Serverless on AWS Lambda

Let’s Talk About

- Serverless
- AWS
- Lambda
- Demo/Explain Project
- Could spend hours going through Lambda - beginners overview. This targeted to users who have never had hands on experience with AWS specifically, Lambda.

What is Serverless

- Serverless:
    - Application which uses resources only when triggered.
    - Still runs on servers but the server management is done by the host provider.
- Serverless application:
    - Are of one or more functions invoked by events
- Serverless architectures:
    - How the serverless pipeline is structured and built
    - Can include third party applications/integrations (slack, github, travisci, etc)
- Serverless deployment tools:
    - Helps speed up development
    - serverless, zappa, dotnet core, gradle https://gradle.org/
- Pros
    - No servers to maintain
    - Take advantage of parallel processing
    - Low cost and less code
    - Can be used for quick prototyping
- Cons
    - Should not be relied upon for critical applications
    - SLA may not meet your company needs
    - Customization and flexibility are limited
    - Locked into a vendor (third-party services)
    - Moving to serverless does not guarantee less complexity

What is AWS

- Cloud services platform that allows customers to run computing applications without the need for hardware. AWS maintains the data centers that are essentially the “cloud” for the client.
- Consists of 16 geographic regions
- Each region has availability zones built in areas of types of possible natural disasters, 42 Availability Zones across all regions
    - We do not need to understand availability zones for this project, but it is important to understand when building highly available infrastructures.
- Services
    - Not all services are available in all regions. Check the SERVICES PAGE for region compatibility. SNS FIFO queue as an example.
    - **Application Hosting**
        i. Use reliable, on-demand infrastructure to power your applications, from hosted internal applications to SaaS offerings.
    - **Websites**
        i. Satisfy your dynamic web hosting needs with AWS’s scalable infrastructure platform.
    - **Backup and Storage**
        i. Store data and build dependable backup solutions using AWS’s inexpensive data storage services.
    - **Enterprise IT**
        i. Host internal- or external-facing IT applications in AWS's secure environment.
    - **Content Delivery**
        i. Quickly and easily distribute content to end users worldwide, with low costs and high data transfer speeds.
    - **Databases**
        i. Take advantage of a variety of scalable database solutions, from hosted enterprise database software or non-relational database solutions.
    - Could run lambda without other services and resources but all you would have is a hello world.
    - Today’s Project uses:
        i. IAM - Roles and Policies
        ii. S3 - File storage
        iii. Simple Notification Service - Messaging and notification service
        iv. CloudWatch - Service executions and stack traces
        v. Lambda - Serverless code execution
        vi. Rekcognition - Image analyzing service

What is AWS Lambda

- Run code without needing to manage servers; AWS manages all the hardware and underlying code.
- Runs in a container
  - Container is a small section of a computer that allocates compute resources to create an isolated environment of different system configurations (i.e. lower version of php) The can be destroyed and rebuilt easily and quickly.
  - reuses container, but not guaranteed
  - Write up to 500MB in the /tmp directory that is non-persistent
  - Image is latest AMI
- Executes on response from event source
  - API Gateway, AWS IoT, CloudFront, CloudWatch Events - Schedule, CloudWatch Logs, CodeCommit, Cognito Sync Trigger, DynamoDB, Kinesis, S3, SNS, or can be triggered by your own code.
  - Response in milliseconds
- Scales at the rate of events
  - max 1000 concurrent executions)
  - Lambda functions being invoked asynchronously can absorb reasonable bursts of traffic for approximately 15-30 minutes, after which incoming events will be rejected as throttled
  - Stateless for scaling performance, handle state another way

Languages

- Java
  - Personally never used Java in Lambda function or in life
  - Know that it also supports Scala
- C#
  - Slower to load, but subsequent requests are quick
  - Join the lambda sharp meetup and learn more about using C# in lambda
  - Uses dotnet core to deploy
- Python
  - My personal choice
  - Good to use a framework to package deployments if you have a lot of dependencies. Can be a pain to re-deploy manually.
  - Zappa is a deployment framework for python.
- Node.js
  - My least personal choice
  - Serverless is the most popular framework to package and create infrastructure
  - Alexa apps are supported in the Javascript AWS SDK

- Event
  - Each language receives event information about the event request
  - The service sending the event will determine what kind of information is passed
- Context
  - Runtime information for that specific request.
  - Execution time remaining before AWS Lambda terminates
  - The CloudWatch log group and log stream
  - AWS request ID that invoked the Lambda function. You can use the request ID for any follow up inquiry with AWS support.

How to Use?

- Project files and documentation
  - Show github, explain how to download
  - Show github wiki, explain how to use
- First we need to setup the external resources and determine the pipeline.
  - What are we going to build…. Explain
  - Could use a framework that will create and configure for you automatically, but believe that if you are learning, hands on is the best way to learn.
  - Easier to debug when having issues using a framework.
- Deployment
  - via Console
  - via Command line
  - Requires packaging dependencies or compiling
  - Upload to S3 > 50MB
  - Today, Single file lambda function, deployed via command line
- Configure Lambda
  - Set trigger
  - Choose amount of memory, max execution time (up to 5 min) it then auto allocates CPU power, network bandwidth, and disk I/O
  - Environment variables
  - Grant permissions using IAM
  - Timeout
  - VPC
- Secrets
  - Encrypt with KMS
  - Encrypt with KMS and provided certificate
  - Parameter Store
- Monitoring automatic in CloudWatch
  - # of requests
  - Latency
  - Error rates

    - Throttled requests
- Debugging
    - CloudWatch Logs - error stacktrace
    - Use logging in your code
    - IDE integration
- Constraints
    - Inbound network connections are blocked
    - Outbound connections only supports TCP/IP (except port 25)
    - Debugging system calls are not allowed
    - Cannot access the infrastructure Lambda runs on and cannot installing via ‘yum’
    - Stateless
    - Max execution time is 5 minutes, min 1 second
    - Only supports 4 languages, with 2 of those languages supporting 2 versions each

Use Cases

- Backend application - resizing an image after it uploads
- Data processing - transform data before sending to final destination, Step Functions to coordinate calls between lambda functions
- Analytics - analyze, aggregate and store event, data, or logs immediately
- API - an endpoint that will trigger the function
- Schedule - a function which will trigger at scheduled times
- Bots - an Alexa skill which triggers a lambda function

Cost

- Request charges:
    - First 1 million requests per month are free
    - $0.20 per 1 million requests thereafter ($0.0000002 per request)
- Compute charges:
    - The memory chosen determines the number of free tier seconds and price per 100ms.

Memory (MB)	Free tier seconds per month	Price per 100ms ($)
128	3,200,000	0.000000208
192	2,133,333	0.000000313
256	1,600,000	0.000000417
320	1,280,000	0.000000521

384	1,066,667	0.000000625

Etc…

- Other charges:
    - The cost for other resources (i.e. S3 write charges regardless of source)
    - Billing and Alerts
        - AWS breaks down billing pretty well, have additional tools you can use to analyze billing
        - Set billing alerts so you'll be notified if a service is charging you too much.
        - If you accidently spent too much, talk to AWS, they are pretty forgiving.

Sources

- https://aws.amazon.com/lambda/
- https://aws.amazon.com/lambda/details/
- http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction.html
- https://aws.amazon.com/lambda/faqs/
- https://icons8.com/

Resources:
http://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html
http://docs.aws.amazon.com/lambda/latest/dg/building-lambda-apps.html
http://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction-function.html
https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

- What can you do in the lambda function
    - Write/retrieve states
- Serverless architecture guidelines:
    - Execute code on demand (no idle servers).
    - Write functions that serve a single purpose.
    - Design push-based, event-driven pipelines.
    - Should be quick to execute and not cause latency for the end user.
    - Use third-party services in your architecture (i.e. GitHub, Auth0, etc)

- Choosing 256MB of memory allocates approximately twice as much CPU power to your Lambda function as requesting 128MB of memory, and half as much CPU power as choosing 512MB of memory.
- You can set your memory in 64MB increments from 128MB to 1.5GB.
- Deployment times may vary with the size of your code, but AWS Lambda functions are typically ready to call within seconds of upload.
- Lambda Execution Environment and Available Libraries
  http://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html
- Create a mobile back-end using AWS Lambda Invoke it from your mobile app using the AWS Lambda SDK included in the AWS Mobile SDK.
- Use HTTPS by defining a custom RESTful API using Amazon API Gateway
- What happens if my function fails while processing an event?
  For Amazon S3 bucket notifications and custom events, AWS Lambda will attempt execution of your function three times in the event of an error condition in your code or if you exceed a service or resource limit. For ordered event sources that AWS Lambda polls on your behalf, such as Amazon DynamoDB Streams and Amazon Kinesis streams, Lambda will continue attempting execution in the event of a developer code error until the data expires. You can monitor progress through the Amazon Kinesis and Amazon DynamoDB consoles and through the Amazon CloudWatch metrics that AWS Lambda generates for your function. You can also set Amazon CloudWatch alarms based on error or execution throttling rates.
- How do I automate deployment for a serverless application?
  You can automate your serverless application's release process using CodePipeline. CodePipeline is a continuous delivery service that enables you to model, visualize and automate the steps required to release your serverless application. To learn more about serverless CI/CD, visit our documentation.
- How do I deploy and manage a serverless application?
  You can deploy and manage your serverless applications using the AWS Serverless Application Model (AWS SAM)
- troubleshoot a serverless application?
  You can enable your Lambda function for tracing with AWS X-Ray by adding X-Ray permissions to your Lambda function's execution role and changing your function's "tracing mode" to "active.
- Lambda@Edge?
  Lambda@Edge allows you to run code at global AWS edge locations without provisioning or managing servers, responding to end users at the lowest network latency. You just upload your Node.js code, triggered in response to Amazon CloudFront requests
- What Amazon CloudFront events can be used to trigger my functions?
  Your functions will automatically trigger in response to the following Amazon CloudFront events:
  Viewer Request - This event occurs when an end user or a device on the Internet makes an HTTP(S) request to CloudFront, and the request arrives at the edge location closest

to that user.
Viewer Response - This event occurs when the CloudFront server at the edge is ready to respond to the end user or the device that made the request.
Origin Request - This event occurs when the CloudFront edge server does not already have the requested object in its cache, and the viewer request is ready to be sent to your backend origin webserver (e.g. Amazon EC2, or Application Load Balancer, or Amazon S3).
Origin Response - This event occurs when the CloudFront server at the edge receives a response from your backend origin webserver.

- Can use threads and processes

- Benefits
    - Continuously scales
    - Only charged per use, based on the configuration of the lambda function, billed in increments of 100 milliseconds
    - Fault Tolerance, automatically supports through multiple availability zones
- Cons
    - Stateless
    - Max execution time is 5 minutes, min 1 second
    - Only allows 4 languages
    - Cannot access the infrastructure Lambda runs on
- If the Lambda function is invoked through AWS Mobile SDK, you can learn more about the mobile application calling the Lambda function.