



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Cloud Native Application - Serverless II

Cloud Infrastructure Engineering

**Nanyang Technological University
& Skills Union - 2022/2023**

Course Content

- Self Study Check In
- Introduction to Serverless
- Pros & Cons of Serverless
- Activity

Q1: What is the maximum execution time for an AWS Lambda function?

- a. 5 minutes
- b. 10 minutes
- c. 15 minutes

Q2: What is an event source for an AWS Lambda function?

- a. An external trigger that causes the function to run.
- b. An input parameter passed to the function.
- c. A configuration setting that determines the runtime environment for the function.

Q3: How to increase the CPU allocation of the Lambda Function?

- a. Increase CPU
- b. Increase Memory
- c. There is no way to increase CPU

Q3: How can you grant permissions for your Lambda Function to access a AWS Service?

- a. Using Lambda Resource-based Policy
- b. Using Lambda Execution role
- c. Using Lambda URL - AWS_IAM Authorizer

Activity

Instructor

- Ask to use AWS use single region for all learner for easier monitoring

AWS Lambda - Basics



Why AWS Lambda?

- No servers to manage
 - No patching required
- Limited by time - short executions
- Run on-demand
 - Not continuously running(eg:- ec2)
- Scaling is automated
 - How?
 - Concurrency options

Benefits of AWS Lambda

- Cheaper(Especially for inconsistent loads)
- Can integrate almost any AWS service(using SDK)
- Supports many programming languages



Lambda- Programming Language Support

- Node.js/JavaScript
- Python
- Java
- C#
- Golang
- Ruby
- For other languages:
 - Custom Runtime API
 - Lambda Container Image
 - Note:- Container image must implement Lambda Runtime API

AWS Lambda - Security

- What is your Lambda function permitted to do?

- How do you give granular access to your lambda function?

- Who can invoke your lambda function? Can we restrict?

Lambda - Execution Role

- What is your Lambda function permitted to do?

Lambda Execution Role : grants permissions to access AWS resources from your Lambda function

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#) .

- ☒ Create a new role with basic Lambda permissions
- ☐ Use an existing role
- ☐ Create a new role from AWS policy templates

 Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named hello-world-role-m8mp3cc3, with permission to upload logs to Amazon CloudWatch Logs.

Lambda - Execution Role

- Q) What permissions you should add in your lambda execution role?

```
import json
import boto3
import uuid

dynamodb = boto3.resource('dynamodb',region_name='us-east-1')
table = dynamodb.Table('Notes')

def create_note(event, context):
    print(event)
    data = json.loads(event['body'])
    note = {
        'id': data['id'],
        'content': data['content'],
        'createdAt': str(data['createdAt'])
    }

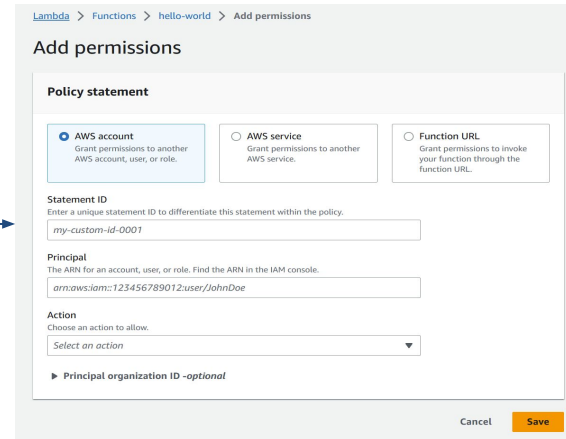
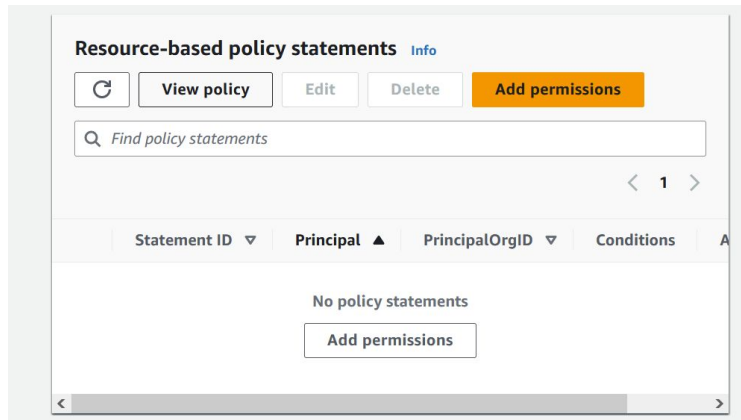
    table.put_item(Item=note)

    response = {
        'statusCode': 200,
        'body': json.dumps(note)
    }
    return response
```

Lambda - Resource Based Policies

- Who can invoke your lambda function? Can we restrict?
- How do you give granular access to your lambda function?

Resource-based policy: AWS services and other AWS accounts **receive permissions to invoke** your function



Lambda - Resource-based Policies

Resource-based policy document

```
1 {  
2   "Version": "2012-10-17",  
3   "Id": "default",  
4   "Statement": [  
5     {  
6       "Sid": "lambda-28f75fec-65c6-41b7-a60e-aa0bdd8a1219",  
7       "Effect": "Allow",  
8       "Principal": {  
9         "Service": "s3.amazonaws.com"  
10      },  
11      "Action": "lambda:InvokeFunction",  
12      "Resource": "arn:aws:lambda:ap-southeast-1:001687235961:function:hello-world",  
13      "Condition": {  
14        "StringEquals": {  
15          "AWS:SourceAccount": "001687235961"  
16        },  
17        "ArnLike": {  
18          "AWS:SourceArn": "arn:aws:s3:::test-bucket-ntu-s3-event-lambda"  
19        }  
20      }  
21    }  
22  ]  
23 }
```

Activity:

1. Create a lambda function(AWS management console)
2. Create a S3 bucket
3. Add a S3 trigger
(Choose your S3 bucket as the trigger)
 - Note:- Resource-based policy automatically created
4. Check Resource-based policy in Lambda

Lambda - General configuration

RAM:

- We need to configure **RAM(Max Memory)** for Lambda Functions
 - Min: 128MB
 - Max: 10GB
- No need to configure CPU
 - More RAM you add, the more vCPU credits you get

Timeout: 3 seconds(default), Maximum 15 mins(900 seconds)

General configuration [Info](#) Edit

Description A starter AWS Lambda function.	Memory 128 MB	Ephemeral storage 512 MB
Timeout 0 min 3 sec	SnapStart Info None	

Where to check the logs if
my Lambda function fails?

Lambda - Logging, Monitoring & Tracing

How to check the total number of requests,
latency, Success/Error rates, and execution
duration?

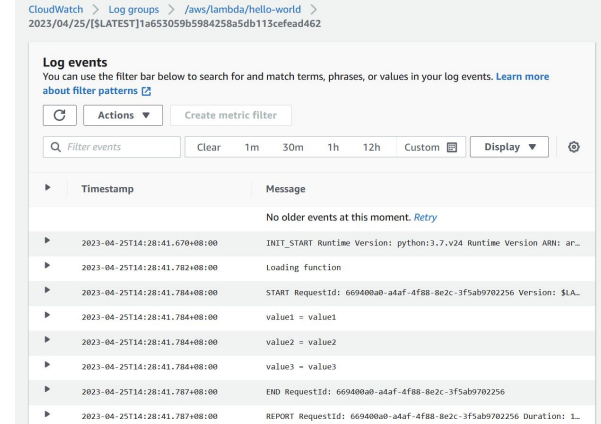
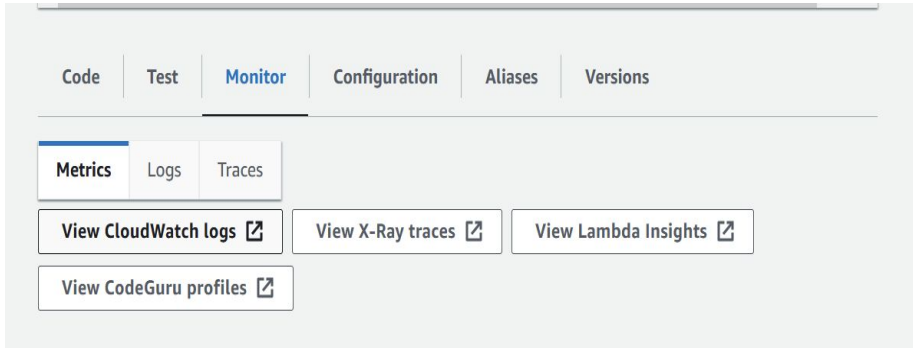
Which component of my application causing
performance bottleneck?

Lambda - Logging



CloudWatch Logs

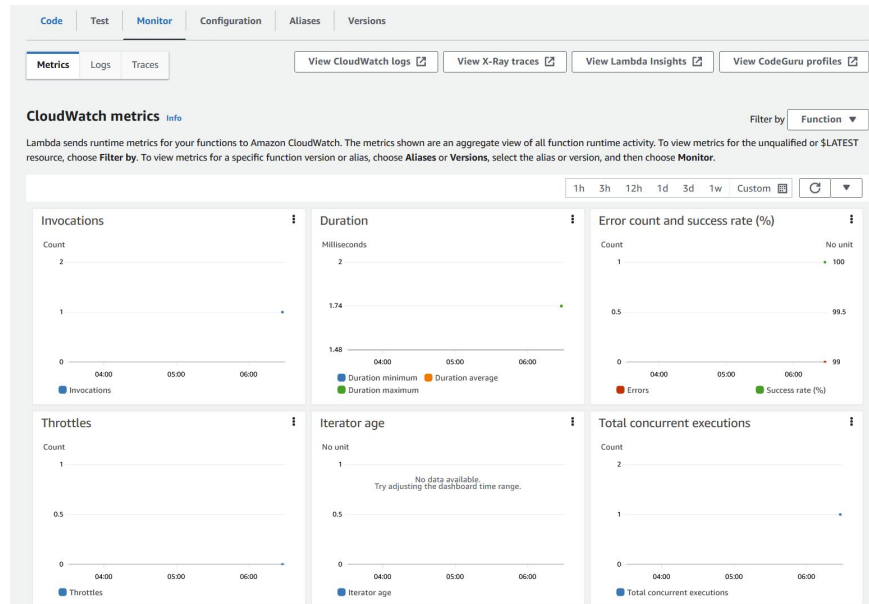
- By default, Cloudwatch logs are stored in AWS CloudWatch Logs.
- Make sure to include an execution role with an IAM policy that authorizes writes to CloudWatch Logs
- `/aws/lambda/<function name>`



Lambda - Monitoring



CloudWatch Metrics



Invocations – number of times your function is invoked

Duration – amount of time your function spends processing an event

Errors – number of invocations that result in a function error

Throttles – number of invocation requests that are throttled (no concurrency available)

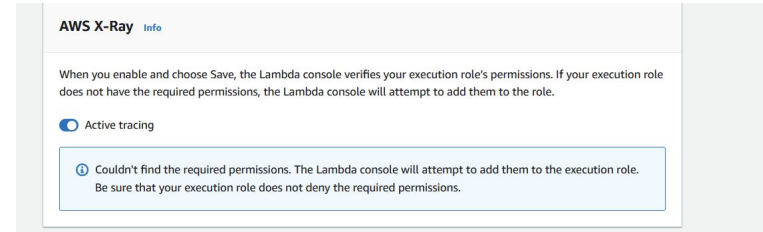
Total Concurrent Executions – number of function instances that are processing events

Lambda - Tracing



X-Ray

- You can enable in Lambda configuration
- Ideal tracing for microservices architecture



Why?

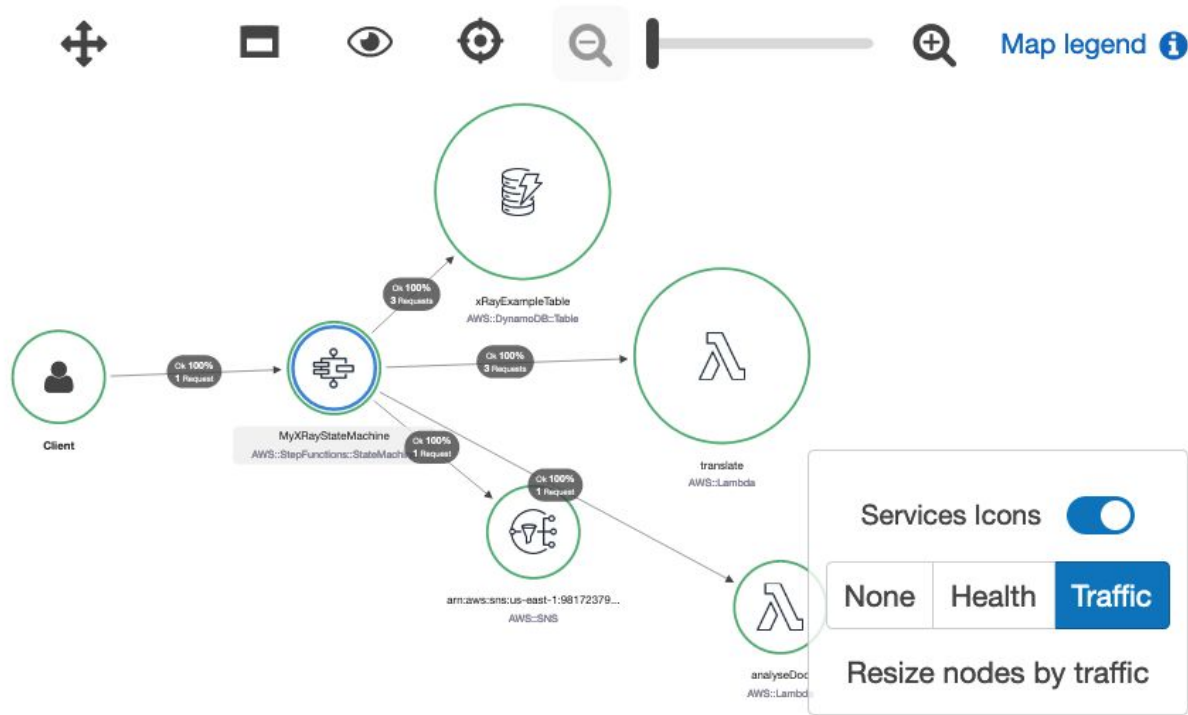
To provide an end-to-end view of your application to help you more efficiently **pinpoint performance bottlenecks** and **identify impacted users**

You can use the AWS X-Ray SDK to annotate the data sent to X-Ray, trace downstream calls, and record exceptions.

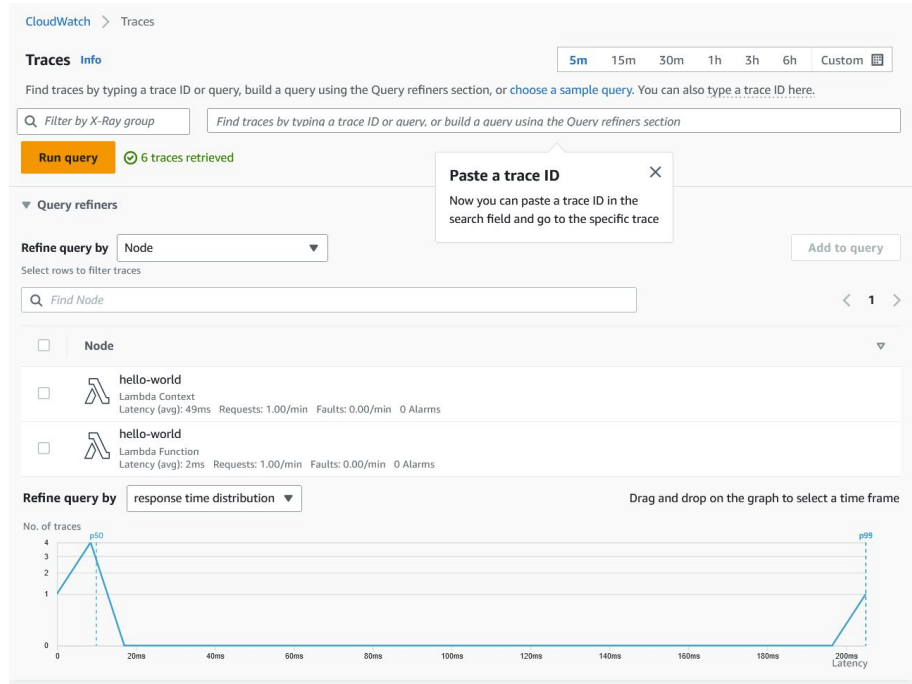
Example:

<https://docs.aws.amazon.com/xray/latest/devguide/xray-scorekeep.html>

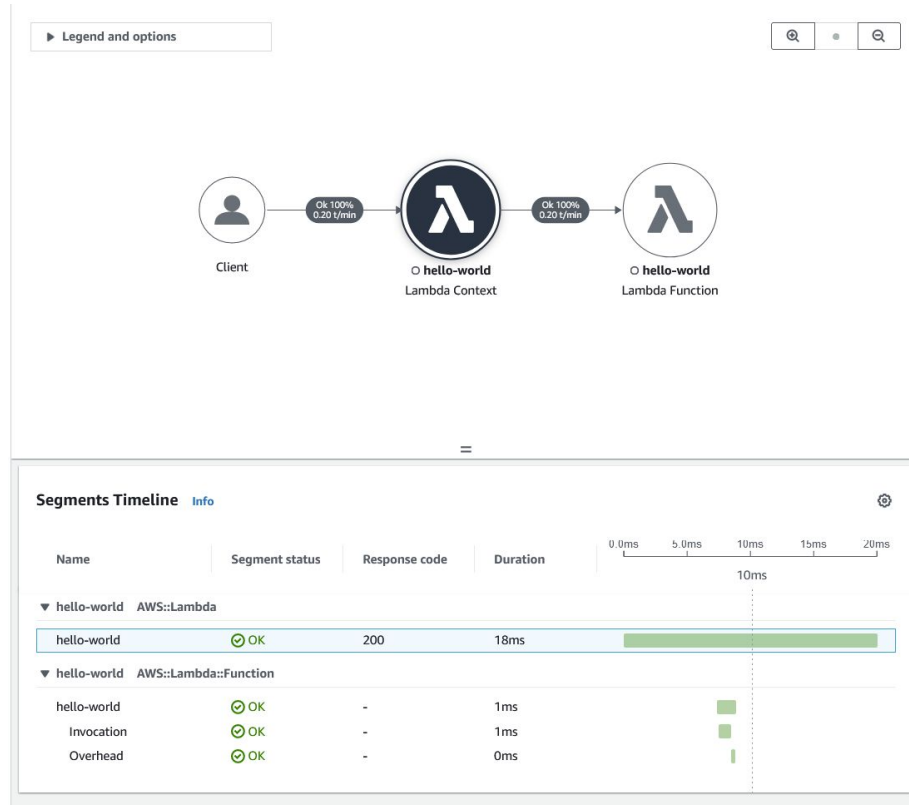
- Permissions required in Execution role:
AWSXRayDaemonWriteAccess



Lambda - Tracing



Lambda - Tracing



Lambda - CloudWatch Logs Insights

CloudWatch > Logs Insights

Logs Insights
Select log groups, and then run a query or choose a sample query.

5m 30m 1h 3h 12h Custom

Select log group(s)

/aws/lambda/hello-test-new

```
1 fields @timestamp, @message, @logStream, @log
2 | filter @message like "ERROR"
3 | sort @timestamp desc
4 | limit 20
```

Run query Cancel Save History

Queries are allowed to run for up to 60 minutes.

Logs Visualization Export results Add to dashboard

Showing 1 of 1 records matched
9 records (1.3 kB) scanned in 5.7s @ 1 records/s (238.003 B/s)

Hide Histogram

#	@timestamp	@message	@logStream	@log
1	2023-04-25T15:34:35.161Z	2023-04-25T07:34:35.161Z 06ddc23f-88b7-4af2-ae15-4b417ecca8bd ERROR Invoke...	2023/04/25/[SLATEST]f8b8869bba49491dbcca6427e01262b	0016872359f...

Field Value

@ingestionTime 1682488075649

@log 0016872359f1:/aws/lambda/hello-test-new

@logStream 2023-04/25/[SLATEST]f8b8869bba49491dbcca6427e01262b

@message 2023-04-25T07:34:35.161Z 06ddc23f-88b7-4af2-ae15-4b417ecca8bd ERROR Invoke Error {"errorType":"Error","errorMessage":"Something went wrong","stack":...

@requestId 06ddc23f-88b7-4af2-ae15-4b417ecca8bd

@timestamp 1682488075161

errorMessage Something went wrong

errorType Error

stack.0 Error: Something went wrong

stack.1 at Runtime.exports.handler (/var/task/index.js:8:11)

stack.2 at Runtime.handleOnColdStreaming (/var/runtime/Runtime.js:74:25)

CloudWatch Logs Insights

- Which allows to query your Lambda Cloudwatch logs

Eg:- Get the latest 20 invocation logs having errors

Lambda - CloudWatch Logs Insights

How to create your query?

CloudWatch > Logs Insights

Logs Insights

Select log groups, and then run a query or choose a sample query.

5m 30m 1h 3h 12h Custom

Select log group(s)

/aws/lambda/hello-test-new

```
1 fields @timestamp, @message, @logStream, @log
2 | filter @message like 'ERROR'
3 | sort @timestamp desc
4 | limit 20
```

Run query Cancel Save History

Queries are allowed to run for up to 60 minutes.

Logs Visualization

Export results Add to dashboard

Showing 1 of 1 records matched
9 records (1.3 kB) scanned in 5.7s @ 1 records/s (238.003 B/s)

Hide histogram

#	@timestamp	@message	@logStream
1	2023-04-25T15:34:35.161Z	2023-04-25T07:34:35.161Z 06ddc23f-80b7-4af2-ae15-4b417ecac8bd ERROR Invoke Error	2023/04/25/[\$LATEST]fbb8090ba49491db0cca6427e01262b

Field	Value
@timestamp	2023-04-25T15:34:35.161Z
@logStream	2023/04/25/[\$LATEST]fbb8090ba49491db0cca6427e01262b
@message	2023-04-25T07:34:35.161Z 06ddc23f-80b7-4af2-ae15-4b417ecac8bd ERROR Invoke Error
@requestId	06ddc23f-80b7-4af2-ae15-4b417ecac8bd
@timestamp	1662488875161
errorMessage	Something went wrong
errorType	Error
stack.0	Error: Something went wrong
stack.1	at Runtime.exports.handler (/var/task/index.js:8:11)
stack.2	at Runtime.handleOnceOnStreaming (/var/runtime/Runtime.js:74:25)

Query help

Commands

▼ fields

Retrieve one or more log fields. You can also use functions and operations such as `abs(a+b)`, `sqrt(a/b)`, `log(a)+log(b)`, `string(trim(0))`, `dateofcoul`, `isPresent()`, and others in this command.

```
fields errorMessage, errorType, stack.0, stack.1
```

Apply

▼ filter

Retrieve log fields based on one or more conditions. You can use comparison operators such as `=`, `!=`, `>`, `>=`, `<`, `<=`, boolean operators such as `and`, `or`, and `not`, and regular expressions in this command.

```
filter @message like /(?!(Exception|error|fail|500)/
```

Apply

▼ stats

Calculate aggregate statistics such as `sum()`, `avg()`, `count()`, `min()` and `max()` for log fields.

```
stats count() by bin(5s)
```

Apply

▼ sort

Sort the log fields in ascending or descending order.

```
sort @timestamp asc
```

Apply

▼ limit

Limit the number of log events returned by a query.

```
limit 10
```

Discovered fields

Learn more

Search for a field

Field	Percentage
@ingestionTime	100%
@logStream	100%
@message	100%
@timestamp	100%
@requestId	66%
@type	22%
@billedDuration	-
@duration	-
@initDuration	-
@maxMemoryUsed	-
@memorySize	-
errorMessage	-
errorType	-
stack.0	-
stack.1	-
stack.2	-

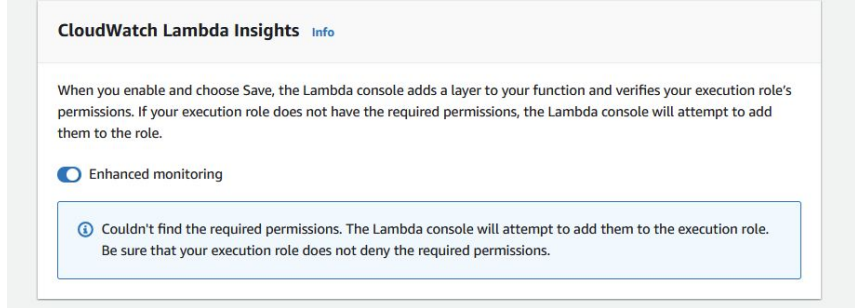
Learn more:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_AnalyzeLogData-discoverable-fields.html



Lambda - Cloudwatch Lambda Insights

Why?

- Right Sizing: over- and under-utilized Lambda functions(over/under provisioned memory)
- Performance Monitoring:
 - Max Memory usage, Network Usage



Watch:

<https://www.youtube.com/watch?v=m452gokXTME>

Lambda - Cold Start Problem

Problem: First request has higher latency than the rest (init duration)

The screenshot shows the AWS Lambda console interface. The 'Test' tab is selected. The execution result is 'succeeded'. The execution log shows a single log entry: 'value1'. The summary section displays the following details:

Code SHA-256	Request ID
gt+EEZE6AIOcOh2AT/vBDK4fgF489YwkZ9K1k41c/SE=	636ea3aa-5d5b-4e22-876c-f7bd1986b1c7

Init duration	Duration
174.21 ms	17.04 ms

Billed duration	Resources configured
18 ms	128 MB

Max memory used
57 MB

Log output

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

```
2023-04-25T06:54:37.535Z    undefined    INFO    Loading function
START RequestId: 636ea3aa-5d5b-4e22-876c-f7bd1986b1c7 Version: $LATEST
2023-04-25T06:54:37.540Z    636ea3aa-5d5b-4e22-876c-f7bd1986b1c7    INFO    value1 = value1
2023-04-25T06:54:37.540Z    636ea3aa-5d5b-4e22-876c-f7bd1986b1c7    INFO    value2 = value2
2023-04-25T06:54:37.540Z    636ea3aa-5d5b-4e22-876c-f7bd1986b1c7    INFO    value3 = value3
END RequestId: 636ea3aa-5d5b-4e22-876c-f7bd1986b1c7
REPORT RequestId: 636ea3aa-5d5b-4e22-876c-f7bd1986b1c7  Duration: 17.04 ms  Billed Duration: 18 ms  Memory Size: 128 MB  Max Memory Used: 57 MB  Init Duration: 174.21 ms
```

Solution :

- 1) “Provisioned Concurrency”
 - Concurrency is allocated before the function is invoked

The screenshot shows the 'Configure provisioned concurrency' dialog in the AWS Lambda console. The 'Qualifier type' is set to 'Alias'. The 'Alias' dropdown is empty. The 'Provisioned concurrency' field is set to '0 available'. The 'Pricing' field is set to '\$0.00 per month in addition to pricing for duration and requests. Pricing'. The 'Cancel' and 'Save' buttons are at the bottom right.

- 2) SnapStart (only for Java)

Lambda - Storage options

1) /tmp

Ephemeral storage [Info](#)

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

512

MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

2) Lambda Layers

3) S3

Lambda > Layers > Add layer

Add layer

Function runtime settings

Runtime Node.js 14.x	Architecture x86_64
-------------------------	------------------------

Choose a layer

Layer source [Info](#)

Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

☒ **AWS layers**
Choose a layer from a list of layers provided by AWS.

☐ **Custom layers**
Choose a layer from a list of layers created by your AWS account or organization.

☐ **Specify an ARN**
Specify a layer by providing the ARN.

AWS layers

Layers provided by AWS that are compatible with your function's runtime.

Choose

Cancel


Add

Lambda - Storage options

4) EFS

Lambda > Functions > hello-test-new > Add file system


Add file system

**VPC not configured**
To connect to a file system, you must first connect your function to the VPC where your file system runs.

File system

You can associate an existing Amazon Elastic File System (Amazon EFS) file system with your function. Visit the Amazon EFS console to [create a new file system](#).

EFS file system
Choose an existing EFS file system to use with your Lambda function.



Local mount path
Only absolute paths are supported.

Cancel Save

Lambda - Concurrency

The screenshot shows the AWS Lambda console's Configuration tab for a specific function. The left-hand navigation pane lists various configuration options, with 'Concurrency' selected and highlighted in blue. The main content area is divided into two sections. The top section, titled 'Concurrency', contains two settings: 'Function concurrency' set to 'Use unreserved account concurrency' and 'Unreserved account concurrency' set to '10'. An 'Edit' button is located in the top right corner of this section. The bottom section, titled 'Provisioned concurrency configurations', includes a descriptive paragraph about scaling with provisioned concurrency and a link to 'Learn more'. Below the text are buttons for 'Refresh', 'Edit', 'Remove', and 'Add'. A search bar with the placeholder 'Find configuration' is also present. At the bottom of this section is a table header with columns for 'Qualifier', 'Type', 'Provisioned concurrency', 'Status', and 'Details'. The table body is currently empty, displaying the message 'No configurations' and an 'Add configuration' button.

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
Monitoring and operations tools
Concurrency
Asynchronous invocation
Code signing
Database proxies
File systems
State machines

Concurrency Edit

Function concurrency
Use unreserved account concurrency

Unreserved account concurrency
10

Provisioned concurrency configurations

To enable your function to scale without fluctuations in latency, use provisioned concurrency. You can use Application Auto Scaling to automatically adjust provisioned concurrency to maintain a configured target utilization. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. [Learn more](#)

Refresh Edit Remove Add

Qualifier	Type	Provisioned concurrency	Status	Details
No configurations				
Add configuration				

Lambda - Pricing

Lambda Cost Calculator: <https://calculator.aws/#/addService/Lambda10>

Try to change the number of requests, memory and see the cost.



Lambda - Invocation options

1. Synchronous Invokes

Synchronous invocations are the most straightforward way to invoke your Lambda functions. In this model, your functions execute immediately when you perform the Lambda Invoke API call.

The Invocation-type flag specifies a value of “**RequestResponse**”. This instructs AWS to execute your Lambda function and wait for the function to complete. When you perform a synchronous invoke, you are responsible for checking the response and determining if there was an error and if you should retry the invoke.

2. Asynchronous Invokes

Asynchronous invokes **place your invoke request in Lambda service queue** and we process the requests as they arrive. You should use AWS X-Ray to review how long your request spent in the service queue by checking the “dwell time” segment.

Ref:

- <https://docs.aws.amazon.com/lambda/latest/operatorguide/invoke-modes.html>
- <https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html>

Synchronous Invokes

Here is a list of services that invoke Lambda functions synchronously:

- Amazon API Gateway
- Elastic Load Balancing (Application Load Balancer)
- Amazon Cognito
- Amazon Lex
- Amazon Alexa
- Amazon Kinesis Data Firehose

Asynchronous Invokes

Here is a list of services that invoke Lambda functions asynchronously:

- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon Simple Email Service
- CloudFormation
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS CodeCommit

Asynchronous Invokes

[Lambda](#) > [Functions](#) > [hello-world](#) > Edit asynchronous configuration

Edit asynchronous configuration

Retries [Info](#)

Maximum age of event
The maximum amount of time to keep unprocessed events in the queue.

6 h 0 min 0 sec

Retry attempts
The maximum number of times to retry when the function returns an error.

2

Dead-letter queue [Info](#)

Dead-letter queue service
You can send unprocessed events from an asynchronous invocation to an Amazon SQS queue or an Amazon SNS topic.

Amazon SQS

Queue
The name of the queue or topic.

Message Queues:

<https://aws.amazon.com/message-queue/>



Message Queue Architecture



AWS SQS

Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components.

Benefits of AWS SQS

Security – You control who can send messages to and receive messages from an Amazon SQS queue. You can choose to transmit sensitive data by protecting the contents of messages in queues by using default Amazon SQS managed server-side encryption (SSE), or by using custom SSE keys managed in AWS Key Management Service (AWS KMS).

Durability – For the safety of your messages, Amazon SQS stores them on multiple servers. Standard queues support at-least-once message delivery, and FIFO queues support exactly-once message processing and high-throughput mode.

Availability – Amazon SQS uses redundant infrastructure to provide highly-concurrent access to messages and high availability for producing and consuming messages.

Scalability – Amazon SQS can process each buffered request independently, scaling transparently to handle any load increases or spikes without any provisioning instructions.

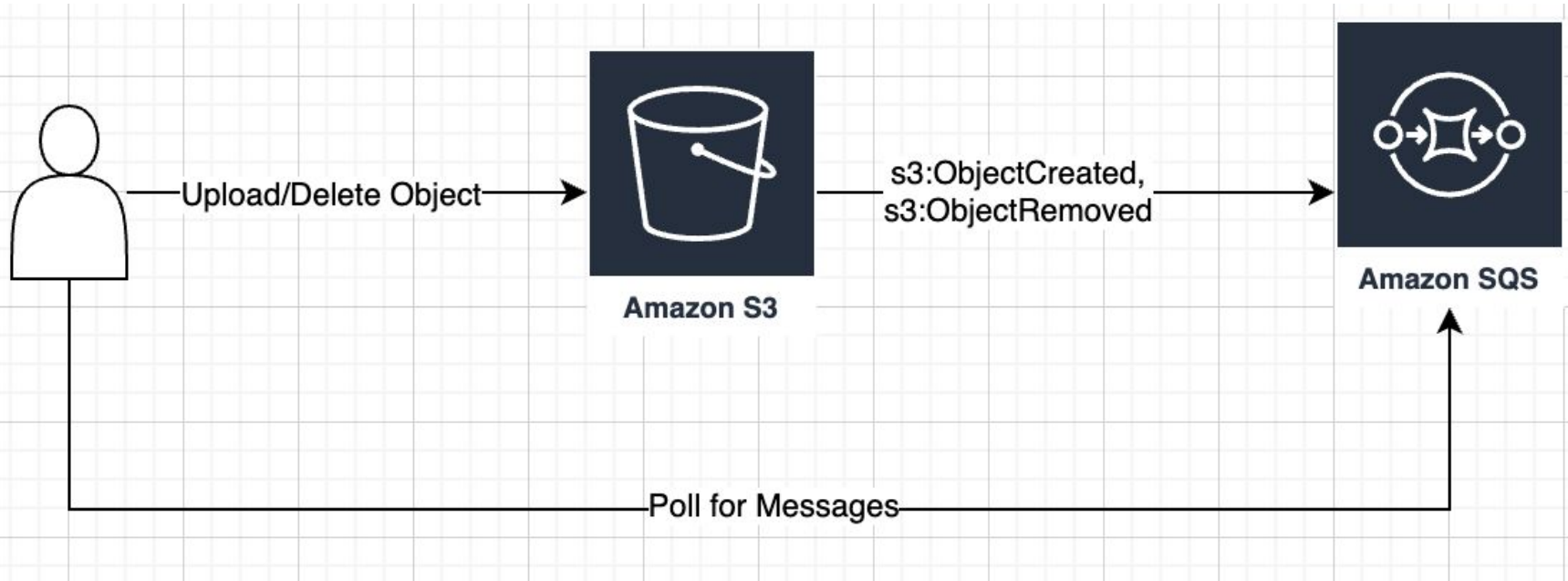
Reliability – Amazon SQS locks your messages during processing, so that multiple producers can send and multiple consumers can receive messages at the same time.

Customization – Your queues don't have to be exactly alike—for example, you can set a default delay on a queue. You can store the contents of messages larger than 256 KB using Amazon Simple Storage Service (Amazon S3) or Amazon DynamoDB, with Amazon SQS holding a pointer to the Amazon S3 object, or you can split a large message into smaller messages.

Try S3 event with SQS



Activity



Activity

Demo: Creating above architecture with S3 and SQS using AWS Console



Trying it in Terraform



Activity

Create a new repo in Github and clone it to your local computer

Create your required files and file structure:

- 1) .gitignore -> terraform template
- 2) README.md
- 3) backend.tf
- 4) main.tf
- 5) provider.tf

Activity

Create a provider.tf file with region set to “ap-southeast-1”:

Create a backend.tf file(**optional but recommended**) with bucket set to “sctp-ce2-tfstate-bkt” and region set to “ap-southeast-1” .

Activity

Create a main.tf file using the guide below(NOTE: You will need to make some changes based on the guide below. Mostly the same):

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_notification#add-notification-configuration-to-sqs-queue

Activity

Once all files above have been created, Run the following commands:

terraform init

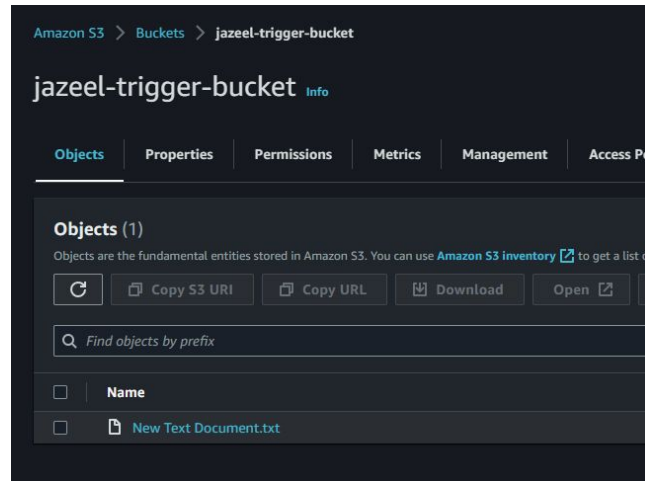
terraform plan

terraform apply



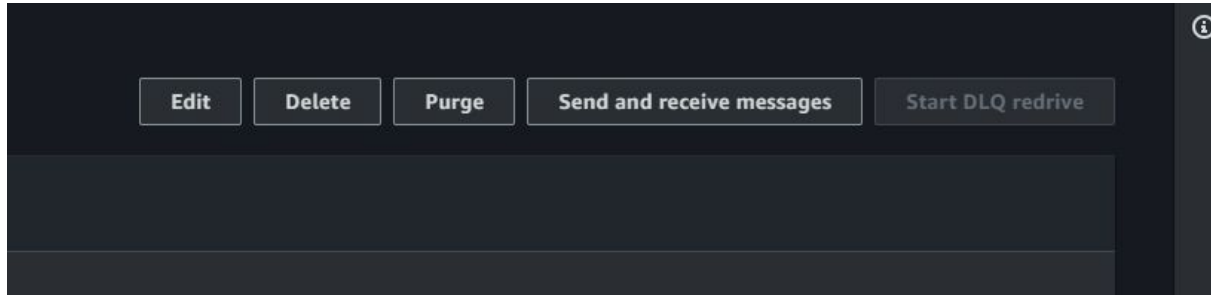
Activity

Once the terraform apply has completed, Go to AWS Console and go to the S3 bucket created and upload any file into it.



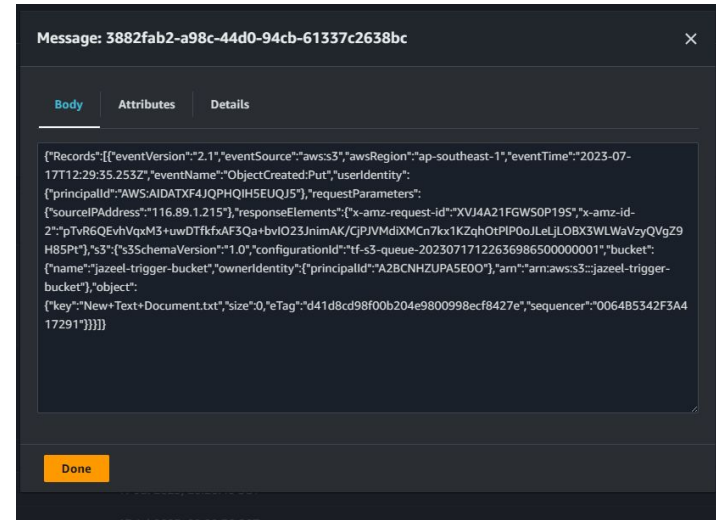
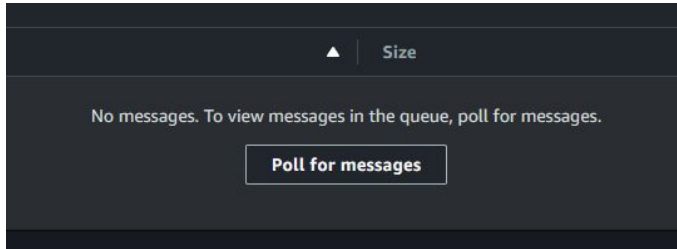
Activity

Once the upload has completed, Go to AWS SQS service and go to the SQS Queue that you created. And then click on “Send and Receive messages” on the top right



Activity

Click on “Poll for messages”



Questions?



Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.
- Check the AWS account after learner clean up.

END



Try S3 event for Lambda



Activity

Let spend 2-3 mins to

- Learner create new repository on github



Activity

Instructor demo how to create lambda function with S3 event on the new repository.

Activity

Learners to create lambda function with S3 event on the new repository



Break for 10-15 mins

Learners check new lambda function that has been created on AWS.



Activity

Instructor demo how to access Lambda Function log on AWS Cloudwatch



AWS X-Ray & AWS Cloudwatch

If still have more time. If not, there will be a session to cover this lesson in more detail (3.15 Application Logging - CloudWatch)



AWS Cloudwatch

Amazon CloudWatch collects and visualizes real-time logs, metrics, and event data in automated dashboards to streamline your infrastructure and application maintenance.

AWS Lambda automatically monitors Lambda functions on your behalf, pushing logs to Amazon CloudWatch. To help you troubleshoot failures in a function, after you set up permissions, Lambda logs all requests handled by your function and also automatically stores logs generated by your code through Amazon CloudWatch Logs.

You can insert logging statements into your code to help you validate that your code is working as expected. Lambda automatically integrates with CloudWatch Logs and pushes all logs from your code to a CloudWatch Logs group associated with a Lambda function, which is named `/aws/lambda/<function name>`.

There will be separated topic that will focus only on AWS Cloudwatch

AWS X-Ray

AWS X-Ray makes it easy for developers to **analyze** the behavior of their production, **distributed applications with end-to-end tracing capabilities**. You can use X-Ray to identify performance bottlenecks, edge case errors, and other hard to detect issues. This enables developers to quickly **find and address problems** in their applications and improve the experience for end users of their applications.

AWS X-Ray creates a **map of services** used by your application with **trace data** that you can use to drill into specific services or issues. This provides a view of connections between services in your application and aggregated data for each service, including average latency and failure rates.

Activity

Instructor demo how to access Lambda Function log on AWS Cloudwatch

