AWS Lambda Workshop

Matt Nappo

Agenda

- AWS Overview
- Traditional vs Serverless Paradigm
- AWS Lambda Overview
- Example Use Cases
- Demo
- Workshop

Please feel free to ask questions at any time!

AWS Overview

Pre Cloud Computing

- Say you’re a startup
- Running a web app
  - Backend services, database, auth servers, frontend
- You need to find and buy your own servers
  - Very expensive
- You need to manage that hardware yourself
  - Requires much expertise
- Hard to scale: If you grow, you need to set up even more infrastructure
  - Potentially around the globe
- Potentially not maximizing server usage

What is Amazon Web Services (AWS)?

- Cloud Computing Platform
- Customers can “rent” servers
- Many products for
    - SageMaker: AI training and deployment
    - RDS: Database hosting
    - SES: Email and notification systems
    - S3: Object Storage
    - Virtual clouds, advanced networking, DNS, blockchain apps, etc…
- Everything is very interoperable
- Main idea: Only pay for what you use

What is Amazon Web Services (AWS)?

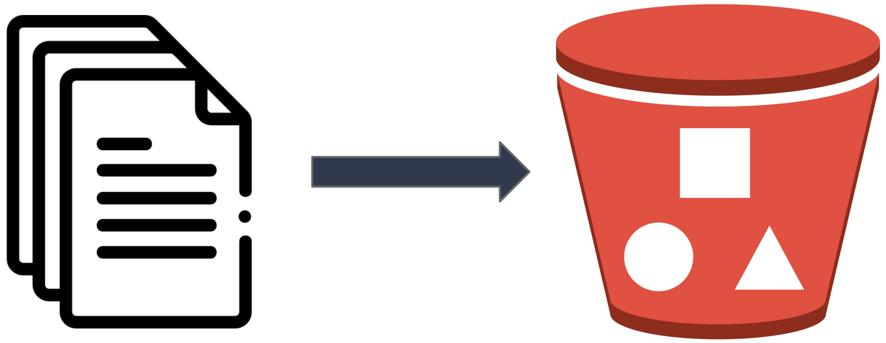
- ~41% market share in cloud computing
- Used by
  - Netflix: $19 million / month
  - Twitch: $15 million / month
  - LinkedIn: $13 million / month
  - Facebook: $11 million / month
- High chance you've interacted with AWS in the last week

AWS Elastic Compute Cloud (EC2)

- EC2 instances are just regular servers that you can run whatever you want on
- Most basic and most commonly used AWS product
- They come in many sizes
- You choose the operating system
- Pay per hour
  - t2.small: 1 CPU core, 2 GB RAM: $0.023 / hour
  - t4g.medium: 2 CPU cores, 4 GB RAM: $0.0336 / hour
  - x2iedn.32xlarge: 128 cores 4 TB RAM: $26.676 / hour

AWS Simple Storage Service (S3)

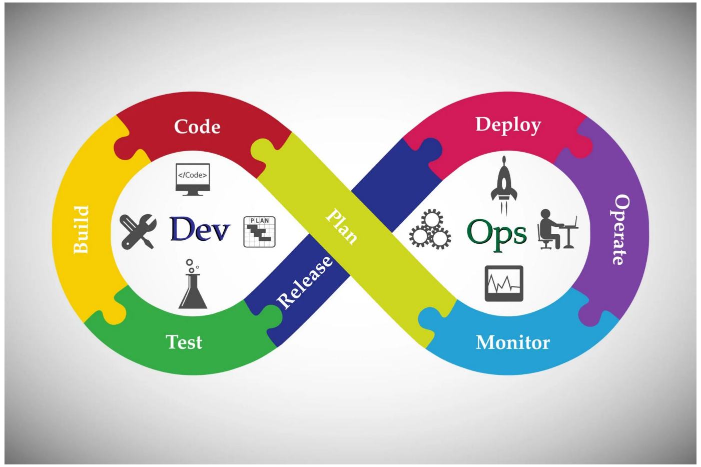
- Used for storing arbitrary files
  - Typically, files are written / uploaded once, and then read many times
- Objects (files) live in “Buckets”
- Very cheap storage (pay per object)
  - $0.005 per 1000 requests
  - $0.023 per GB stored



Traditional vs Serverless

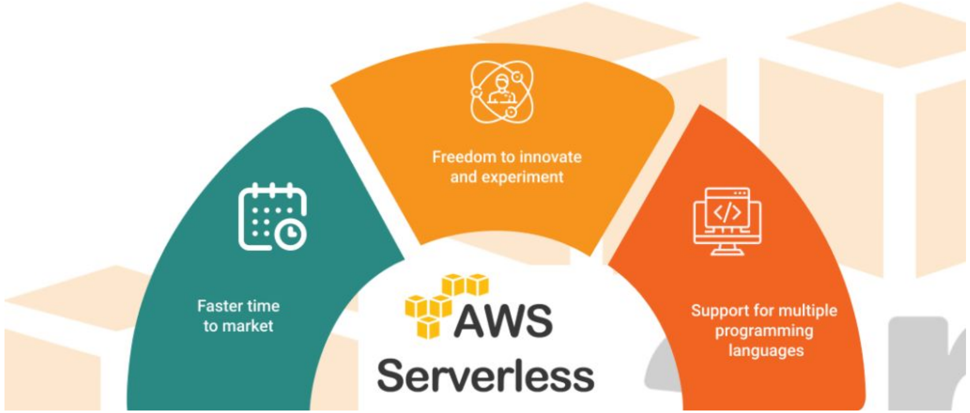
Traditional Paradigm

- We want to deploy some code
- Have to get a server
- Have to choose the right hardware
- Have to pick OS, install stuff, setup the firewall
- Have to keep the system up to date (security)
- Cons:
  - Unused hardware goes to waste
  - Surge of requests → server crashes



Serverless Paradigm

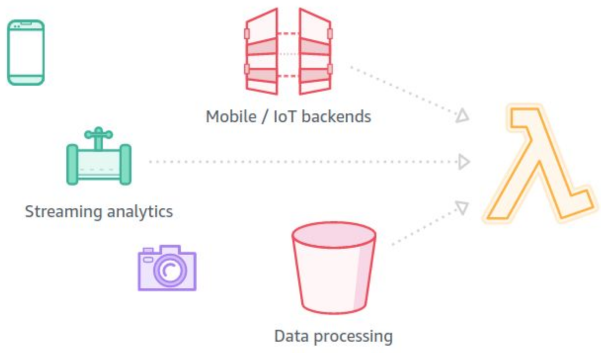
- Abstracting Servers
  - In the Serverless Paradigm, we abstract away the complexities of managing servers
  - Servers are still there but hidden from developers
- Allows for devs to focus more on writing code, and not on deployment


Freedom to innovate and experiment
Faster time to market
Support for multiple programming languages
AWS Serverless

AWS Lambda

AWS Lambda: A Serverless Product

- Developers do two things:
  1. Write their code
  2. Describe how their code will be called
- AWS Lambda handles the rest:
  - Listens for function invocation requests
  - Automatically executes function calls and returns output
  - Automatically provisions servers according to demand
- Lambda Console



AWS Lambda: Invoking Functions

- Many ways to invoke a Lambda function manually:
  - HTTP endpoint
  - Lambda API
  - AWS SDK/CLI
- But Lambdas can also be “triggered” when certain events happen:
  - Changes to an S3 bucket
  - API Gateway requests
  - Reads / writes on a database
  - Scheduled events
- The developer has a lot of control when configuring event triggers

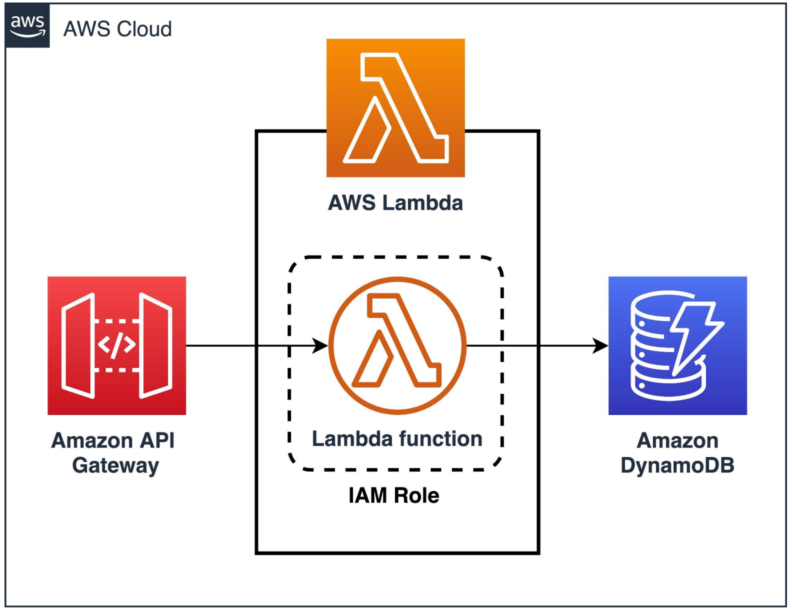
AWS Lambda: Benefits

- No need to worry about hardware, CPUs, operating systems, system maintenance, etc
- Automatically scales number of workers
  - Scale to zero
- Cheaper, as you only pay for each function invocation
  - EC2 nano: $0.0042/hr (512MiB)
  - Lambda: $0.0000000083/ms (512MiB)
- 1 mil free requests/month
- Stateless

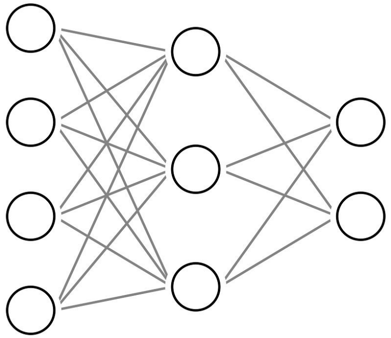
Example Use Cases

Example Apps

- Scalable AI deployment
- Token / password generator
- Memo / notes app
- Image processing apps
- File conversion app
- Notification systems
- Discord bot



Example: AI Deployment



Idea:

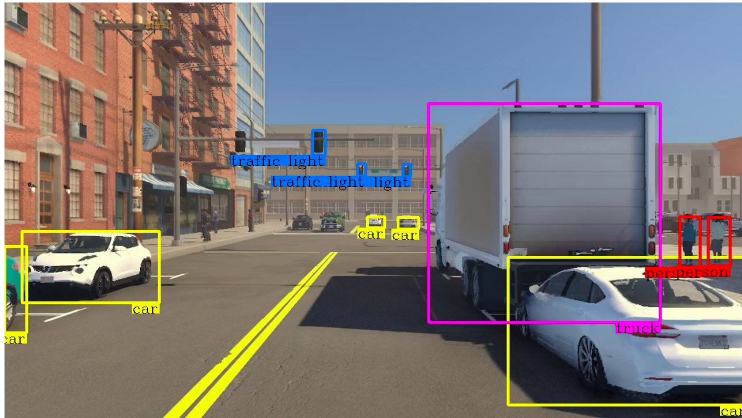
- You're an AI startup that has many AI models
- You want users to be able to use them

How to build:

- Write functions that take in user input, run the AI, and return output
- Possible Event trigger:
    - If vision AI: S3 image upload
    - If text-based AI: API request

AI Deployment: Object Detection

- Say we have an object detection model
    - Looks at an image, returns positions of objects in the image
- Write a function that takes input image and runs our model on it
- Upon image upload to S3 bucket:
    - Automatically run the model on the image
    - Return or store output



Example: Token Generator

- Imagine we’re building a system that provisions resources
  - Building a large scale API: need to generate user API keys
  - Building Steam: generate game keys
  - Building password manager: generate passwords
- We write our token generation function(s)
- We specify the event triggers:
  - Create token upon purchase: Database write trigger
  - If we want a regular REST API: use API Gateway trigger

Example: Memo App

App idea:

- Want an app where users can write memos / notes at any time
- Users can also generate a PDF file of all their past memos, nicely formatted

How to build:

- We write a function to generate a PDF given a bunch of text files from S3
- Function is triggered by a text file upload to an S3 bucket

Let’s get started!

Workshop

- Make an AWS account
- Create a lambda function
- Run in test environment
- Create API Gateway
  - Create request model
- Run lambda as a real API

Full code:

github.com/mattnappo/lambda-demo

Sources

https://www.skyhighsecurity.com/about/newsroom/blogs.html#:~:text=In%20terms%20of%20revenue,%20Amazon,reported%20revenues%20of%20%249%20billion.

https://www.contino.io/insights/whos-using-aws