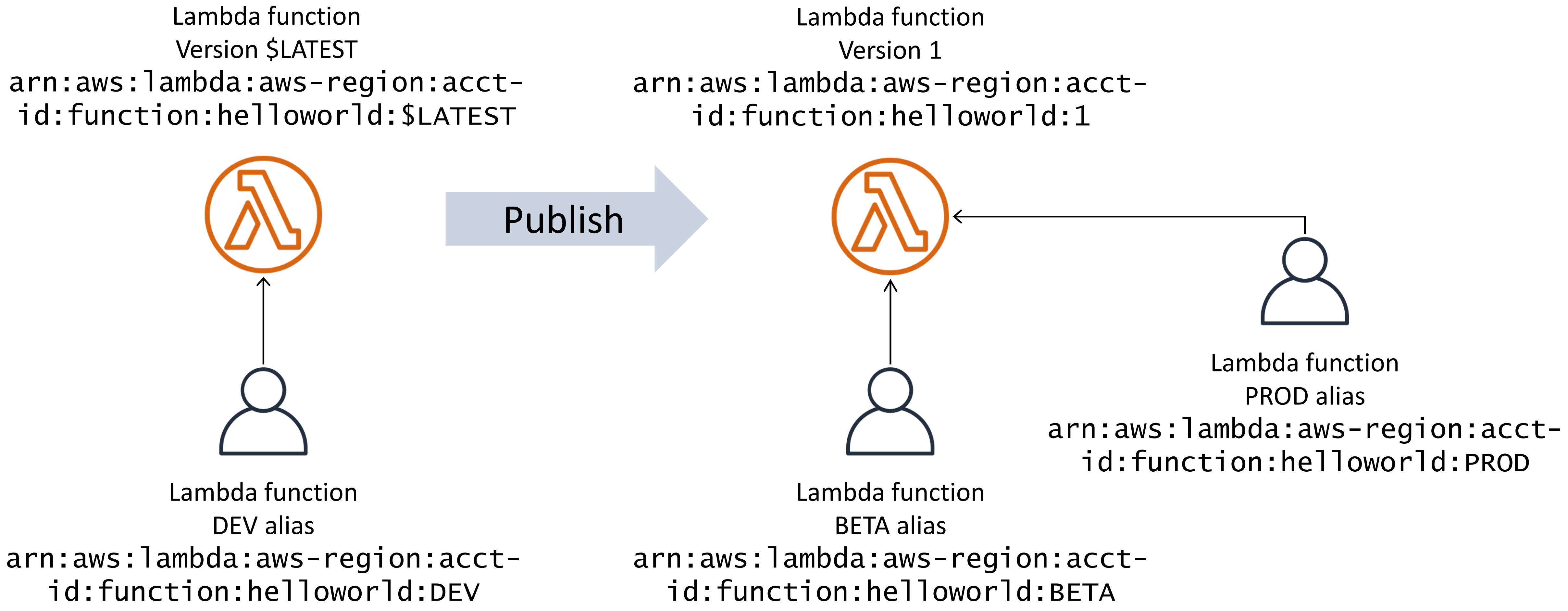
Restrictions

- Inbound network connections are blocked by AWS Lambda.
- Only TCP/IP sockets are supported.
- ptrace (debugging) system calls are restricted.
- TCP port 25 traffic is restricted as an anti-spam measure.

Versioning and aliases

- Versions are immutable copies of code plus configuration.
- Aliases are mutable pointers to versions.
- You can change the function code and settings only on the unpublished version of a function.
- Each version or alias gets its own Amazon Resource Name

Versioning and aliases: Example



Lambda function
Version $LATEST
```
arn:aws:lambda:aws-region:acct-id:function:helloworld:$LATEST
```

Lambda function
Version 1
```
arn:aws:lambda:aws-region:acct-id:function:helloworld:1
```

Lambda function
PROD alias
```
arn:aws:lambda:aws-region:acct-id:function:helloworld:PROD
```

Lambda function
DEV alias
```
arn:aws:lambda:aws-region:acct-id:function:helloworld:DEV
```

Lambda function
BETA alias
```
arn:aws:lambda:aws-region:acct-id:function:helloworld:BETA
```

Lambda layers

Centrally manage code and data that is shared across multiple functions.

- Reduce size of deployments
- Speed up deployments
- Limits:
  - 250 Mb unzipped package size
  - 5 layers

AWS Lambda

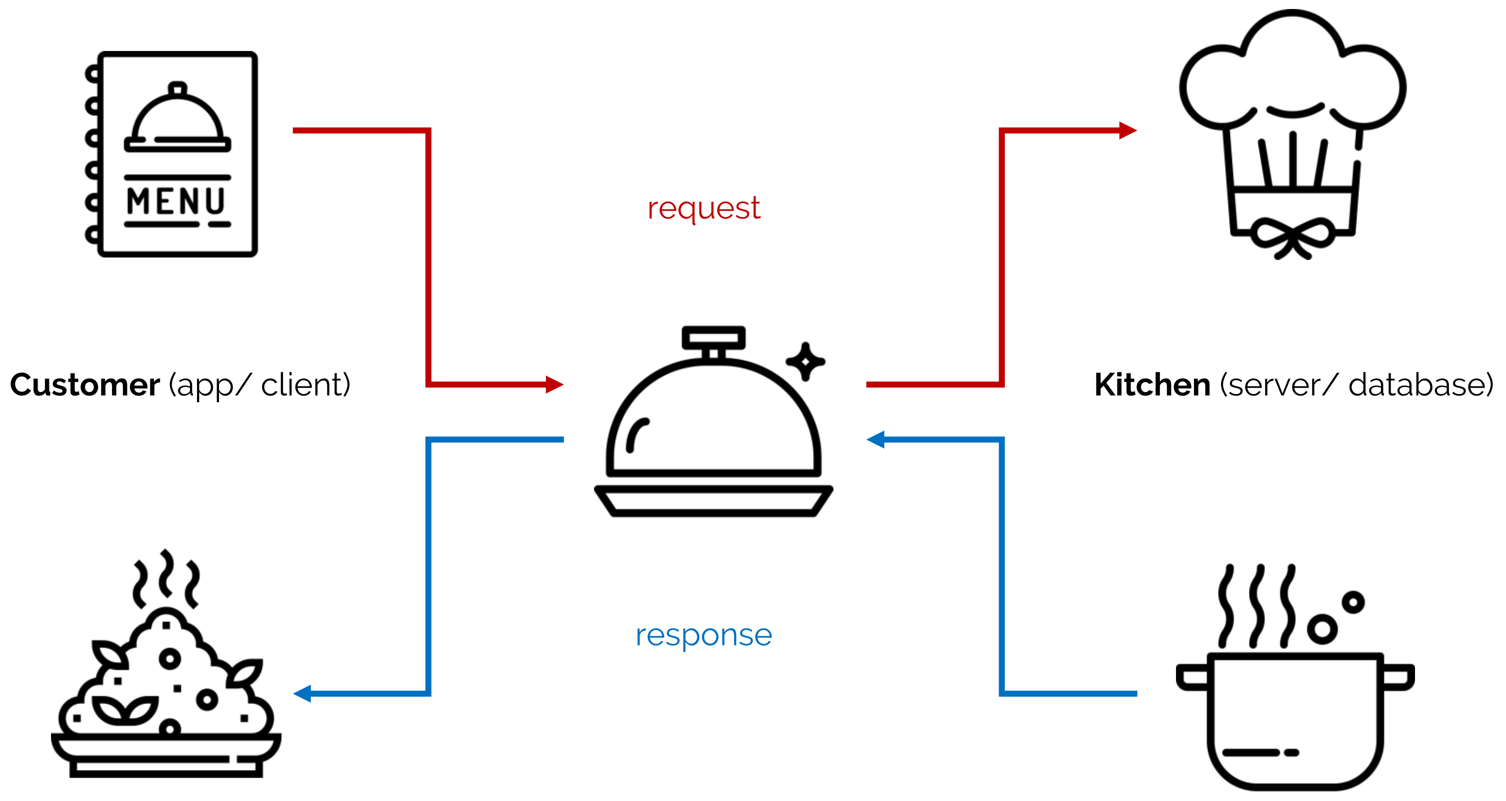
Lambda@Edge

- Run code closer to your users
- Run at edge locations
- Run on CloudFront events
    - Receive a request from viewer
    - Forwards a request origin
    - Receives response from origin
    - Returns response to viewer

Amazon API Gateway

What is an API?

Acts as a middle man between the applications

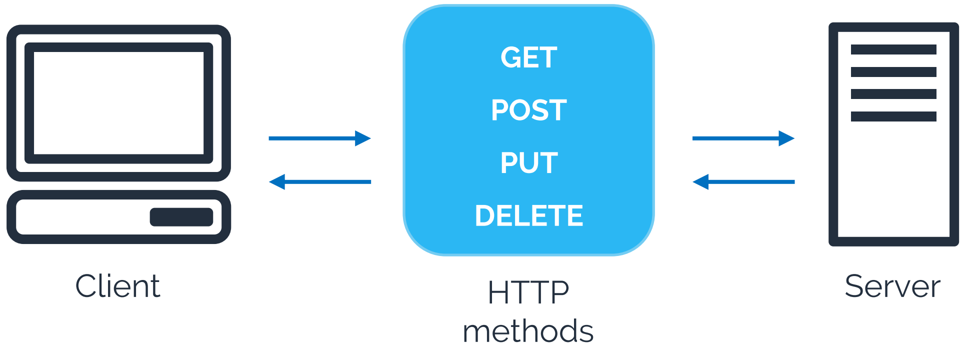


API Gateway

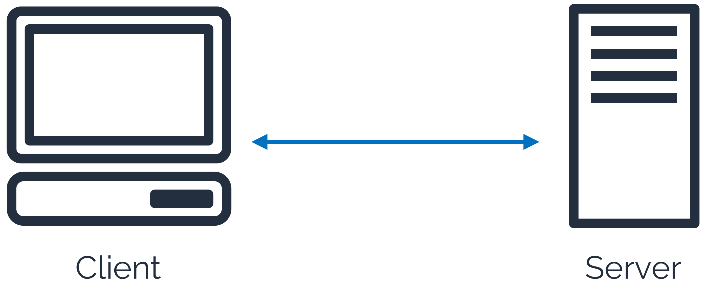
- Interface between frontend and backend
- APIs are set of instructions that defines how developers interface with an application
- Used with SDK to give partial access of the application to end users

Two Types of APIs

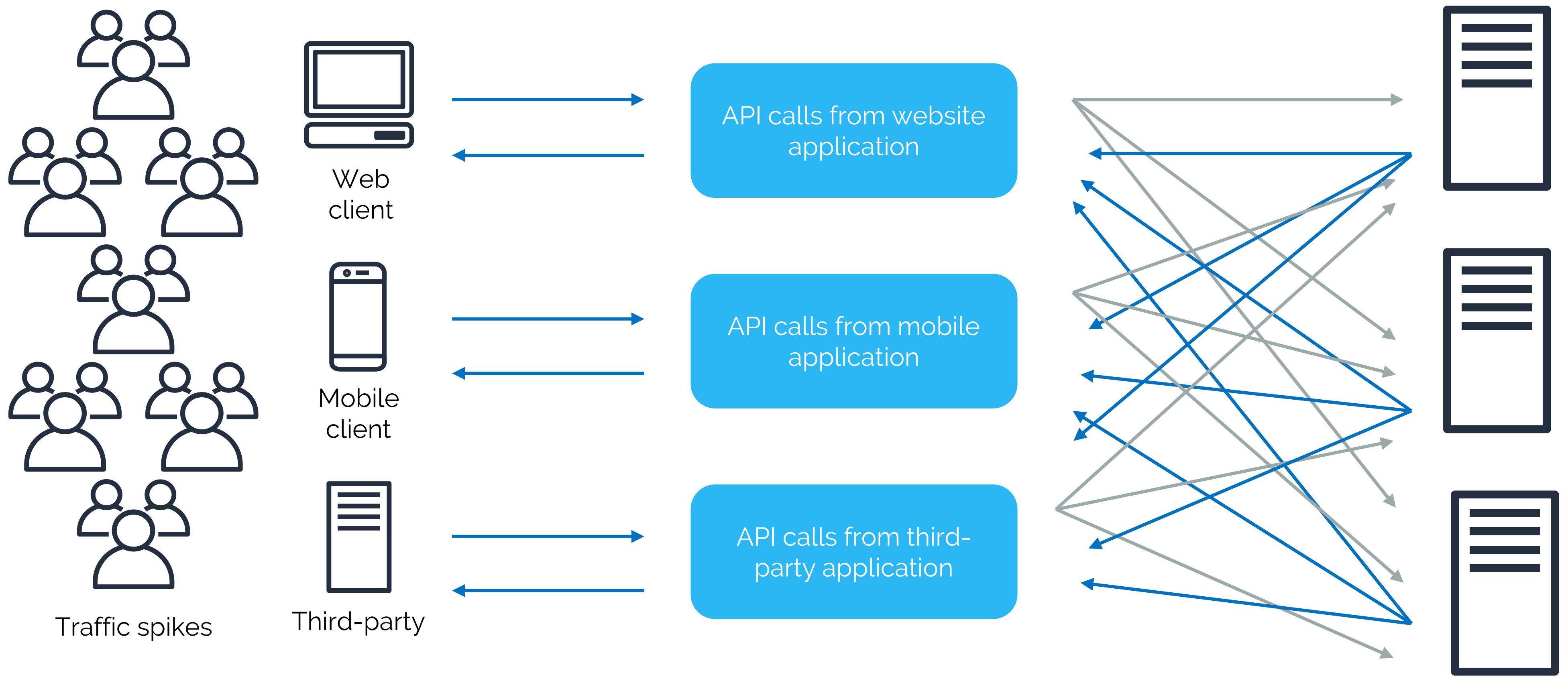
RESTful APIs
(request-response)



WebSocket APIs
(bidirectional)



Full Fledged Restful API


Web client
Mobile client
Third-party
Traffic spikes
API calls from website application
API calls from mobile application
API calls from third-party application

Amazon API Gateway

- Fully managed service
- Create RESTful and WebSocket APIs
- Can handle any number of concurrent API calls
- No minimum fees

Components

- Resource- Parts of your application
- Method- Way of Interaction (GET, POST, PUT, PATCH, DELETE)

  e.g. /pets could be the path of a *pets* resource. The *GET /pets* method would return a list of pets. Similarly, a user could add a pet by using the *POST /pets* method, which would send the information to the backend service

Endpoint Types

- Edge-optimized API endpoints
- Regional API endpoints
- Private API endpoints

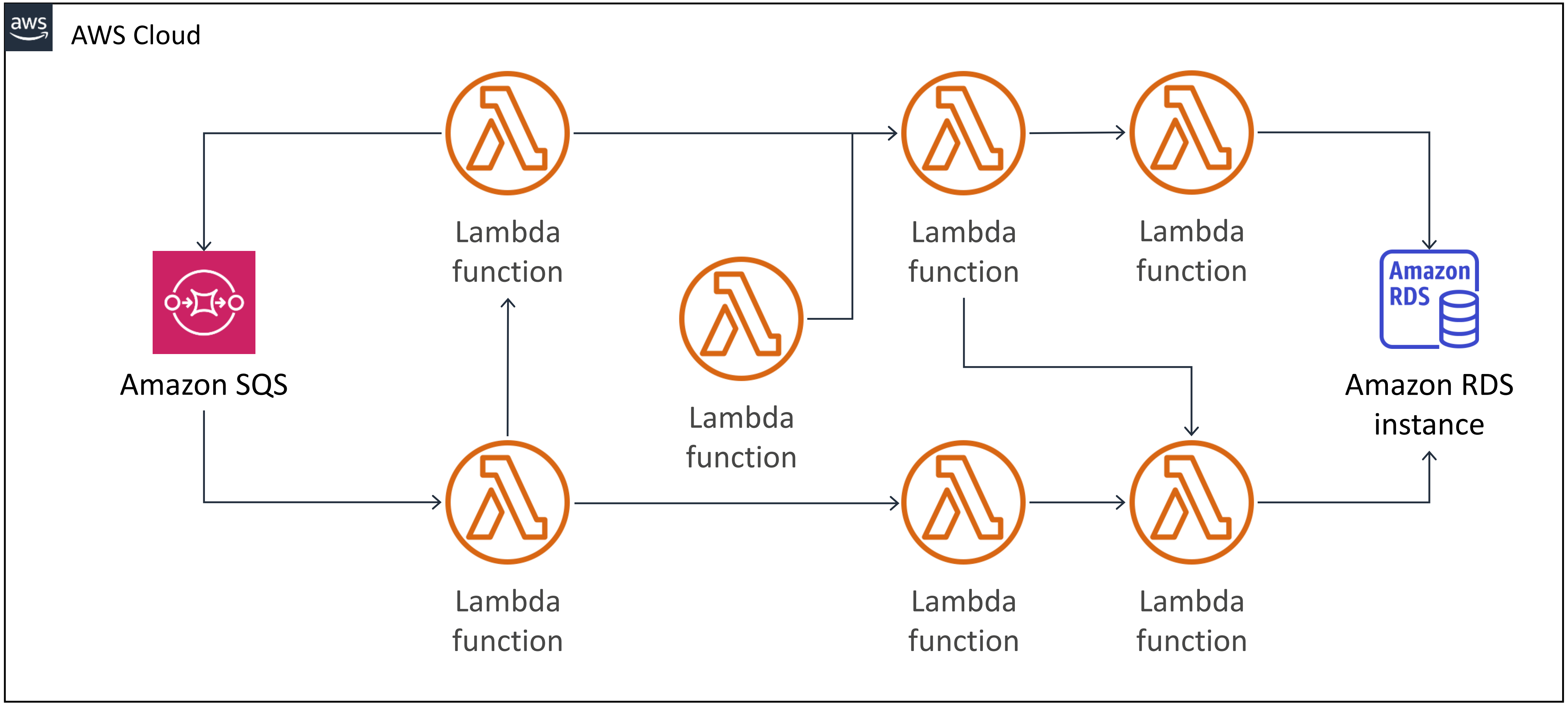
API Process

- After creating your API, you must deploy it to make it callable by your users
- Stages of APIs are created like versions
- Set up stage settings to enable
  - Caching
  - Throttling
  - Logging
  - Limits

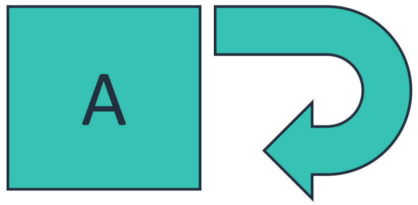
API Gateway Review

- Exposes only HTTPS endpoints
- Accept multiple payloads
- Communicate to multiple backends
  - EC2 Servers
  - Lambda functions
  - AWS Step Functions
  - HTTP Endpoints
- Endpoints are always public

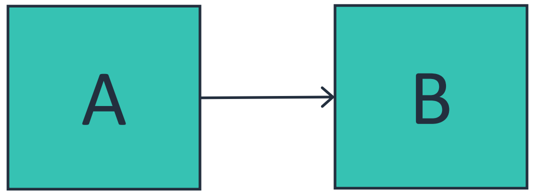
What problem do we need to solve?(cont)



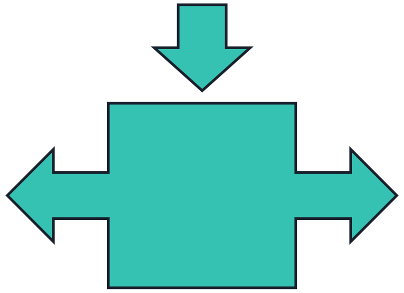
What do you want to do?



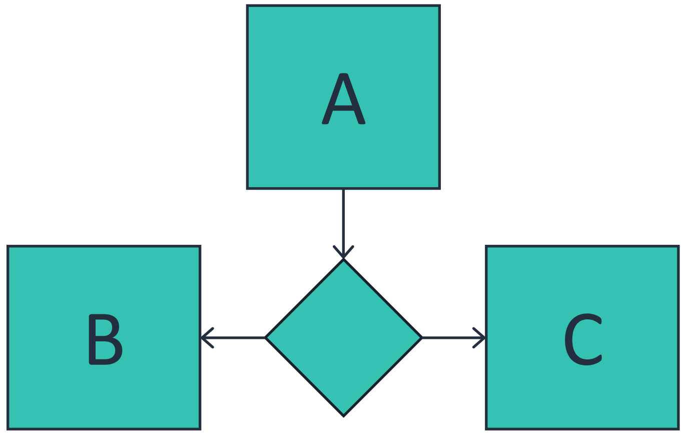
"I want to retry failed tasks."



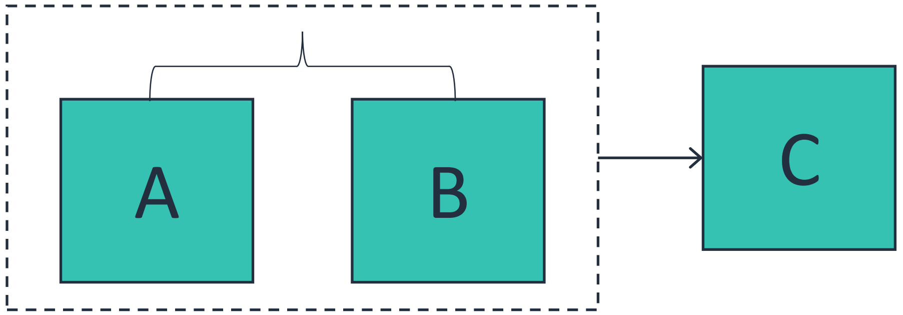
"I want to sequence tasks."



"I want to try/catch/finally."



"I want to select based on data."

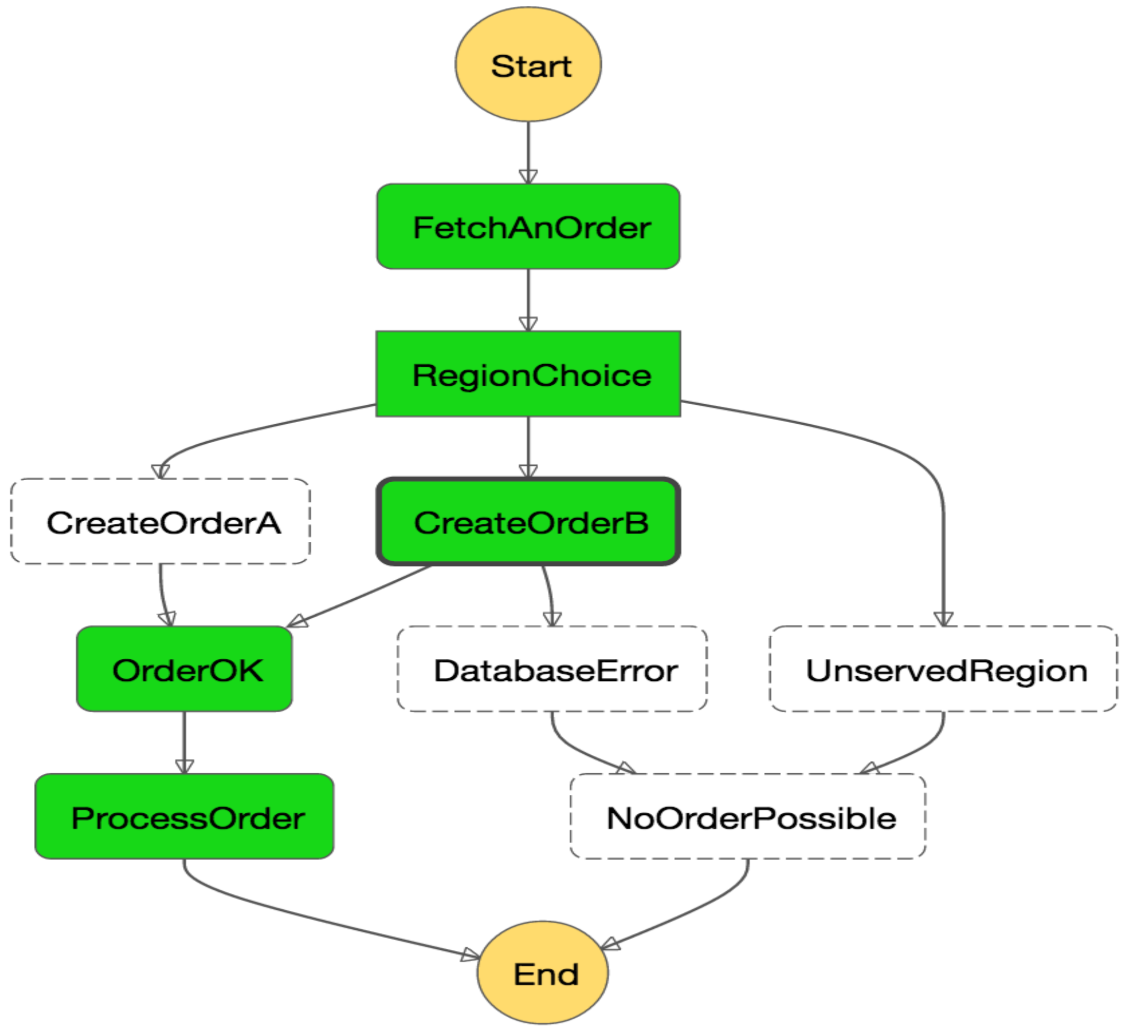


"I want to run tasks in parallel."

Introduction to AWS Step Functions

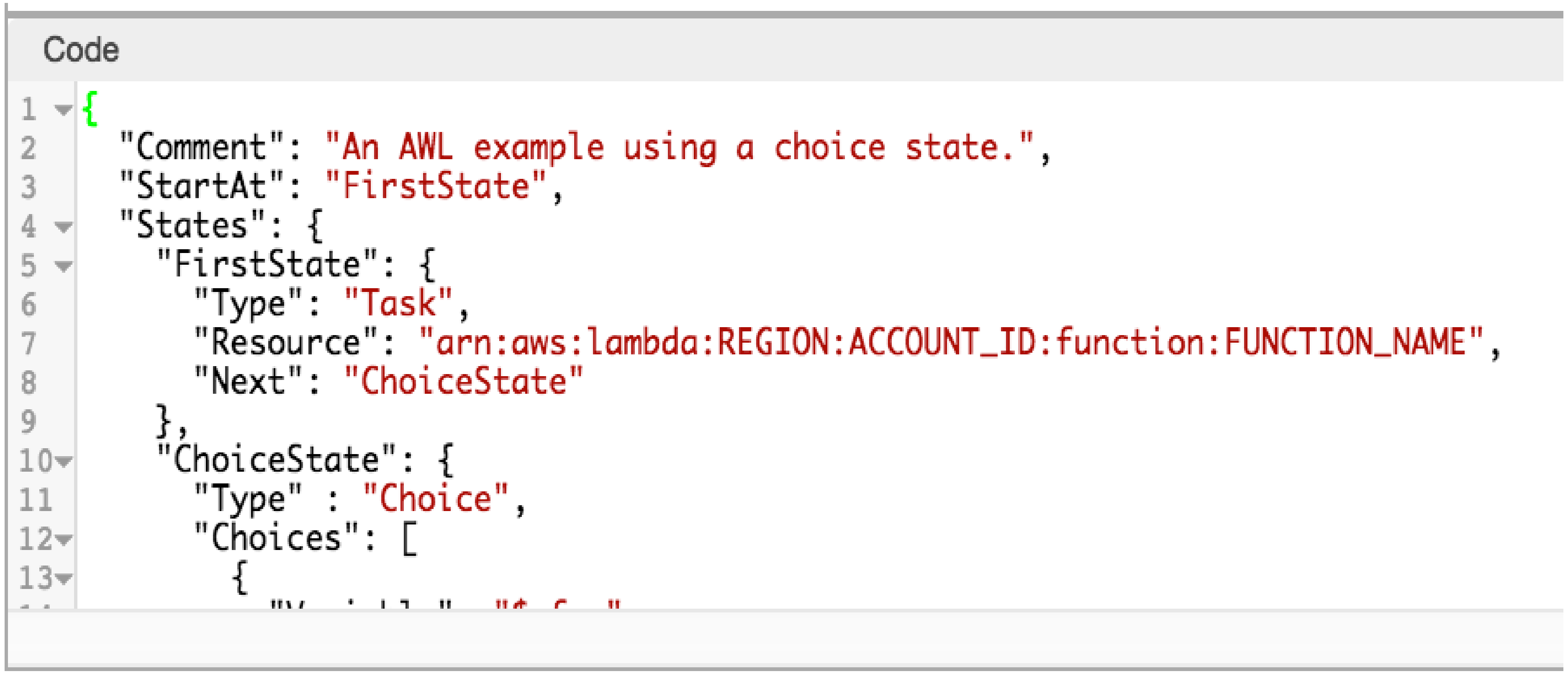
AWS Step Functions

- You define a workflow called a state machine made up of states.
- Each Order is an execution through this state machine.
- Each execution starts with an input and the states transform the input.
- Step Functions keep track of the state of each execution for up to one year.



Application lifecycle in AWS Step Functions

- Define in JSON

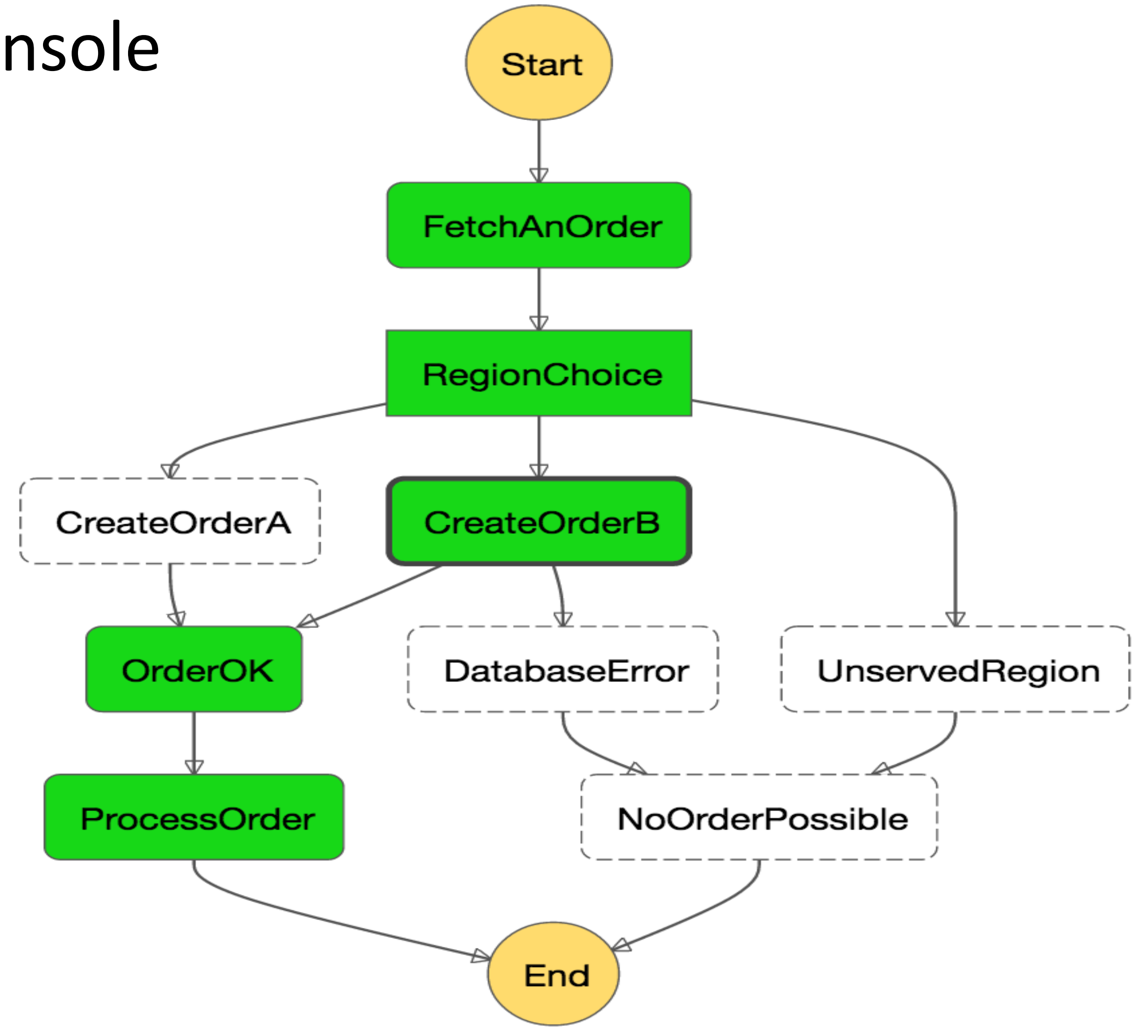


Code

```
{
  "Comment": "An AWL example using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type" : "Choice",
      "Choices": [
        {
```

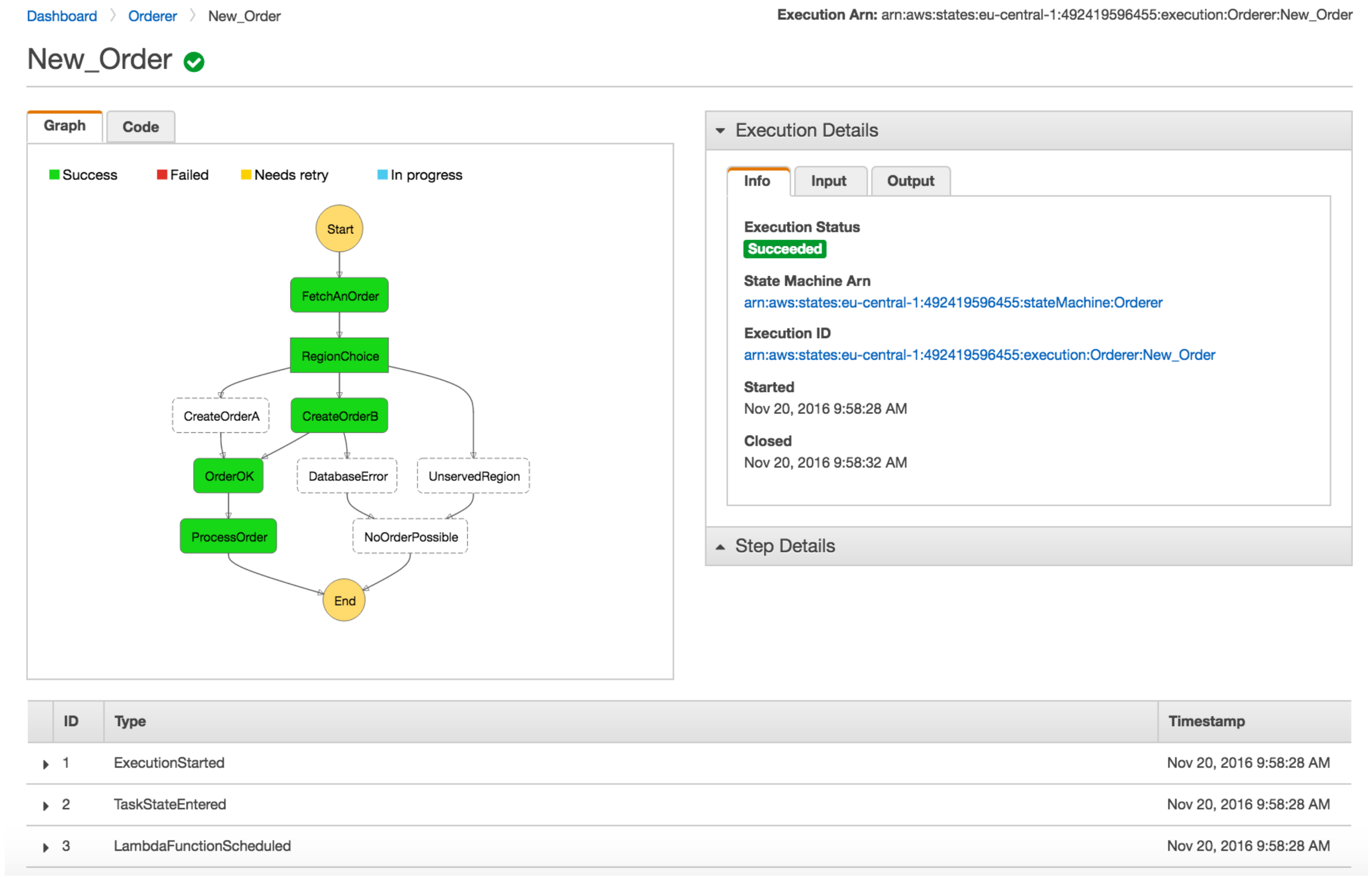
Application lifecycle in AWS Step Functions (cont)

- Visualize in the Console



Application lifecycle in AWS Step Functions (cont)

- Monitor Executions



Dashboard 〉 Orderer 〉 New_Order

Execution Arn: arn:aws:states:eu-central-1:492419596455:execution:Orderer:New_Order

New_Order ✓

Execution Status
Succeeded

State Machine Arn
arn:aws:states:eu-central-1:492419596455:stateMachine:Orderer

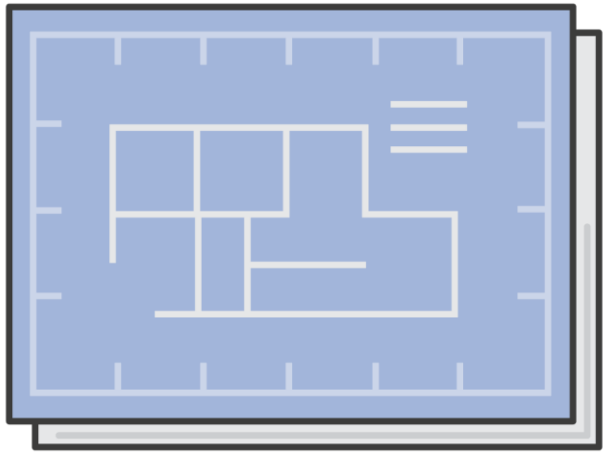
Execution ID
arn:aws:states:eu-central-1:492419596455:execution:Orderer:New_Order

Started
Nov 20, 2016 9:58:28 AM

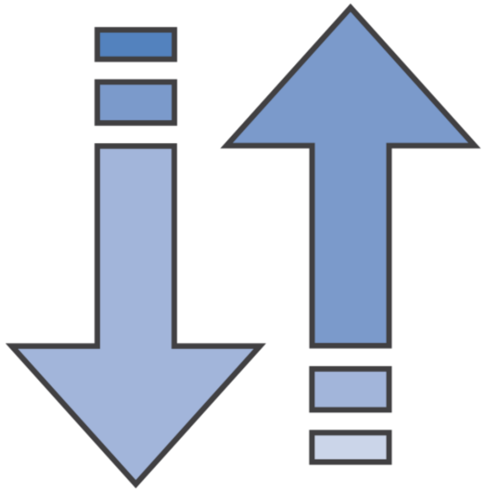
Closed
Nov 20, 2016 9:58:32 AM

ID	Type	Timestamp
1	ExecutionStarted	Nov 20, 2016 9:58:28 AM
2	TaskStateEntered	Nov 20, 2016 9:58:28 AM
3	LambdaFunctionScheduled	Nov 20, 2016 9:58:28 AM

Benefits of AWS Step Functions



Productivity
Build
applications quickly



Agility
Scale and
recover reliably

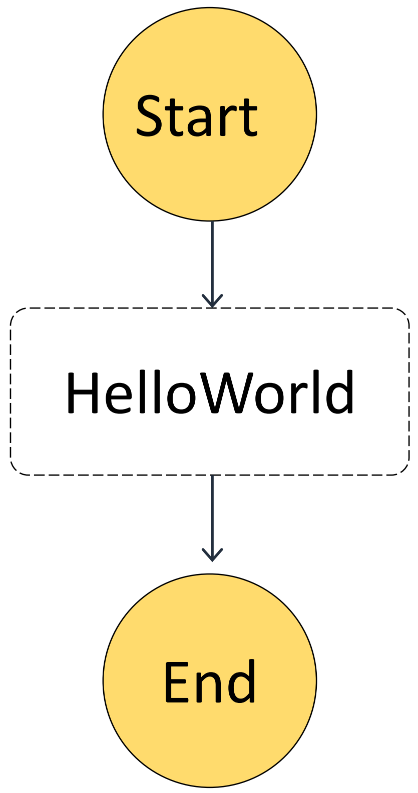
Adaptability
Evolve
applications easily

Terminology

Term	Meaning
State Machine	Workflow template
Execution	Specific workflow based on a template
Task	Lambda function or activity
Activity	Handle for external compute
Task Token	ID for instance of activity
Heartbeat	Ping from task indicating that it is still running
Failure	When execution fails
Success	When execution succeeds

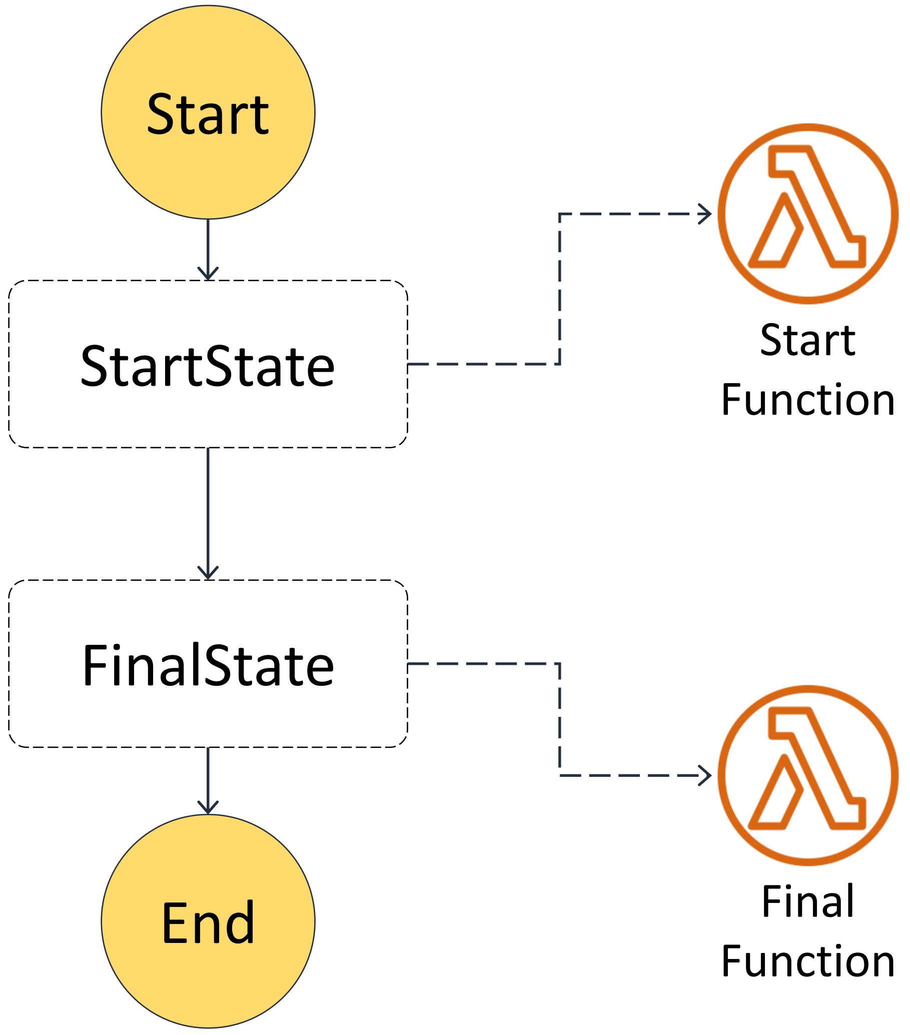
State machine

```
{
"Comment": "A Hello World Example",
"StartAt": "HelloWorld",
"States": {
      "HelloWorld": {
      "Type": "Task",
      "Result": “Hello World!",
      "End": true
  }
 }
}
```


$$\begin{tikzcd}[row sep=large]
\text{Start} \arrow[d] \\
\text{HelloWorld} \arrow[d] \\
\text{End}
\end{tikzcd}$$

Amazon States Language

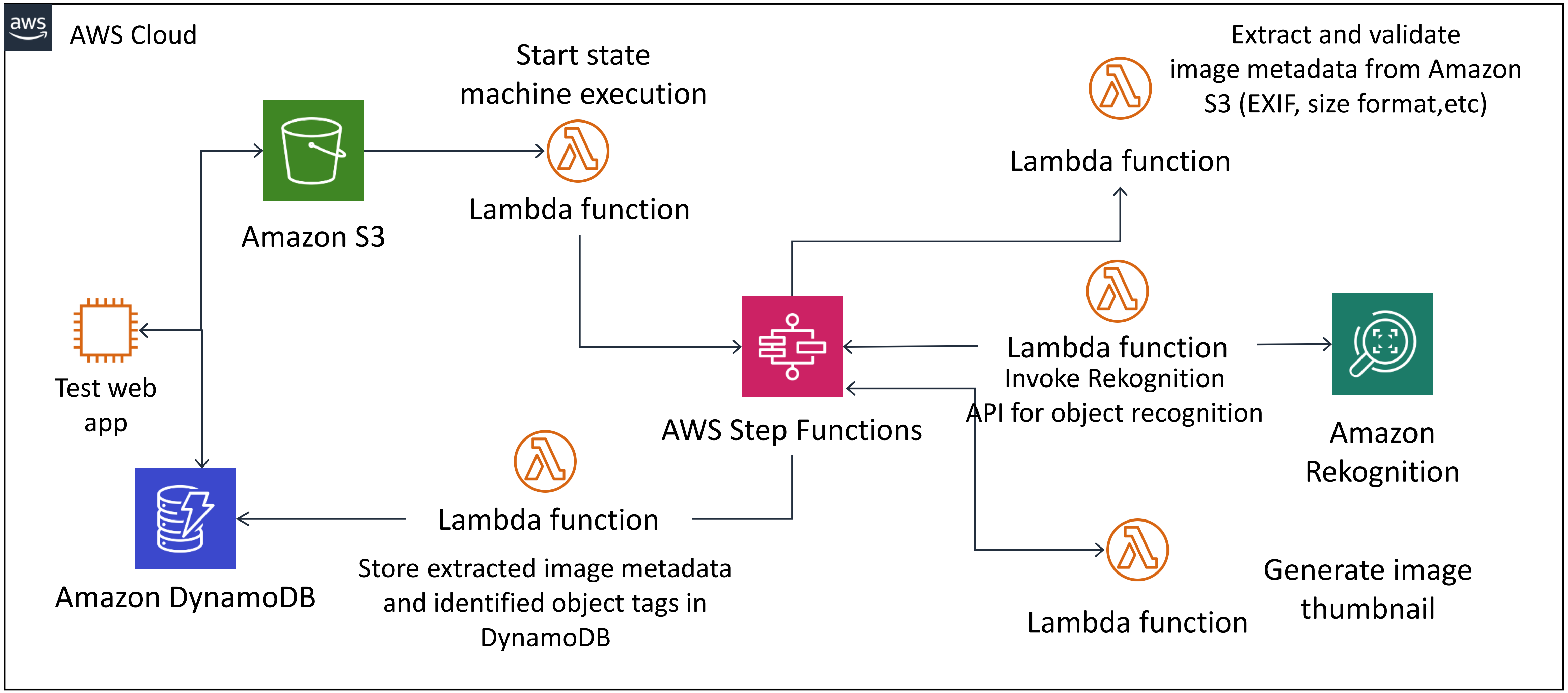
```
{
  "Comment": "An example of the Amazon States
          Language.",
  "StartAt": "StartState",
  "States": {
    "StartState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east…,
      "Next": "FinalState"
    }
    "FinalState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east…,
      "End": true
    }
  }
}
```



State types

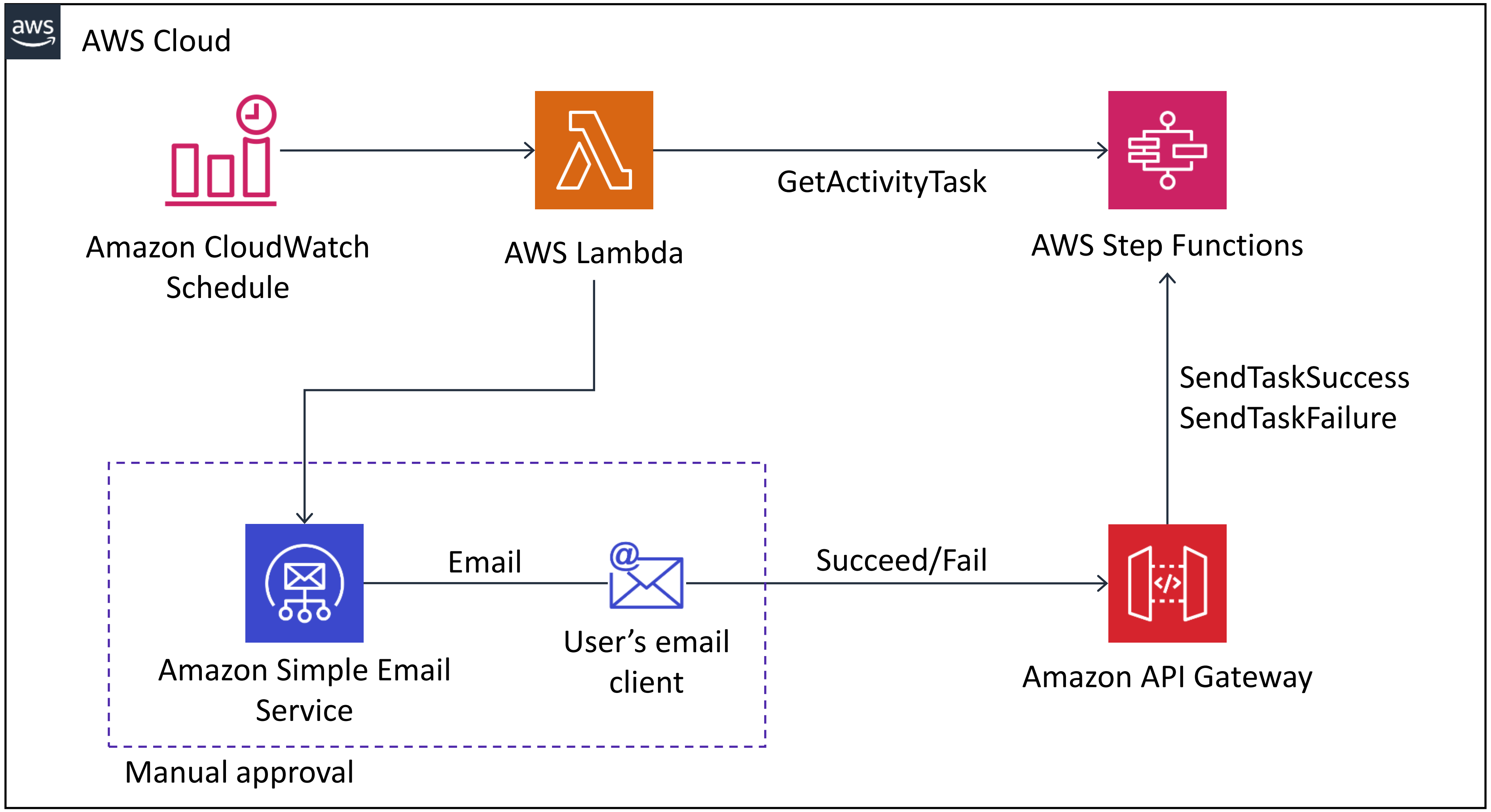
State	Definition
Task	A single unit of work
Choice	Adds branching logic
Parallel	Fork and join the data across tasks
Wait	Delay for a specified time
Fail	Stops an execution and marks it as a failure
Succeed	Stops an execution successfully
Pass	Passes its input to its output

Use Case: Image processing



AWS Cloud
Start state machine execution
Amazon S3
Lambda function
Extract and validate image metadata from Amazon S3 (EXIF, size format,etc)
Lambda function
Test web app
AWS Step Functions
Lambda function
Invoke Rekognition API for object recognition
Amazon Rekognition
Amazon DynamoDB
Lambda function
Store extracted image metadata and identified object tags in DynamoDB
Lambda function
Generate image thumbnail

Use Case: Activity for human interaction


aws
AWS Cloud
Amazon CloudWatch Schedule
AWS Lambda
GetActivityTask
AWS Step Functions
SendTaskSuccess
SendTaskFailure
Email
Amazon Simple Email Service
User’s email client
Succeed/Fail
Amazon API Gateway
Manual approval