

Group 1 Capstone Project

-SaleSpheres

Alan How

Jian Tao

Elezabeth

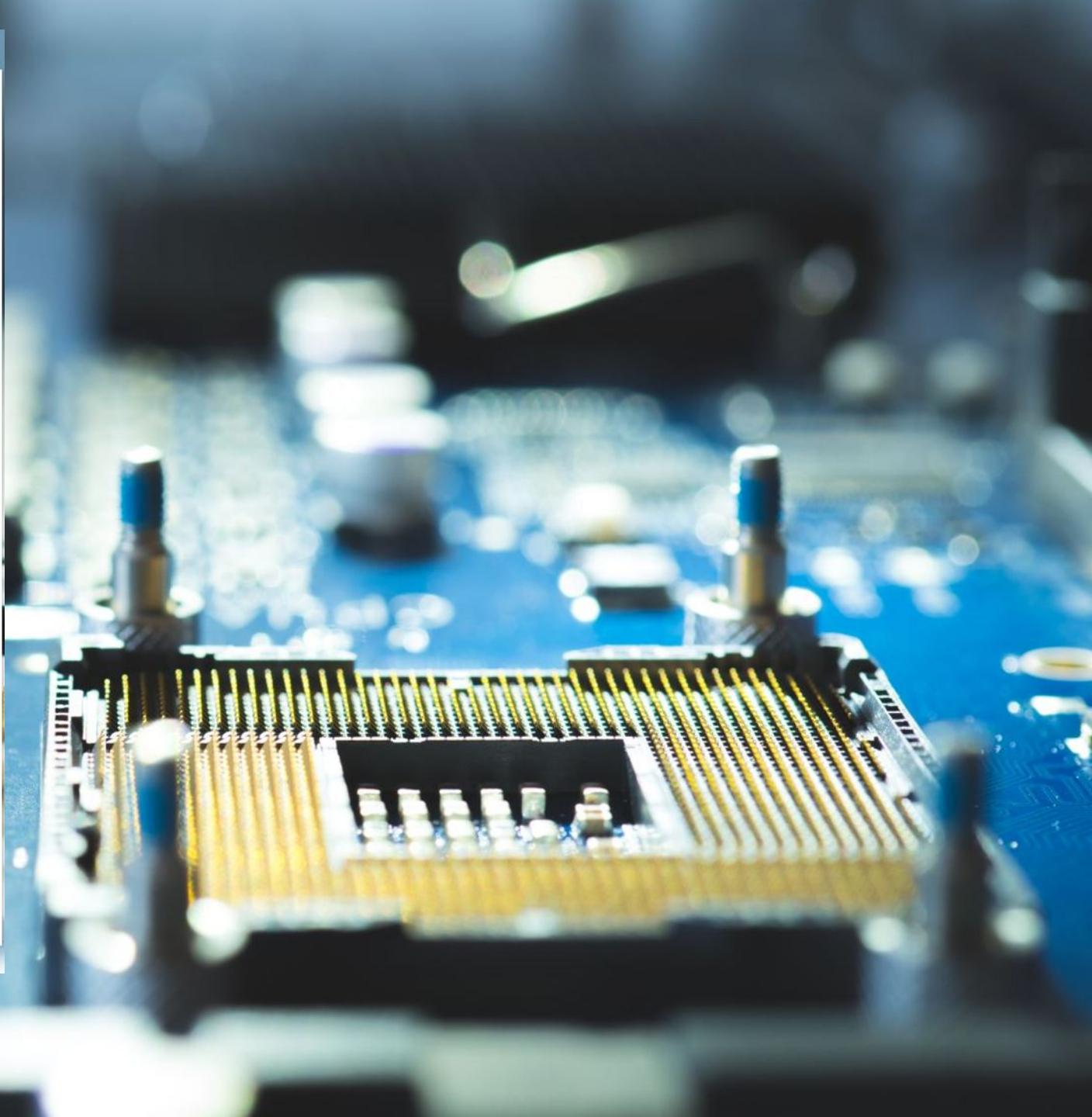
Monitoring Large Application SRE(Site Reliability Engineering)

- A comprehensive monitoring solution that can detect and flag out systems and applications failure early



STAGES OF PROJECT

- **Create a PROJECT** :- Create a repository in [github](#) and give public access and correct access to groupmates as collaborators
- **Add serverless application** :- added serverless application to ensure the portability and ability to deploy on any platform.
- **Application deployed to AWS**
- **Application logging** :-System health would be centrally managed using a monitoring tool cloud watch to watch the health of all microservices and alert us in case of any issues
- **Application Monitoring Dashboard** :-A dashboard that shows applications and logs would be produced which would allow us to create custom dashboards and shows logs and matrices from all microservices and systems.
 - By providing this traceable and observing comprehensive solution stakeholders can detect systems and application failures early .





Company Profile

- SaleSpheres is a software company driving and pushing innovation frontiers.
- Support Engineers are passionate in emerging technologies and desires to excel and work for success of the company
- Every individual has different values and capabilities to contribute and with great attitude of creating a collaborative environment.

Customer – MNC Company (XYZ)



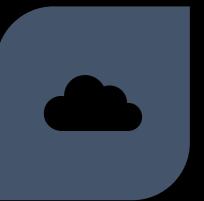
SOFTWARE COMPANY



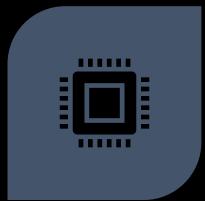
CUSTOMERS ARE MANY
LIKE INSIGHT, DBS , SMBC



FINDING ON HEALTH
CHECK FOR A SOLUTION
FROM US TO CLEAR THEIR
PAINS AND NEEDS



CLOUD SOLUTIONING
HEALTH CHECK



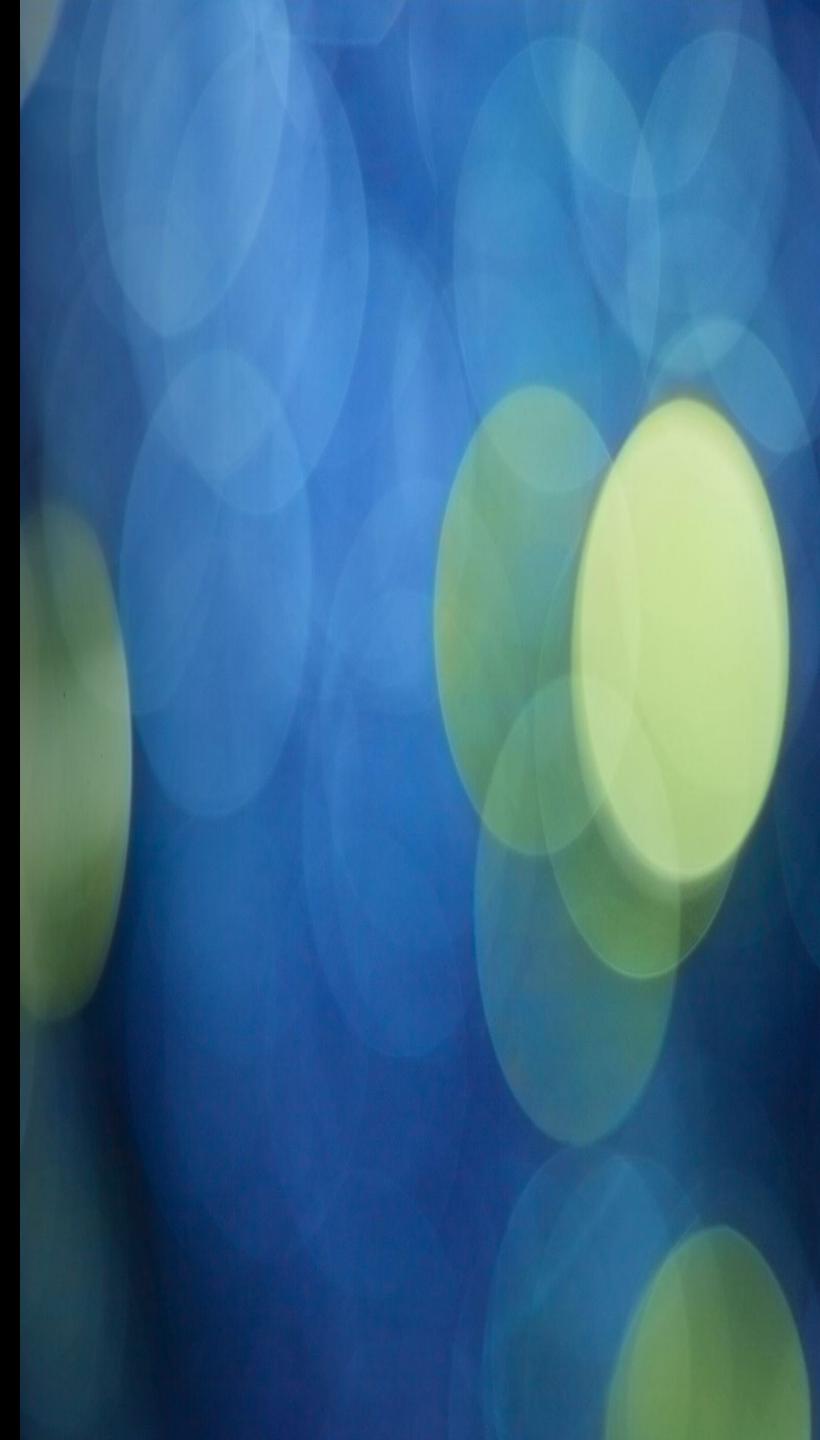
DASH BOARD CONTAINS INVOCATIONS,
DURATION, ERRORS , CONCURRENT
EXECUTION, THROTTLES ,



WE ARE GIVING A PROOF
OF CONCEPT(POC) HOW
TO MONITOR THE CLOUD
SOLUTION

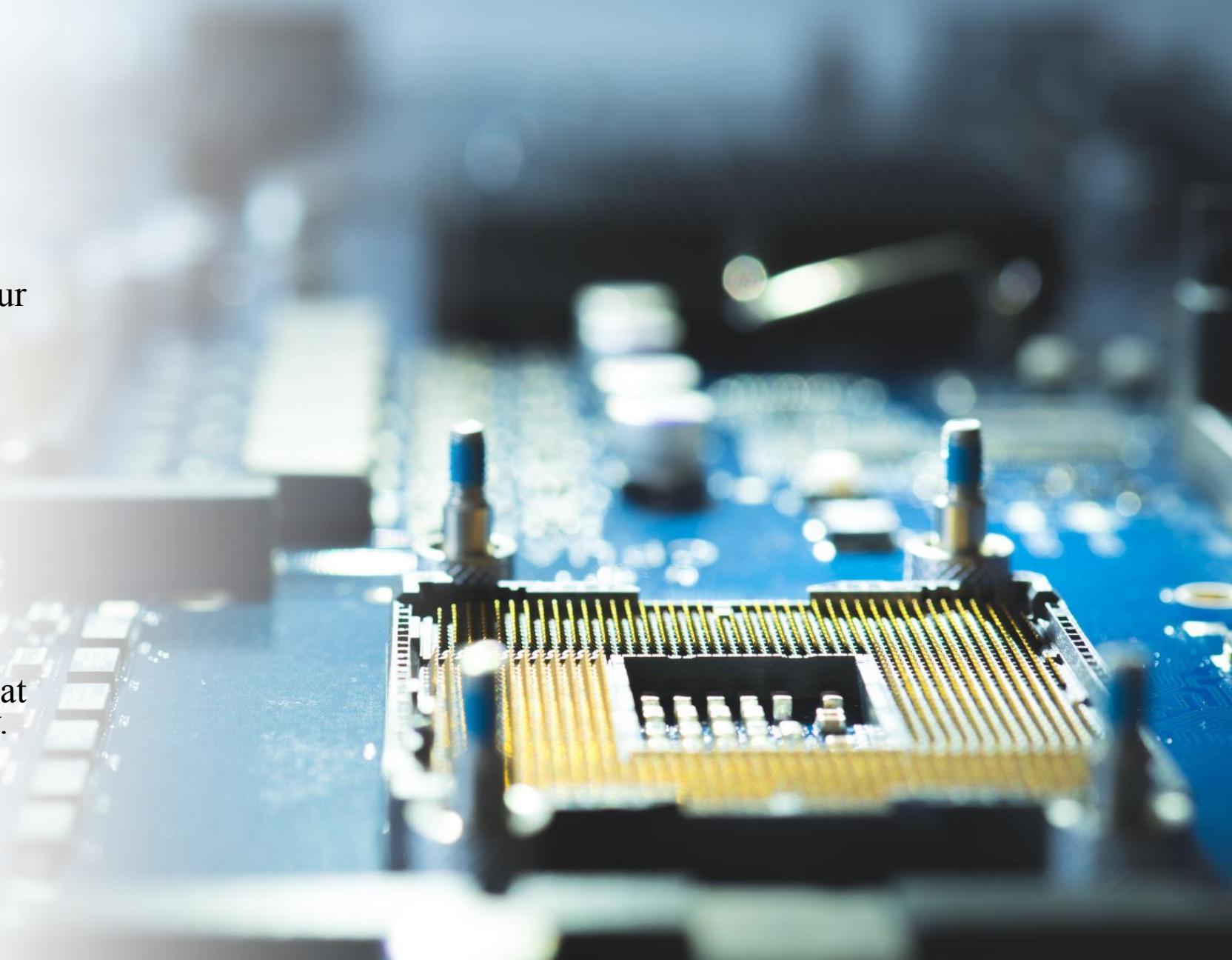


FIRST WE ARE
MONITORING
SERVERLESS PLATFORM
BASED ON LAMBDA



Site Reliability Engineer Team

- Our engineers are the pillars of our innovative products.
- They leverage their expertise in AWS cloud engineering skills to design, develop, and implement robust and scalable software solutions.
- Their skills range from front-end development for intuitive user interfaces to back-end systems that power our products' functionality.



Project-SaleSpheres

- Repository:
<https://github.com/flowstarts2020/SaleSpheres.git>
- The Site Reliability Engineer Team at **SaleSpheres** is currently working on an exciting project manifold complexity in systems' health monitoring.
- This project involves the development of application in AWS that implements a serverless architecture components as well as exploring Sense Deep a third-Party tool.
- The project will commerce with this objective and with further iteration improvement versions to be better.

A photograph of a person's hand pointing towards a computer monitor. The monitor displays a block of Python code. The code appears to be part of a larger script, likely for a 3D modeling application like Blender, dealing with object selection and mirroring operations. The background is dark, and the screen is brightly lit, creating a strong contrast.

```
mirror_mod = modifier_obj.modifiers.new("mirror", "MIRROR")
# Set mirror object to mirror
mirror_mod.mirror_object = context.scene.objects.active
# Set operation
if operation == "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
elif operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

# Selection at the end - add
one.select = 1
other.select = 1
context.scene.objects.active = context.scene.objects.active
print("Selected" + str(modifier))
mirror_mod.select = 0
bpy.context.selected_objects.append(data.objects[one.name])
data.objects[one.name].select = 1
print("please select exactly one object")

-- OPERATOR CLASSES ----

@types.Operator:
class X_mirror_to_the_selected(types.Operator):
    bl_idname = "object.mirror_mirror_x"
    bl_label = "Mirror X"
    bl_options = {'REGISTER', 'UNDO'}

    @classmethod
    def poll(cls, context):
        if context.active_object is not None:
```

To get the serverless architecture running locally:

Step 1: Create a repository in GitHub

The screenshot shows the GitHub interface for creating a new repository. At the top, there's a header with navigation icons, a search bar, and links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the header, the main title is "Create a new repository". A sub-instruction says "A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository." Under "Repository template", there's a dropdown menu set to "No template". A note below it says "Start your repository with a template repository's contents." In the "Owner" field, the user "flowstarts2020" is selected. The "Repository name" field is filled with "SaleSphere", which is highlighted in green. A red warning message "⚠ New repository name must not be blank" is displayed next to the field. Below the field, a suggestion "Great repository names are short and memorable. Need inspiration? How about [fluffy-lamp](#)?" is shown. There's a "Description (optional)" input field with a placeholder "Your description". At the bottom, there are two radio button options: "Public" (selected) and "Private". The "Public" option is described as "Anyone on the internet can see this repository. You choose who can commit." The "Private" option is described as "You choose who can see and commit to this repository."

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * Repository name *

flowstarts2020 / **SaleSphere**

⚠ New repository name must not be blank

Great repository names are short and memorable. Need inspiration? How about [fluffy-lamp](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.



Step 2:

Clone the code repository
into local machine

- Clone by the following command on visual studio code - PS C:> git clone <https://github.com/flowstarts/2020/SaleSpheres.git>

```
module.exports.error = async (event) => {
  const response = {
    "timestamp": Date.now().toString(),
    "status_code": 400,
    "body": "body error"
  }

  const err = new Error(response)

  return {
    statusCode: 400,
    body: JSON.stringify(
      {
        message: "error",
        input: err.message.body,
      },
      null,
      2
    ),
  };
};
```

Step 3: Create index.js file

Example of our index.js file
which was done via visual
studio code

```
module.exports.handler = async (event) => {
  const response = {
    "timestamp": Date.now().toString(),
    "status_code": 200,
    "body": "body"
  }
  console.log(response)
  return {
    statusCode: 200,
    body: JSON.stringify(
      {
        message: "Go Serverless v3.0! Your function executed successfully!",
        input: response,
      },
      null,
      2
    );
  };
}

module.exports.error = async (event) => {
  const response = {
    "timestamp": Date.now().toString(),
    "status_code": 400,
    "body": "body error"
  }

  const err = new Error(response)

  return {
    statusCode: 400,
    body: JSON.stringify(
      {
        message: "error",
        input: err.message.body,
      },
      null,
      2
    );
  };
}
```

```
salesphere
service: salesphere
frameworkVersion: '3'

provider:
  name: aws
  runtime: nodejs18.x
  region: ap-southeast-1

functions:
  api:
    handler: index.handler
    events:
      - httpApi:
          path: /
          method: get

  error:
    handler: index.error
    events:
      - httpApi:
          path: /error
          method: get
```

```
error:
  handler: index.error
  events:
    - httpApi:
        path: /error
        method: get

plugins:
  - serverless-offline
```

Step 4: Create serverless.yml

Example of a serverless.yml file which was done via visual studio code

Step 5:

Deploy and verify that the serverless application is working

Test and execute the following commands > `$ npm install`

then deploy with ->

`$ serverless deploy`

Step 6:

Run and test with GitHub Actions

- Create main.yml in .github/workflows folder.
- An example of a main.yml

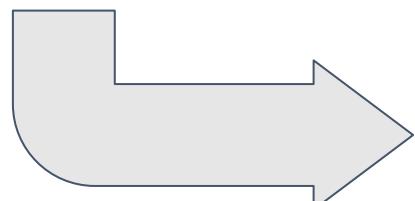
```
name: CICD for Serverless Application
run-name: ${{ github.actor }} is doing CICD for Serverless Application

on:
  push:
    branches: [ main, "*" ]

jobs:
  pre-deploy:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🤖 The job was automatically triggered by a ${{ github.event_name }} event"
      - run: echo "👤 This job is now running on a ${{ runner.os }} server hosted by GitHub!"
      - run: echo "🌐 The name of your branch is ${{ github.ref }} and your repository is ${{ github.repository }}."

  install-dependencies:
    runs-on: ubuntu-latest
    needs: pre-deploy
    steps:
      - name: Check out repository code
        uses: actions/checkout@v3
      - name: Run Installation of Dependencies Commands
        run: npm install
```

```
scan-dependencies:  
  runs-on: ubuntu-latest  
  needs: install-dependencies  
  steps:  
    - name: Check out repository code  
      uses: actions/checkout@v3  
    - name: Run Scanning of Dependencies Commands  
      run: npm audit  
  
deploy:  
  name: deploy  
  runs-on: ubuntu-latest  
  needs: install-dependencies  
  strategy:  
    matrix:  
      node-version: [18.x]  
  steps:  
    - uses: actions/checkout@v3  
    - name: Use Node.js ${matrix.node-version}  
      uses: actions/setup-node@v3  
      with:  
        node-version: ${matrix.node-version}  
    - run: npm ci  
    - name: serverless deploy
```



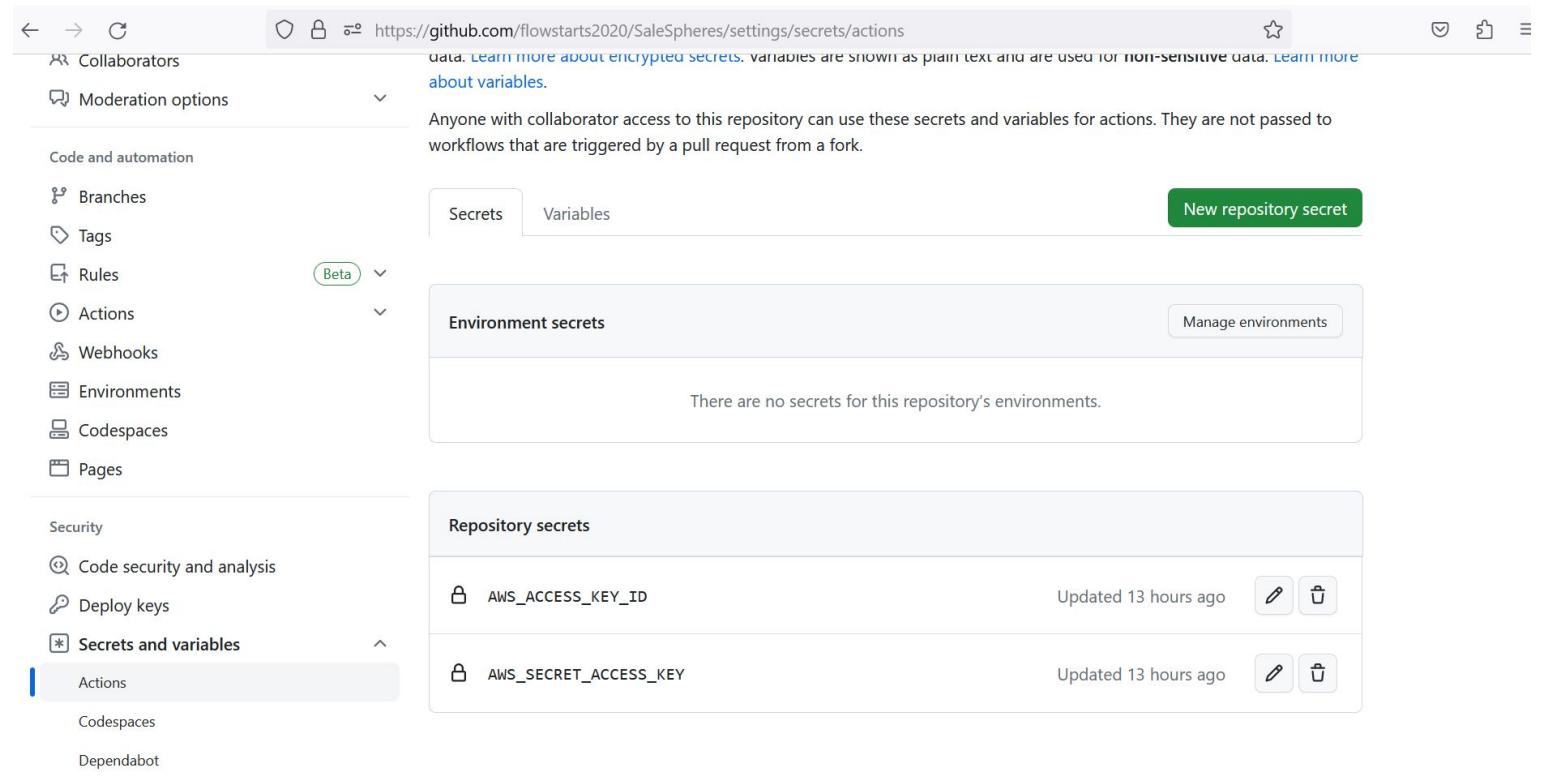
Continue..

```
- name: serverless deploy  
  uses: serverless/github-action@v3.2  
  with:  
    args: deploy  
    env:  
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}  
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
```

Step 7:

ADD
AWS_ACCESS_KEY_ID
&
AWS_SECRET_ACCE
S KEY
to GitHub Secrets

Always keep AWS_ACCESS_KEY_ID and ASW_SECRET_ACCESS_KEY in privately. They are not meant to share or let public known for prevention of data breaches and landed to the wrong hands (hackers)

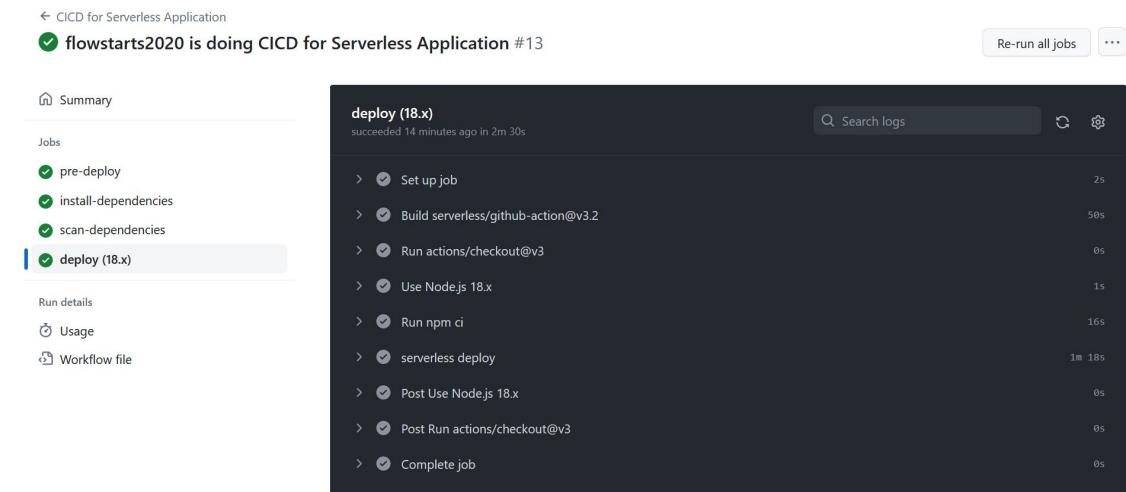


The screenshot shows the GitHub repository settings page for 'flowstarts2020/SaleSpheres'. The URL is https://github.com/flowstarts2020/SaleSpheres/settings/secrets/actions. The left sidebar has sections like Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables, Actions, Codespaces, Dependabot), and a 'Beta' section for Actions. The 'Secrets and variables' section is currently selected. The main area shows 'Environment secrets' (Manage environments) and 'Repository secrets'. Under Repository secrets, there are two entries: 'AWS_ACCESS_KEY_ID' and 'AWS_SECRET_ACCESS_KEY', both updated 13 hours ago. Each entry has edit and delete icons.

Secret	Value	Last Updated	Actions
AWS_ACCESS_KEY_ID	(hidden)	Updated 13 hours ago	
AWS_SECRET_ACCESS_KEY	(hidden)	Updated 13 hours ago	

Step 8:

Push changes
to GitHub to
start the
workflow



- Commit changes locally and push it to GitHub. Navigate the repo on GitHub, click on the **Actions** tab to see the workflows. Confirm all tests passed and without errors

Step 9:
Access AWS
Console to check
the Lambda
application (on
salesphere-dev-api
and
**SalesSphere-dev-er
ror**)

aws Services Search [Alt+S] X

EC2 S3 Elastic Beanstalk VPC RDS IAM DynamoDB

AWS Lambda X Functions (32) Last fetched 30 seconds ago Actions Create function

Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
SenseDeepWatcher	SenseDeep Alarm Watcher	Zip	14.x	2 days ago
assignment-3-12-taufiq-dev-api	-	Zip	Node.js 18.x	7 days ago
SangeethaRollingDice	-	Zip	Node.js 12.x	12 days ago
Bing_Lambda	-	Zip	Python 3.10	last month
liz-lambda-hello-world	A starter AWS Lambda function.	Zip	Python 3.7	3 months ago
liau-module-3-14-assignment-dev-api	-	Zip	Node.js 18.x	8 days ago
SalesSphere-dev-error	-	Zip	Node.js 18.x	13 hours ago
amplify-login-create-auth-challenge-5ccb4bbb	-	Zip	Node.js 12.x	2 years ago
salesphere-dev-api	-	Zip	Node.js 18.x	1 minute ago

The screenshot shows the AWS Lambda Function Overview page for a function named "salesphere-dev-api". The function has no layers. It is triggered by an API Gateway. The last modification was 1 minute ago. The ARN is arn:aws:lambda:ap-southeast-1:255945442255:function:salesphere-dev-api. The application is salesphere-dev. The function URL is provided. The "Test" tab is highlighted with a red box. Other tabs include Code, Monitor, Configuration, Aliases, and Versions. A "Code source" section with an "Upload from" button is at the bottom.

The screenshot shows the AWS Lambda Function Overview page for the same function. The "Test" tab is selected and highlighted with a red box. A "Test event" section is visible, with a "Save" button and a "Test" button highlighted with a red box. The rest of the interface is identical to the first screenshot.

Step 10:

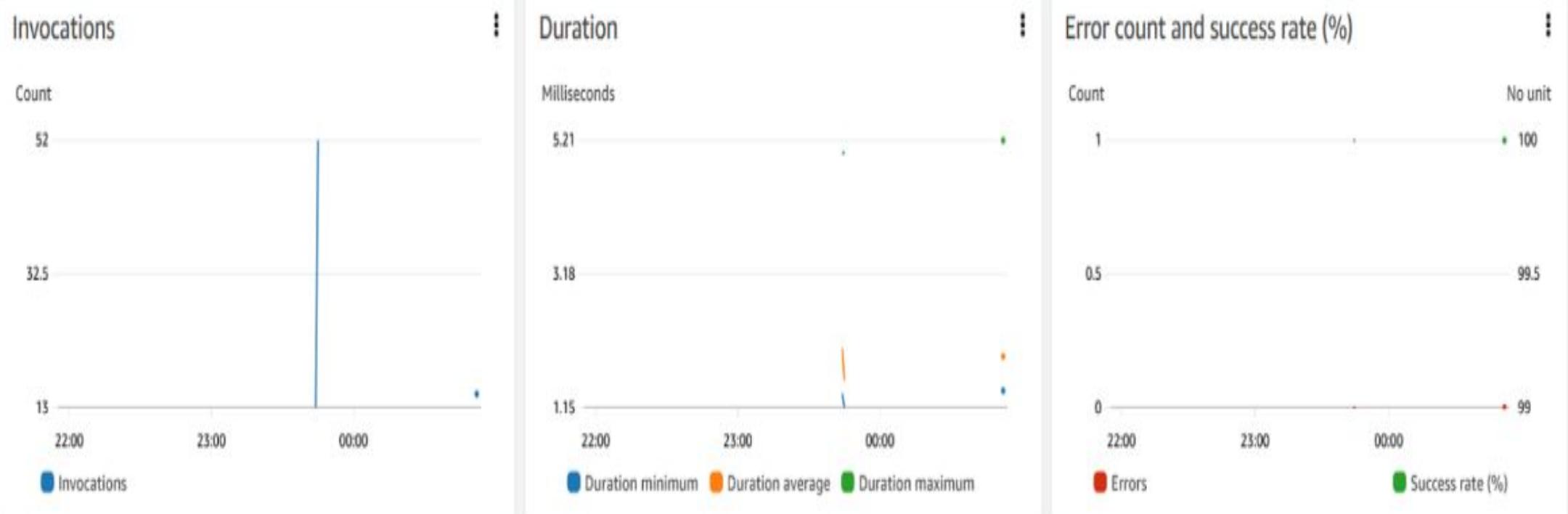
Test to Invoke the Lambda application

This screenshot shows the AWS Lambda console interface for the function `salesphere-dev-api`. At the top, there's a navigation bar with links to Services, a search bar, and various AWS services like EC2, S3, and VPC. Below the navigation bar, the function name `salesphere-dev-api` is displayed. On the right side, there are buttons for Throttle, Copy ARN, and Actions. A message box at the top left states: "This function belongs to an application. Click here to manage it." Below this, the "Function overview" section is shown, featuring a summary card for the function, an API Gateway trigger, and a "Layers" section. To the right of the summary card, there's a "Related functions" dropdown set to "Select a function". Further down, there are sections for Description, Last modified (11 minutes ago), Function ARN (with a copy icon and the value `arn:aws:lambda:ap-southeast-1:255945442255:function:salesphere-dev-api`), Application (`salesphere-dev`), and Function URL (with a "Info" link). At the bottom of the overview section, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions, with the Monitor tab currently selected and highlighted with a red border.

Step 11:
Check any readings when monitoring the Lambda application

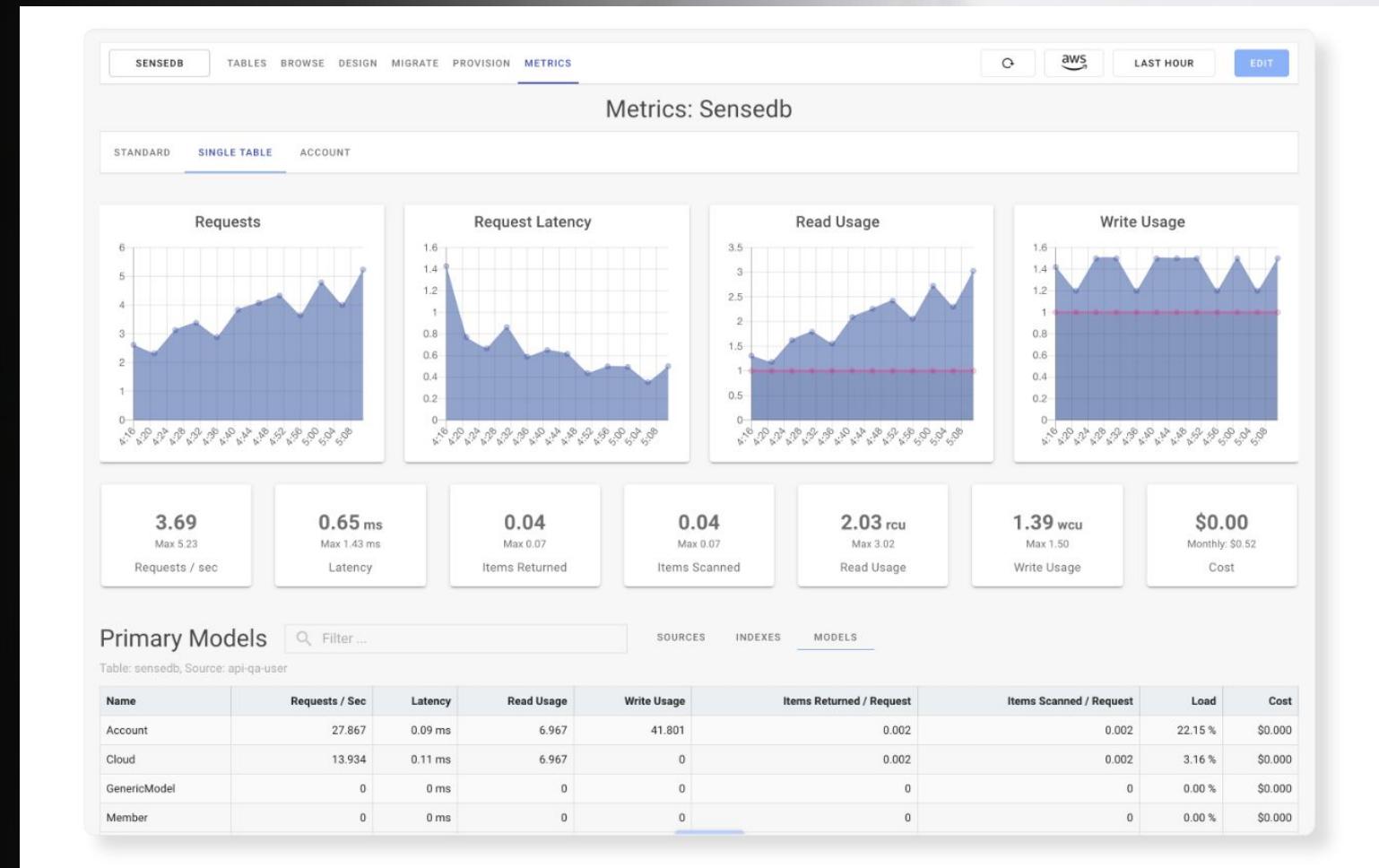
CloudWatch metrics [Info](#)Filter by [Function ▾](#)

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or \$LATEST resource, choose [Filter by](#). To view metrics for a specific function version or alias, choose [Aliases](#) or [Versions](#), select the alias or version, and then choose [Monitor](#).

[1h](#) [3h](#) [12h](#) [1d](#) [3d](#) [1w](#) [Custom](#) 

Step 12: Access SenseDeep application

- <https://www.sensedepth.com/> can be register for free.
- For this project, we will access as a free account Do go to this link <https://www.sensedepth.com/doc/kb/starting/setup.html> on how to register and setup



Application logging

- Access and login SenseDeep application
- Create the widgets of graph views on SUM and Average for the related relevant monitoring Lambda application
- illustrating one screenshot example of a SUM widget on salesphere-dev-api

SenseDeep

Sample Syslog
Syslog All Events / sample /

USER@EXAMPLE.COM HELP

Apr 20, 22:48

Timestamp	Hostname	App	Pid	Message
Apr 20, 20:01:28	google.com.au	shopping	972	Morbi quis tortor id nulla ultrices aliquet. Maecenas leo odio, condimentum id, luctu...
Apr 20, 21:24:48	ow.ly	shopping	973	Nam congue, risus semper porta volutpat, quam pede lobortis ligula, sit amet eleifen...
Apr 20, 22:48:08	etsy.com	shopping	974	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus.

app: shopping
date: 2019-03-01T08:46:40Z
hostname: etsy.com
message: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus.
msgId: ID974
pid: 974
priority: 1
timestamp: Apr 20, 22:48:08
version: 2

Apr 21, 0:11:28	economist.com	shopping	975	Nulla neque libero, convallis eget, eleifend luctus, ultricies eu, nibh.
Apr 21, 1:34:48	europa.eu	shopping	976	In hac habitasse platea dictumst. Etiam faucibus cursus urna.
Apr 21, 2:58:08	dell.com	shopping	977	Cras pellentesque volutpat dui. Maecenas tristique, est et tempus semper, est quam ...
Apr 21, 4:21:28	cargocollective.com	shopping	978	Nulla ut erat id mauris vulputate elementum.
Apr 21, 5:44:48	oracle.com	shopping	979	In hac habitasse platea dictumst. Aliquam augue quam, sollicitudin vitae, consecut...

FEEDBACK



Step 13:

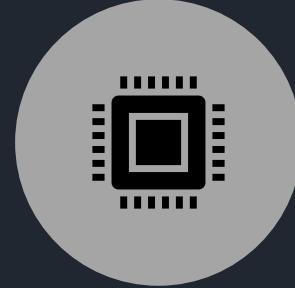
Design our own graphs on various metrics and various statistic types by extracting data from AWS Lambda.

The screenshot shows the SenseDeep interface. On the left, a sidebar lists various tools: Dashboard, Applications, Lambda Studio, Dynamodb Studio, Logs, Event Bridge, Alarms, Alerts, and Account. A trial message "Trial expires in 12 days" is visible at the top. The main area displays a chart titled "SalesSphere Invocations" for the metric "salesphere-dev-api". The chart shows a single data point at approximately 8:52 with a value of about 14.5. To the right, a modal window titled "Select widget" is open, allowing configuration for a new metric. The "Widget Configuration" section has "Graph Metric" selected. Under "Data Source", "Cloud" is set to "SalesSphere" and "Metric Namespace" is set to "AWS/Lambda". The "Metric" dropdown is set to "Invocations" and the "Statistic" dropdown is set to "Sum". In the "Resource Dimensions" section, a dimension "FunctionName=salesphere-dev-api" is listed. At the bottom of the modal are "SAVE", "CANCEL", and "DELETE" buttons.

Application Monitoring Dashboard



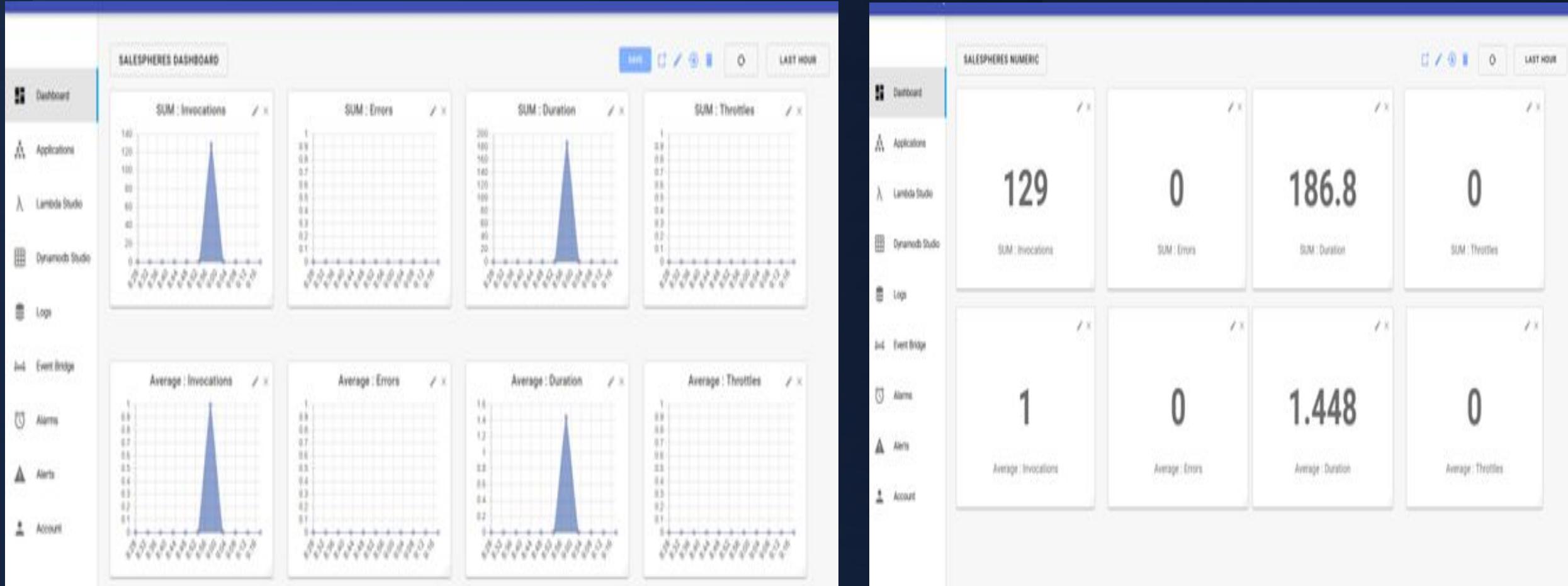
Completion on ready to view
Centralize Dashboard



Next is the illustration of a graph
widget based on hourly data of
Sum and Average from
salesphere-dev-api and
SalesSphere-dev-error

Step 14: Completion on ready to view centralize Dashboard

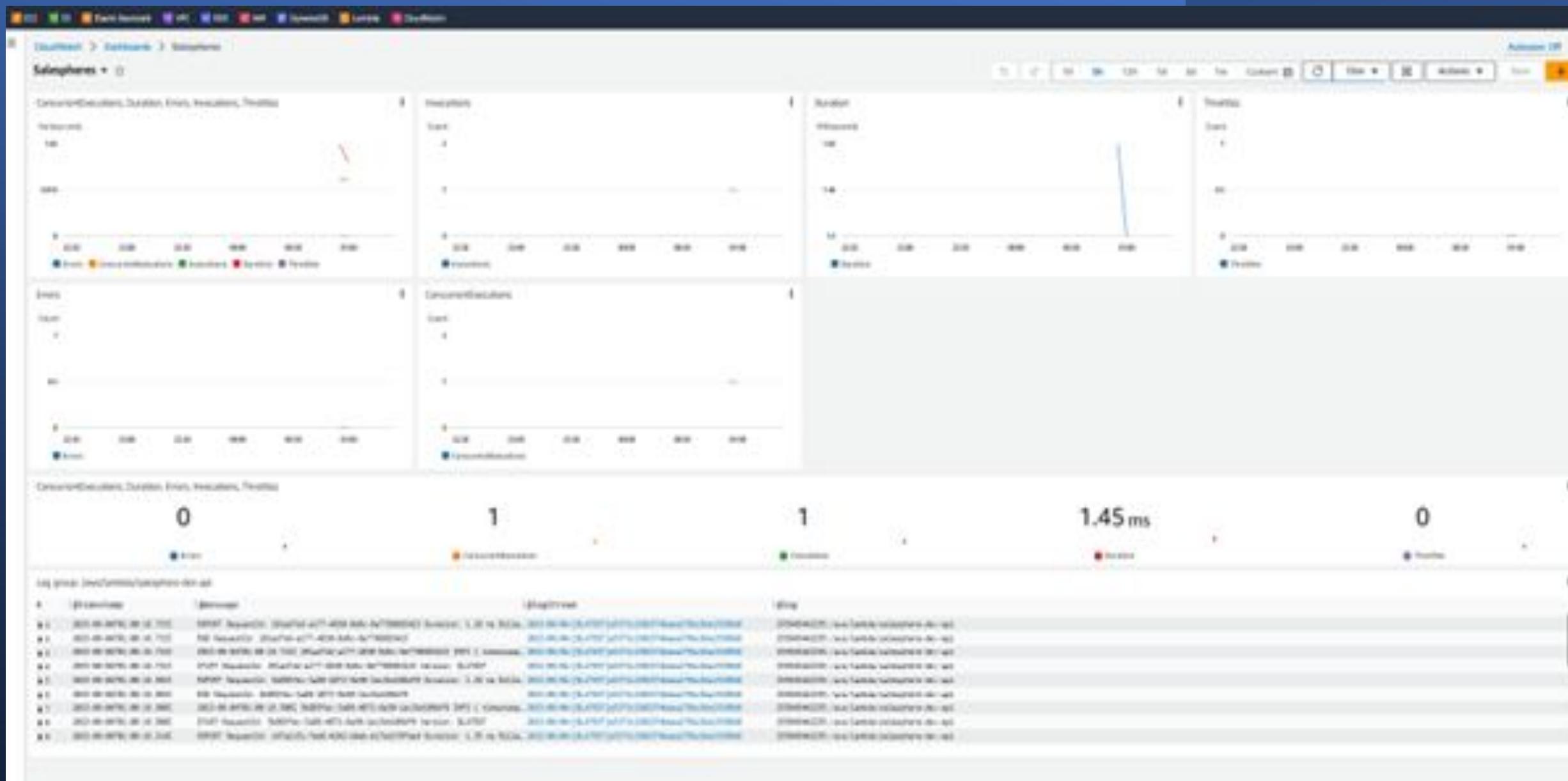
This is the illustration of a detail widget based on SUM and Average hourly on salesphere-dev-api and SalesSphere-dev-error.
SALESPHERES DASHBOARD



FAQ

- What if Sensedeep is not available or outage?
- We have the widgets dashboard created on CloudWatch updating based on sale sphere-dev-api. The overall factors on Concurrent Execution, Duration, Errors, Invocations, Throttles and by each factor. We have an overall factor in numbers. Below is an example





Invocations, Duration, Errors , ConcurrentExecution, Throttles

Invocations -The number of times that your function code is invoked, including successful invocations and invocations that result in a function error.

Duration - The amount of time that your function code spends processing an event. The billed duration for an invocation is the value of Duration rounded up to the nearest millisecond.

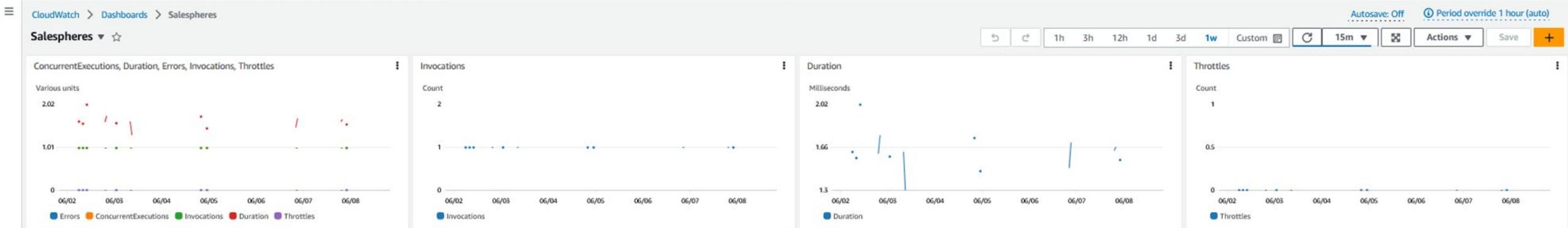
Errors - The number of invocations that result in a function error.

ConcurrentExecutions - The number of function instances that are processing events. If this number reaches your concurrent executions quota for the Region, or the reserved concurrency limit on the function, then Lambda throttles additional invocation requests.

Throttles - The number of invocation requests that are throttled.

Would like to get the logs in real-time.
How would I access?

- You can access on the define Dashboard name



Log group: /aws/lambda/salesphere-dev-api

#	@timestamp	@message	@logStream	@log
1	2023-06-08T14:11:20.936Z	END RequestId: 170bcf86-b285-4d59-abc2-0b3ea530beb1	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
2	2023-06-08T14:11:20.936Z	REPORT RequestId: 170bcf86-b285-4d59-abc2-0b3ea530beb1 Duration: 1.26 ms Bille..	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
3	2023-06-08T14:11:20.935Z	2023-06-08T14:11:20.935Z 170bcf86-b285-4d59-abc2-0b3ea530beb1 INFO { timestamp: 2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
4	2023-06-08T14:11:20.934Z	START RequestId: 170bcf86-b285-4d59-abc2-0b3ea530beb1 Version: \$LATEST	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
5	2023-06-08T14:11:20.722Z	END RequestId: ca61f2b5-bfdc-45da-aefc-9e062fc1feb9	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
6	2023-06-08T14:11:20.722Z	REPORT RequestId: ca61f2b5-bfdc-45da-aefc-9e062fc1feb9 Duration: 1.19 ms Bille..	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
7	2023-06-08T14:11:20.728Z	2023-06-08T14:11:20.728Z ca61f2b5-bfdc-45da-aefc-9e062fc1feb9 INFO { timestamp: 2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
8	2023-06-08T14:11:20.728Z	START RequestId: ca61f2b5-bfdc-45da-aefc-9e062fc1feb9 Version: \$LATEST	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api
9	2023-06-08T14:11:20.519Z	END RequestId: 03f5a2bf-feb8-4ce8-a17f-615fb416ec21	2023/06/08/[\${LATEST}]a59256c9ea894e3b82874ba6c7945145	255945442255:/aws/lambda/salesphere-dev-api

Future plan?

- After a period of evaluation, we are glad to propose the positive results updates to our management. We are glad the project SaleSpheres version 1.0 ready to launch in production
- Next plan of version 2.0 will be on the following,
 - a) Area of API Gateway integration
 - b) Consideration the enhancement the features on the SenseDeep application with logs
 - c) Monitoring on other services like EC2.





Thank You!