Be Serverless

Who Am I?

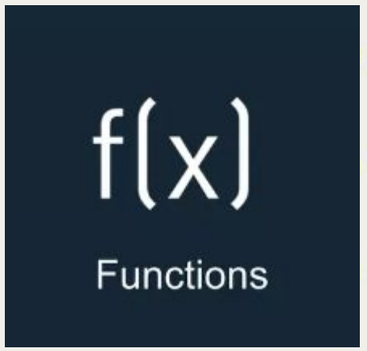
Mobeen Ahmed

CTO at Codup | ex VP of Engineering at Dastgyr | Advisor | Teacher | Mentor
Technology enthusiast | Entrepreneur | AWS Community Builder and Co Lead

Plan for today

- Introduction to serverless
- Introduction to AWS Lambda
- AWS Lambda Execution Models
- AWS Lambda Lifecycle
- AWS Step Functions

Introduction to Serverless



Introduction to Serverless

No servers to provision or manage

Scales with usage

Never pay for idle

Built-in availability and fault tolerance

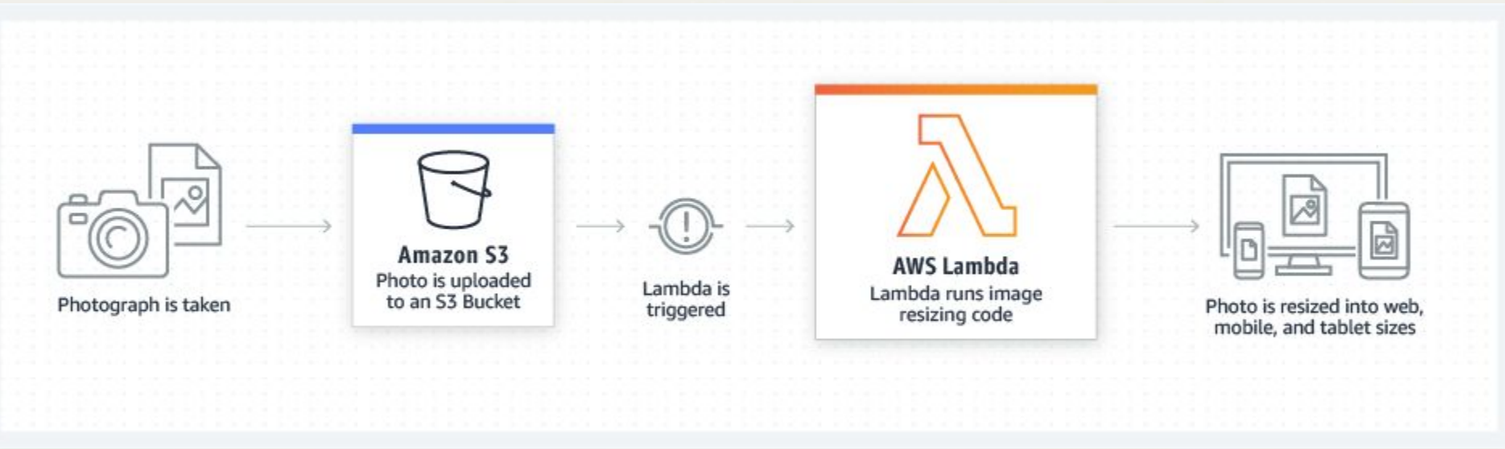
Introduction to Serverless

A serverless architecture is a way to build and run applications and services without having to manage infrastructure. Your application still runs on servers, but all the server management is done by AWS. You no longer have to provision, scale, and maintain servers to run your applications, databases, and storage systems.

another term used in place of “serverless” is “Functions as a Service”

Introduction to AWS Lambda

- Event Sourcing
  - Change in data state
  - Request to an endpoint
  - Change in source state
- Lambda Function
  - NodeJs, Python, C#, Go, Java
- Post event operation
  - Return response
  - Change data
  - Invoke another function



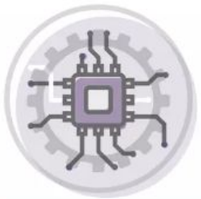
Common Use cases

Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT



Data Processing

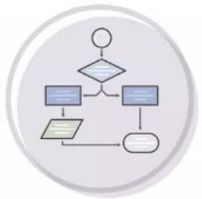
- Real time
- MapReduce
- Batch

Chatbots

- Powering chatbot logic

Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit



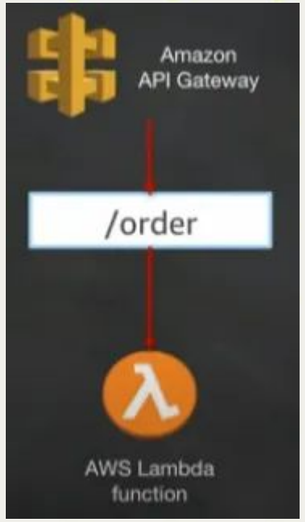
Autonomous IT

- Policy engines
- Extending AWS services
- Infrastructure management

aws

Lambda Execution Models - Push-Based Model

- API Gateway invokes a Lambda function.
- Lambda processes the request and share the response quite quick.
- It’s more like a process that expects Lambda to give the right response to the request source as fast as possible.
- Example
  - Assuming we have to perform an action like getting a piece of certain information from a service like S3 buckets or probably some other preferred service, this model explains going to that service and generates an invoke down to Lambda and take the message for any required action.



Lambda Execution Models - Event-Based Model

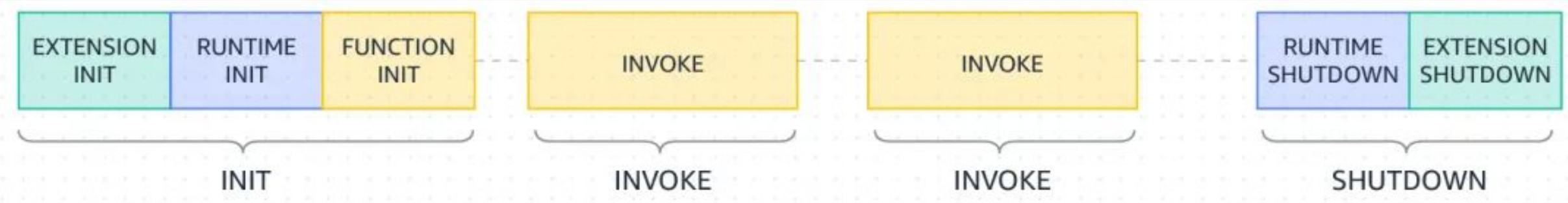
In this case, assuming we have to perform an action like getting a piece of certain information from a service like S3 buckets or probably some other preferred service, this model explains going to that service and generates an invoke down to Lambda and take the message for any required action without expecting a response — so it's more like a one-sided request.

Lambda Execution Models - Poll-Based Model

- Some certain AWS services such as Amazon DynamoDB, Kinesis and others use the Poll-based model when executing Lambda Functions. From such services, there can be different stream of data running around the system while requiring the AWS lambda service to pull out any desired information from there.
- This also known as stream based model

Lambda Execution Lifecycle

Lambda Execution Lifecycle



Lambda Execution Lifecycle - Init

- In the Init phase, Lambda performs three tasks:
  - Start all extensions (Extension init)
  - Bootstrap the runtime (Runtime init)
  - Run the function's static code (Function init)
  - Run any beforeCheckpoint runtime hooks (Lambda SnapStart only)
- The Init phase is limited to 10 seconds. If all three tasks do not complete within 10 seconds, Lambda retries the Init phase at the time of the first function invocation with the configured function timeout.

Lambda Execution Lifecycle - Invoke

- When a Lambda function is invoked in response to a Next API request, Lambda sends an Invoke event to the runtime and to each extension.
- **Cold Start**
  - When the service request to invoke a Lambda and a new sandbox environment is spawned.
- **Warm Start**
  - When invocation does not go through the Init phase and executes the logic directly.

- The billing of a Lambda invocation is based on the duration of the INVOKE phase only. The INIT phase is not billed unless provisioned concurrency is used

Lambda Execution Lifecycle - Shutdown

- When Lambda is about to shut down the runtime, it sends a Shutdown event to each registered external extension. Extensions can use this time for final cleanup tasks. The Shutdown event is a response to a Next API request.
- The entire Shutdown phase is capped at 2 seconds. If the runtime or any extension does not respond, Lambda terminates it via a signal (SIGKILL).

AWS Step Functions

AWS Step Functions

- AWS Step Functions is a serverless orchestration service that lets you integrate with AWS Lambda functions and other AWS services to build business-critical applications.
- Step Functions is based on state machines and tasks.
- In Step Functions, a workflow is called a state machine, which is a series of event-driven steps.
- A Task state represents a unit of work that another AWS service, such as AWS Lambda, performs.

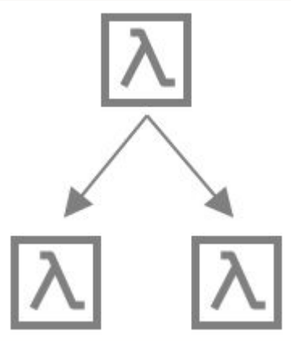
AWS Step Functions - Workflow Types

- **Standard workflows**
  - Standard Workflows are ideal for long-running (up to one year), durable, and auditable workflows. You can retrieve the full execution history using the Step Functions API for up to 90 days after your execution completes. Standard Workflows follow an exactly-once model, where your tasks and states are never run more than once, unless you have specified Retry behavior in ASL.
- **Express workflows**
  - Express Workflows are ideal for high-volume, event-processing workloads such as IoT data ingestion, streaming data processing and transformation, and mobile application backends. They can run for up to five minutes. Express Workflows employ an at-least-once model, where an execution could potentially run more than once.

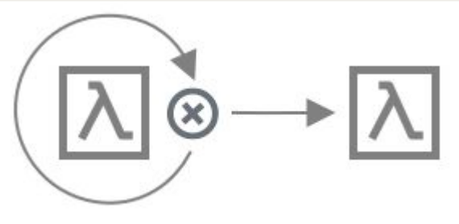
AWS Step Functions - Use Cases

- **Function orchestration**

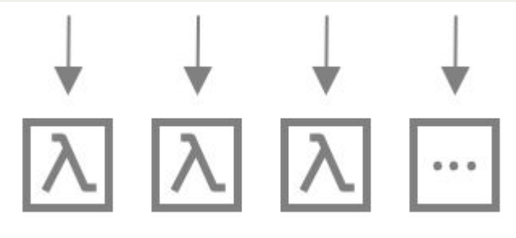
- **Branching**



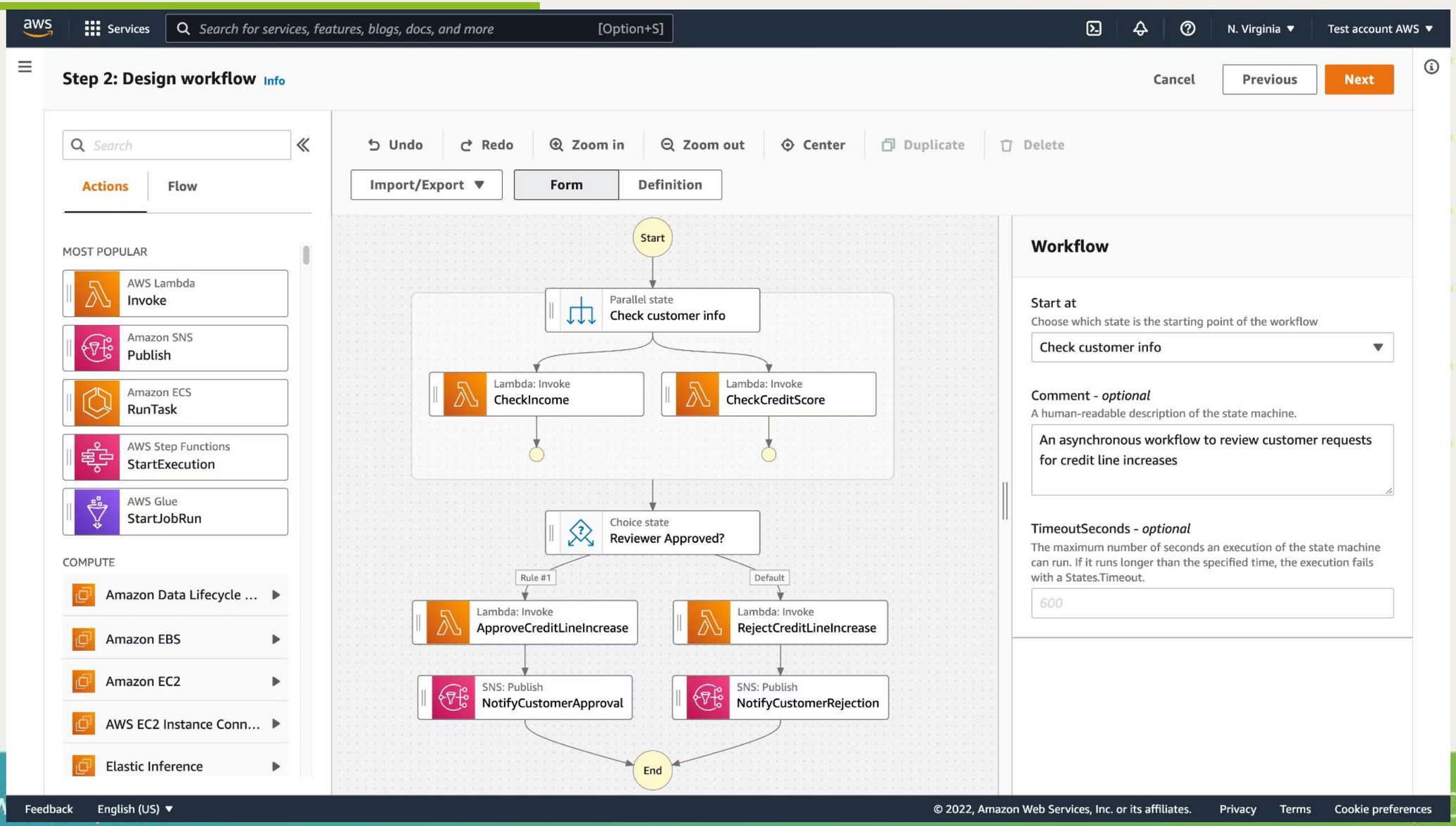
- **Error handling**



- **Parallel processing**



Workflow Designer AWS



AWS Step Functions - Benefits

- Simplified orchestration of microservices-based applications
- Improved application resilience
- Reduced need for integration code
- Separate workflow and business logic

AWS Step Functions - Challenges

- Application code that is harder to understand
- Proprietary language requirement - State machines can be defined only in **Amazon States Language**, so engineers need to learn this language to use Step Functions.
- AWS limits
- Vendor lock-in

Thank You

Contact Me

https://www.linkedin.com/in/mobeen-ahmed-dev/

mobeen@codup.co

www.codup.co