

Overview SDLC

written by junxian428

25/7/2023

Include

- Requirement
- Planning
- Implementation
- Testing
- Deploy

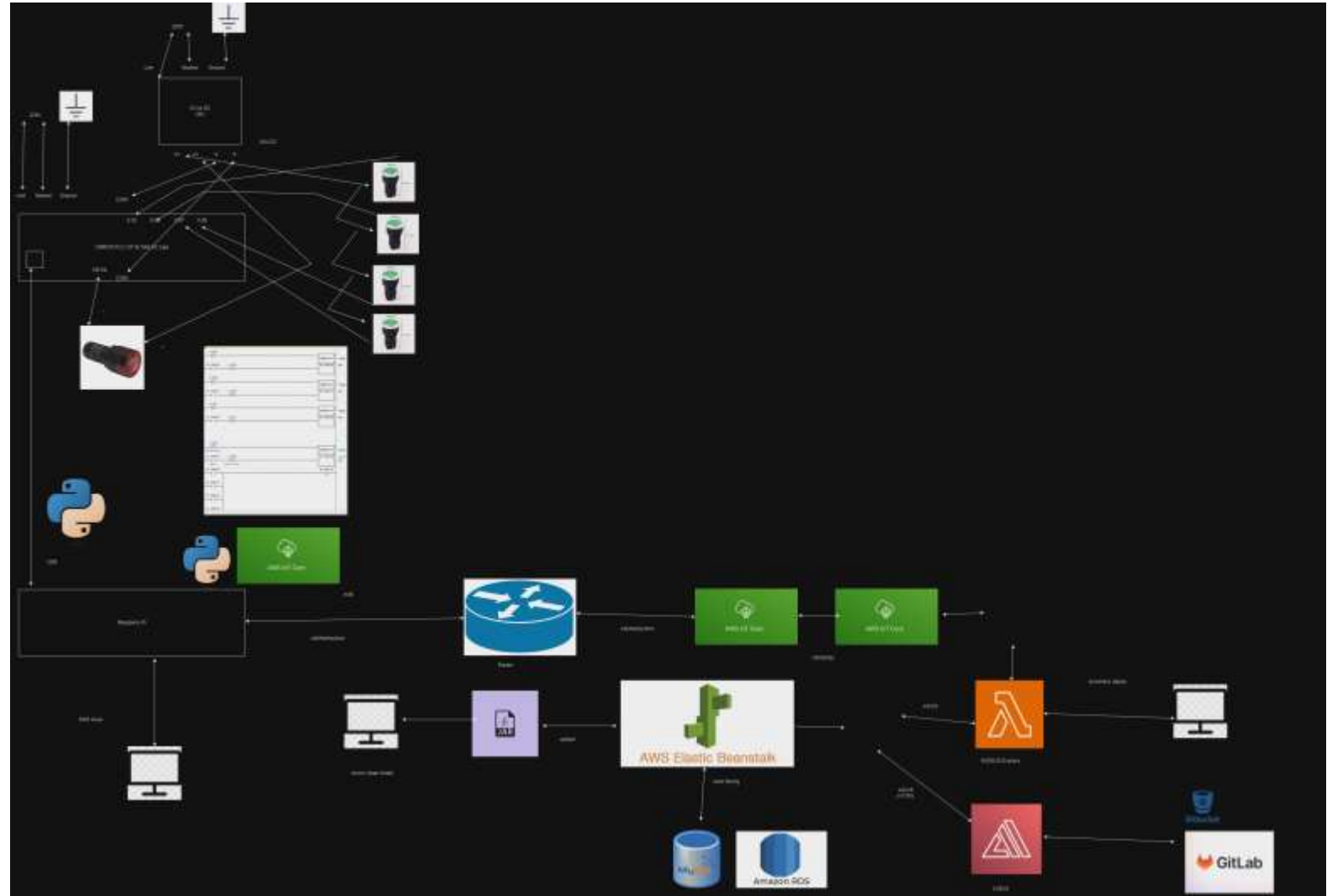
FE/BE architecture and
Event-driven architecture
Serverless

*Microservices

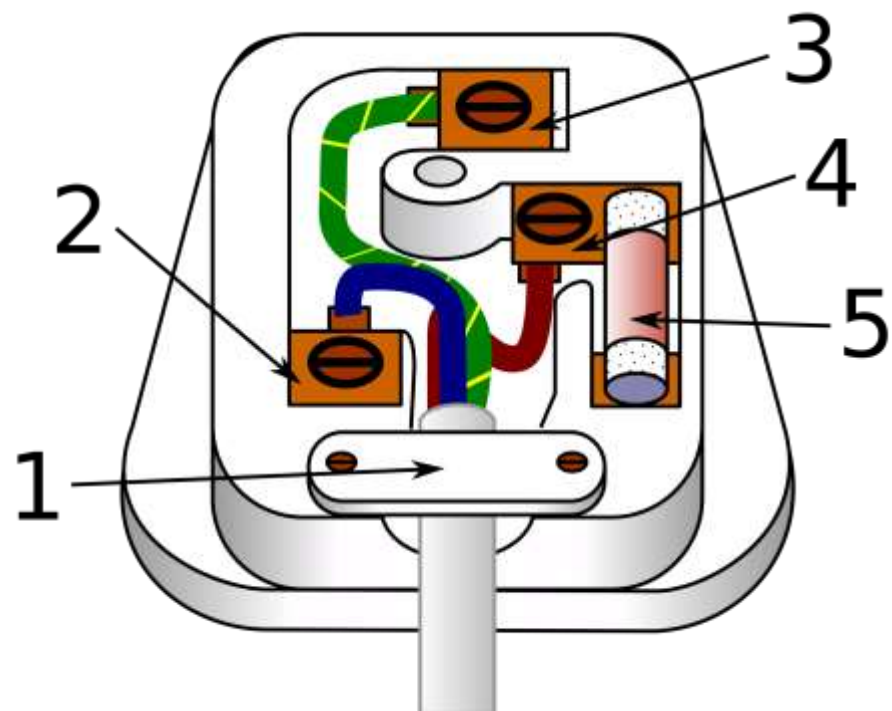
* Java Spring, Python, Qt, VueJS, MySQL

* PLC, Raspberry Pi

*AWS

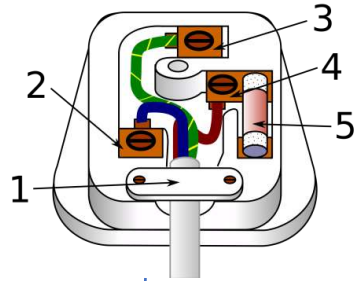


Wiring

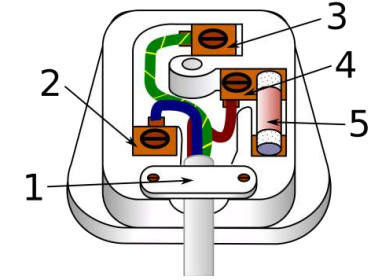
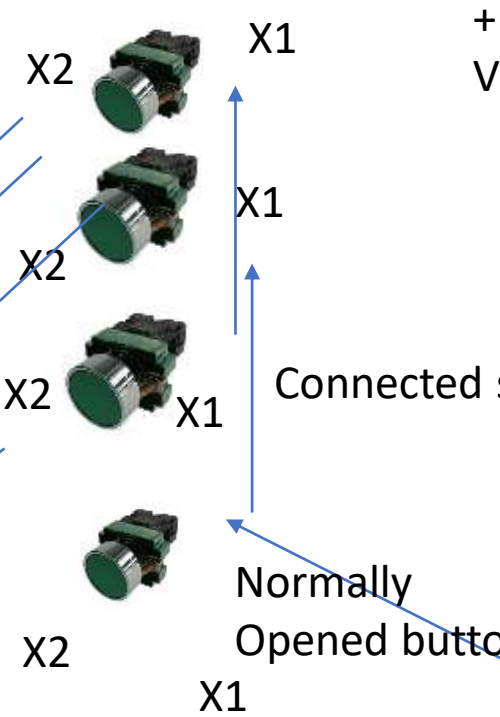


Product Size:





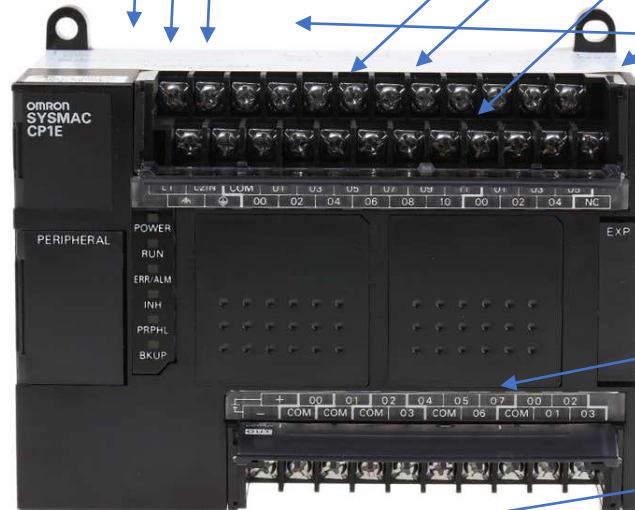
Input	Type
0.03	Button1
0.05	Button2
0.07	Button3
0.09	Button4



Connected series

Normally
Opened button
X1

Product Size:

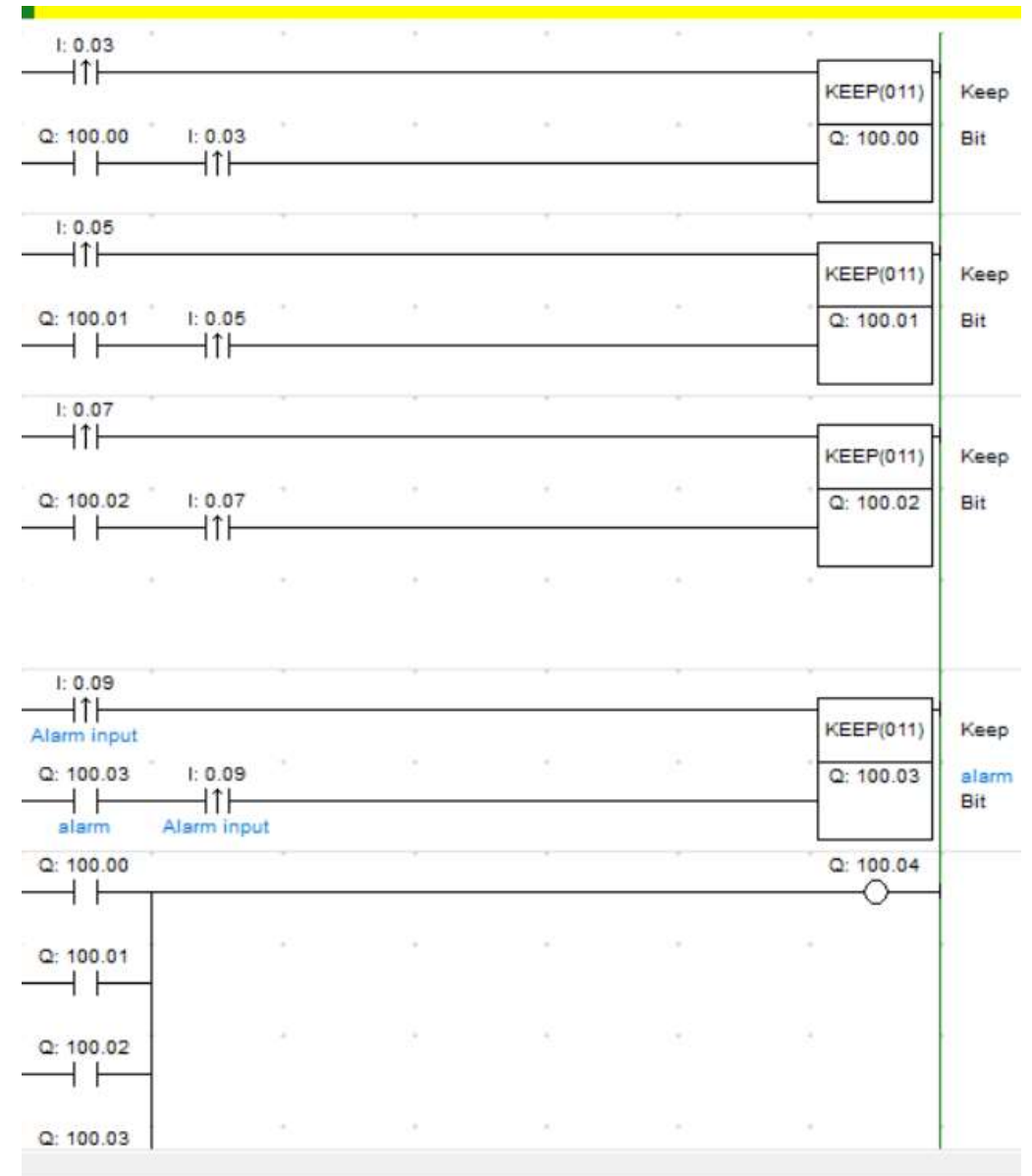


LED 24DC V with
Buzzer



Output	Type
100.04	Led with buzzer

PLC & Ladder Diagram



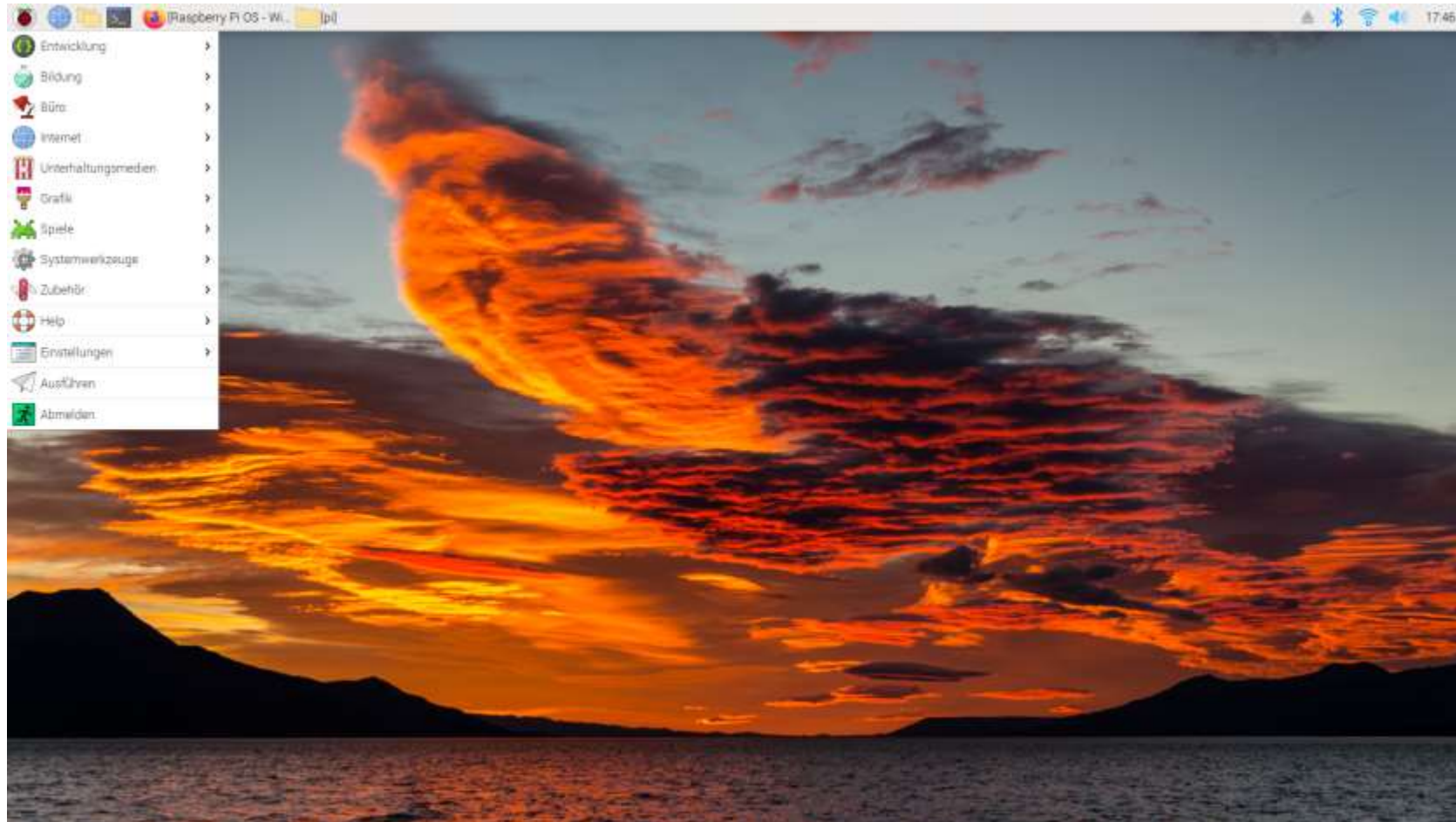
Raspberry Pi



Burn OS into SD card



Raspberry Pi OS



Qt Frontend Python



The image displays the Qt Designer application window. The central canvas shows a form titled "Print Your Value" with a text input field and a "Print" button. The left sidebar contains a "Widget Box" with various Qt widgets and layouts. The right sidebar shows the "Object Inspector" and "Property Editor".

Object Inspector:

Object	Class
Form	QWidget
label	QLabel
lineEdit	QLineEdit
pushButton	QPushButton

Property Editor:

Property	Value
objectName	Form
enabled	True
geometry	(0, 0, 382 x 174)
x	0
y	0
width	382
height	174
sizePolicy	(Preferred, Preferred, 0, 0)
minimumSize	0x0

Action Editor:

Name	Used	Text	Shortcut
------	------	------	----------

LearnDataAnalysis.org

Python C-command to PLC

- Sudo apt-get install code (visual studio code)
- Sample
- https://github.com/junxian428/DesktopApp_PLC_Raspberry

End of hardware

Begin of software

Raspberry Pi to AWS IoT Thing

- Install AWS IoT SDK, zip file and unzip then run `./start.sh`
- Modify code in `pubsub.py`

To Change Code (in `aws-iot-device-sdk-python-v2/samples/pubsub.py`)

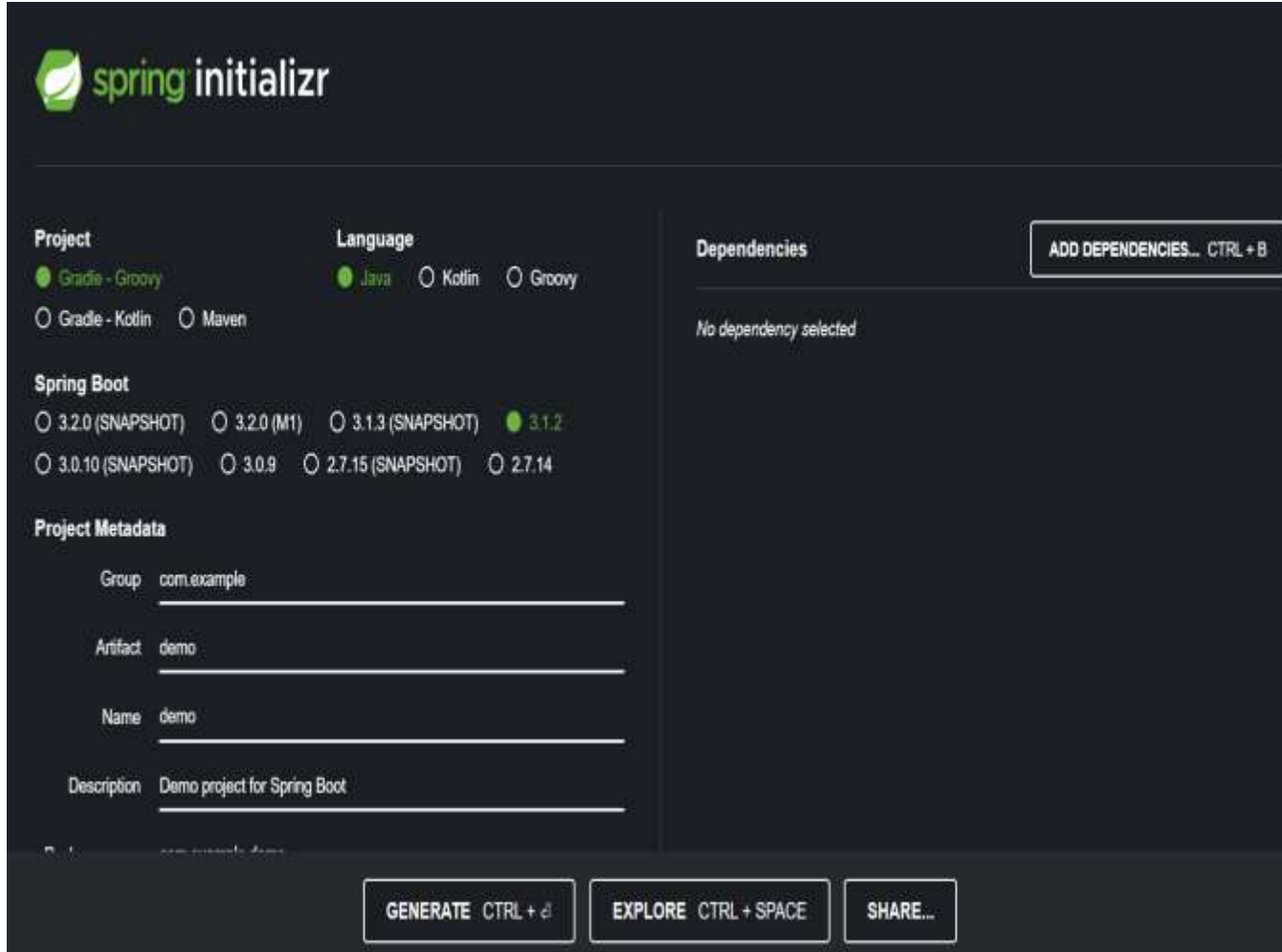
Change the

Message = “your message”

- In order to run,
`./start.sh`



Backend First Methodology



The image shows the Spring Initializr web interface, a tool for generating Spring project skeletons. The interface is dark-themed and contains several sections for configuring a project:

- Project:** Radio buttons for ☒ Gradle - Groovy, ☐ Gradle - Kotlin, and ☐ Maven.
- Language:** Radio buttons for ☒ Java, ☐ Kotlin, and ☐ Groovy.
- Spring Boot:** Radio buttons for various versions: ☐ 3.2.0 (SNAPSHOT), ☐ 3.2.0 (M1), ☐ 3.1.3 (SNAPSHOT), ☒ 3.1.2, ☐ 3.0.10 (SNAPSHOT), ☐ 3.0.9, ☐ 2.7.15 (SNAPSHOT), and ☐ 2.7.14.
- Project Metadata:** Four text input fields: Group (com.example), Artifact (demo), Name (demo), and Description (Demo project for Spring Boot).
- Dependencies:** A section with the text "No dependency selected" and a button "ADD DEPENDENCIES... CTRL + B".
- Buttons:** At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

JWT

https://github.com/junxian428/Java_Spring_JWT

CRUD

<https://start.spring.io/>

https://github.com/junxian428/Spring_RESTAPI_CRUD_STOCK

Database RDS (MySQL)

spring:

datasource:

url: jdbc:mysql://localhost:3306/jwt

username: root

password:

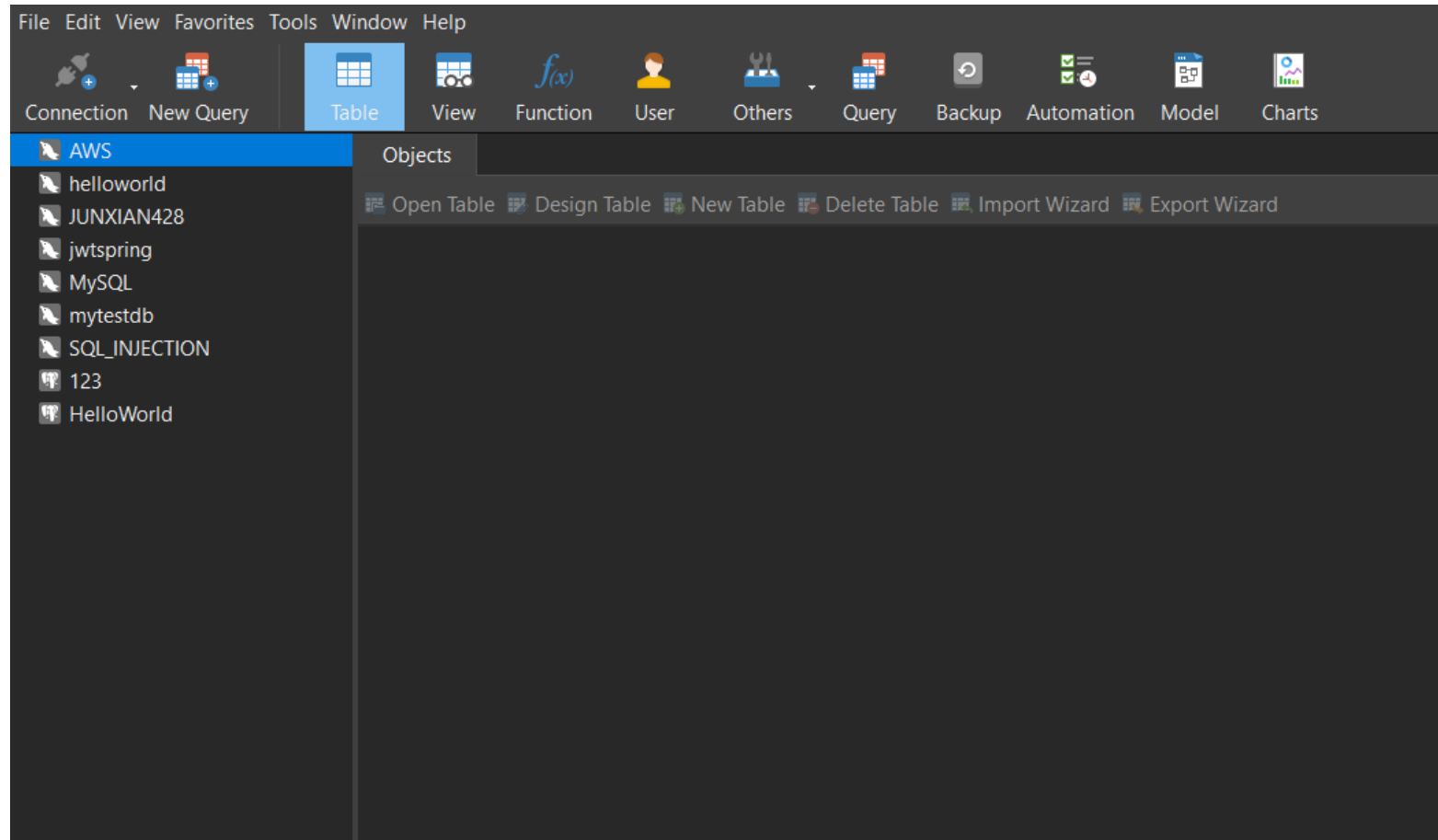
driver-class-name: com.mysql.cj.jdbc.Driver

jpa:

hibernate:

* Change URL to AWS

Navicat connects database



AWS Elastic BeanStalk

- ./mvnw clean install
- Create jar file then upload

A screenshot of an IDE (Visual Studio Code) showing a Java project named 'SPRING_RESTAPI_CRUD_STOCK-MAIN'. The Explorer view on the left shows the project structure, including 'src/main/java' and 'target' directories. The main editor displays the 'StockController.java' file, which contains REST API endpoints for listing and retrieving stock information. The bottom panel shows the 'TERMINAL' output, which includes build logs for 'spring-boot-maven-plugin' and 'maven-install-plugin', indicating a successful build and installation of the application jar file.

```
File Edit Selection View Go Run Terminal Help
StockController.java - Spring_RESTAPI_CRUD_STOCK-main - Visual Studio Code

EXPLORER
SPRING_RESTAPI_CRUD_STOCK-MAIN
  .mvnw
  src
  main
    java \D210044\Java2\StockManagement\RESTAPI4CRUD
      Restapi4CrudApplication.java
      Stock.java
      StockController.java
      StockRepository.java
      StockService.java
    resources
  test \java \D210044\Java2\StockManagement\RESTAPI4CRUD
    Restapi4CrudApplicationTests.java
  target
    classes
    generated-sources
    generated-test-sources
    maven-archiver
    maven-status
    surefire-reports
    test-classes
    RESTAPI4CRUD-0.0.1-SNAPSHOT.jar
    RESTAPI4CRUD-0.0.1-SNAPSHOT.jar.original
    .gitignore
    mvnw
    mvnw.cmd
    pom.xml
    README.md

  TIMELINE
  JAVA PROJECTS
  MAVEN

String responseBody -
// Your
String responseBody = "Success! Request"
return ResponseEntity.ok(responseBody);
}

@GetMapping("/stocks")
public List<Stock> list() {
    return stockService.listAll();
}

@GetMapping("/stocks/{id}")
public ResponseEntity<Stock> get(@PathVariable
    try{
        Stock stock = stockService.get(id);
        return new ResponseEntity<Stock>(sto
    }catch(NoSuchElementException e){
        return new ResponseEntity<Stock>(utt

[INFO] --- spring-boot-maven-plugin:2.6.0:repackage (repack
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-install-plugin:2.5.2:install (default-inst
[INFO] Installing D:\Project\WUEJSSPRING_CRUD_JWT\Spring_R
.0.1-SNAPSHOT.jar to C:\Users\junxian428\.m2\repository\D2
SNAPSHOT.jar
[INFO] Installing D:\Project\WUEJSSPRING_CRUD_JWT\Spring_R
unxian428\.m2\repository\D210044\Java2\StockManagement\RES
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 20.003 s
[INFO] Finished at: 2023-07-24T09:39:30+08:00
[INFO]
PS D:\Project\WUEJSSPRING_CRUD_JWT\Spring_RESTAPI_CRUD_STOCK
* History restored
PS D:\Project\WUEJSSPRING_CRUD_JWT\Spring_RESTAPI_CRUD_STOCK
```

Environments (2) [Info](#)

	Environment name ▲	Health ▼	Applicatio... ▼	Platform ▼
<input type="radio"/>	JavaSpring-env	✔ Ok	JavaSpring	Corretto 17 ru...
<input type="radio"/>	JavaSpringCRUD-env	✔ Ok	JavaSpringCR...	Corretto 17 ru...

Upload and deploy

 To deploy a previous version, go to the [Application versions page](#)

Upload application

 **Choose file**

File must be less than 500MB max file size

Version label

Unique name for this version of your application code.

Current number of EC2 instances: 1

Cancel

Deploy

Choose jar file found in target folder



Set API gateway for Elastic Bean Stalk in order to get HTTPS

The screenshot displays the Amazon API Gateway console interface. The breadcrumb navigation at the top reads: APIs > JavaSpringCRUD (huf9z3dpx4) > Resources > / (ejfwjuvdjb).

The left-hand navigation pane shows the following structure:

- Resources
 - / (expanded)
 - GET
 - /stocks
 - GET
 - POST
 - /id
 - DELETE
 - GET
 - OPTIONS
 - POST

The main content area is titled '/ Methods' and displays the configuration for the GET method:

- GET** (method icon)
- Authorization:** None
- API Key:** Not required

The right-hand pane shows the 'dev Stage Editor' for the 'dev' stage. The 'Invoke URL' is `https://huf9z3dpx4.execute-api.us-east-1.amazonaws.com/dev`. The 'Settings' tab is active, showing the following configurations:

- Cache Settings:** Enable API cache ☐
- Default Method Throttling:** Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current settings are 10000 requests per second with a burst of 5000 requests. [Read more about API Gateway throttling](#).
- Enable throttling:** ☒ **0**
- Rate:** 10000 requests per second
- Burst:** 5000 requests
- Web Application Firewall (WAF):** [Learn more](#). Select the Web ACL to be applied to this stage. **Web ACL:** None [Create Web ACL](#)
- Client Certificate:**

Until now

- You have already deployed backend + database + Hardware
- Now deploy middleware and frontend as well as the IoT Rules and destination confirmation

AWS IoT Core set rules and destination

AWS IoT > Message routing > Rules

Rules (2) [Info](#)

Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.



Activate

Deactivate

Edit

Delete

Create rule

Find rules

< 1 >

<input type="checkbox"/>	Name	Status	Rule topic	Created date
<input type="checkbox"/>	CRUD	Active	sdk/test/python	July 24, 2023, 14:56:14 (UTC+08:00)
<input type="checkbox"/>	HTTPSCRUD	Active	sdk/test/js	July 25, 2023, 10:01:40 (UTC+08:00)

SQL statement

Enter a SQL statement using the following: `SELECT * FROM <Topic Filter> WHERE <Condition>`. For example: `SELECT * FROM 'sdk/test/python' WHERE <Condition>`

`SELECT * FROM 'sdk/test/js'`

SQL Use 1. Column 1

Rule actions

When an event is received (or triggered) from the shadow rule is triggered by an IoT Core message. Before any additional actions that occur when messages arrive, the starting point is a database, including local functions, or sending notifications. You can add up to 10 actions.

Action 1

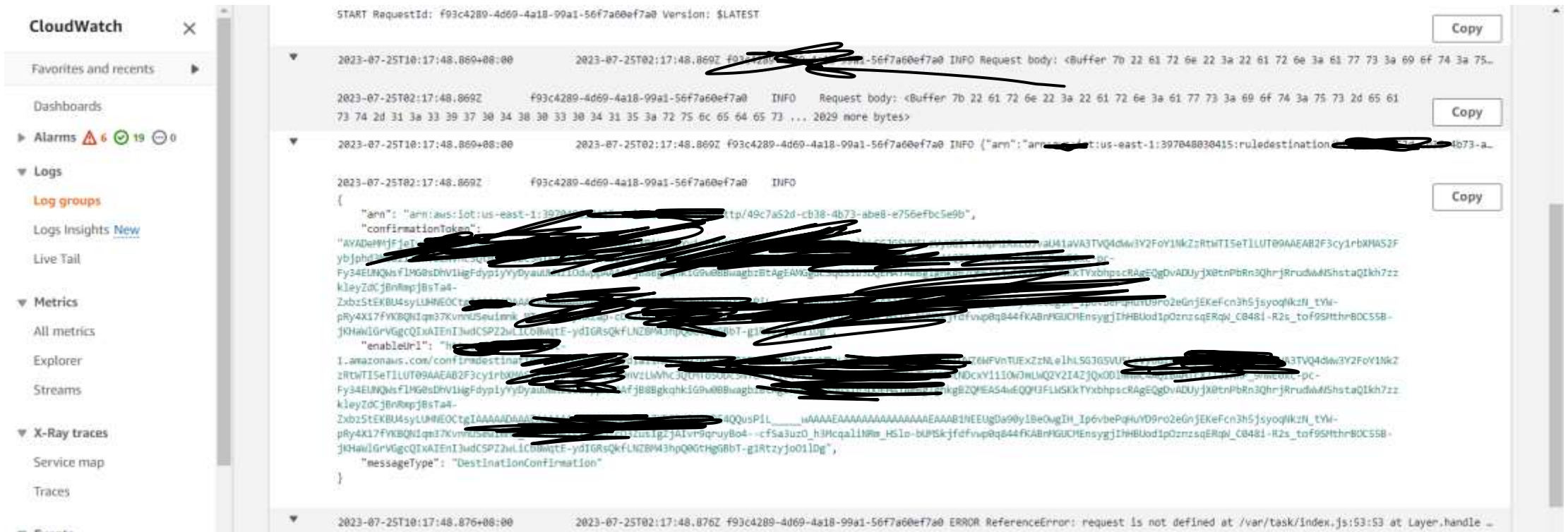
HTTP endpoint

Send a message to a destination using HTTP messages.

Remove

If your destination is not confirmed

- You are required to check the log to get the confirmation token
- By using cloudwatch



- IoT AWS Core from Raspberry Pi will send to sdk/test/python
- Create rules to redirect iot topic into sdk/test/js
- Then create rules for sdk/test/js send HTTPS request to Lambda API gateway

AWS Lambda Serverless

- `npm install -g serverless@3.31.0`



```
PS D:\Project\Serveless> serverless
```

Creating a new serverless project

? What do you want to make? (Use arrow keys)

> AWS - Node.js - Starter

AWS - Node.js - HTTP API

AWS - Node.js - Scheduled Task

AWS - Node.js - SQS Worker

AWS - Node.js - Express API

AWS - Node.js - Express API with DynamoDB

AWS - Python - Starter

AWS - Python - HTTP API

AWS - Python - Scheduled Task

AWS - Python - SQS Worker

AWS - Python - Flask API

AWS - Python - Flask API with DynamoDB

Other

const axios = require("axios");

npm i axios

This is help your
application
deployed into
AWS Lambda
and API gateway
-serverless
deploy

```
app.get("/", (req, res, next) => {
```

```
  // Replace the following URL with the API you want to fetch data from  
  const apiUrl = "";
```

```
  // Making a GET request using Axios
```

```
  axios.get(apiUrl)
```

```
    .then(response => {
```

```
      // The data from the API will be available in the 'response.data' property
```

```
      const responseData = response.data;
```

```
      console.log('Response data:', responseData);
```

```
      return res.status(200).json({
```

```
        message: responseData,
```

```
      });
```

```
    })
```

```
    .catch(error => {
```

```
      console.error('Error fetching data:', error);
```

```
      return res.status(400).json({
```

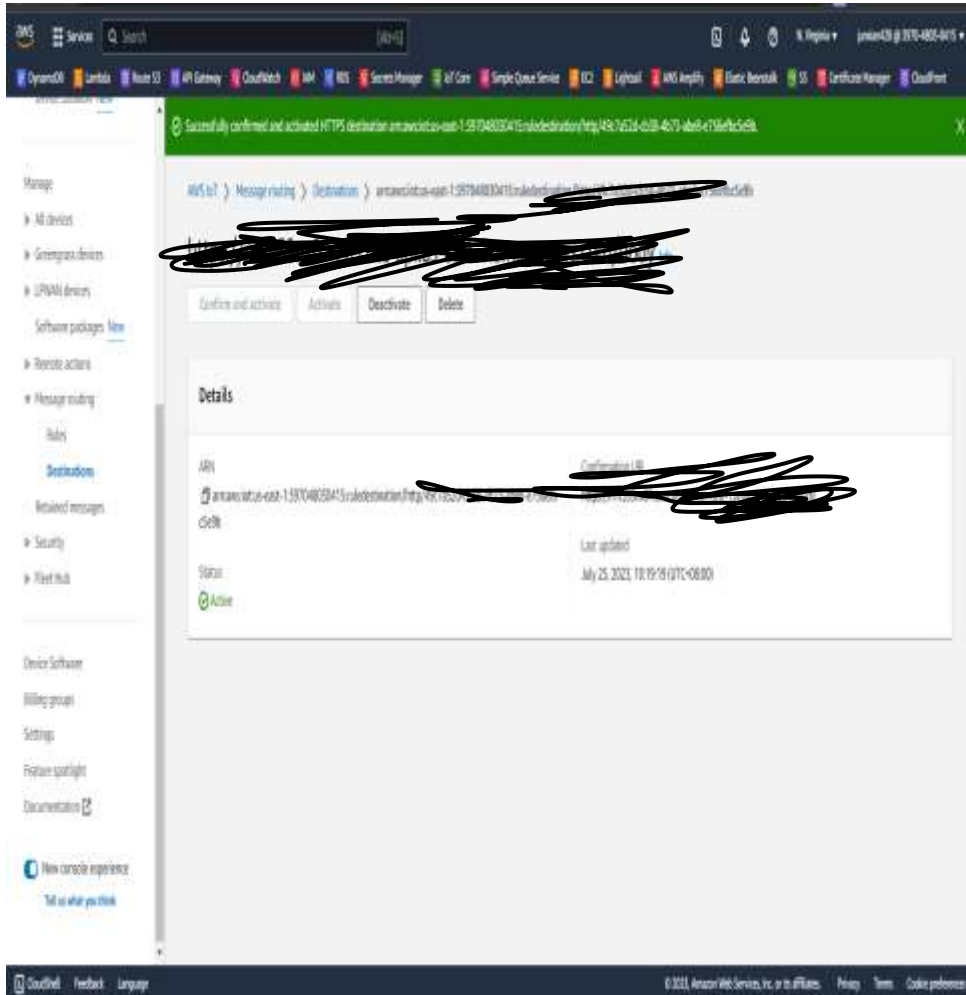
```
        message: error,
```

```
      });
```

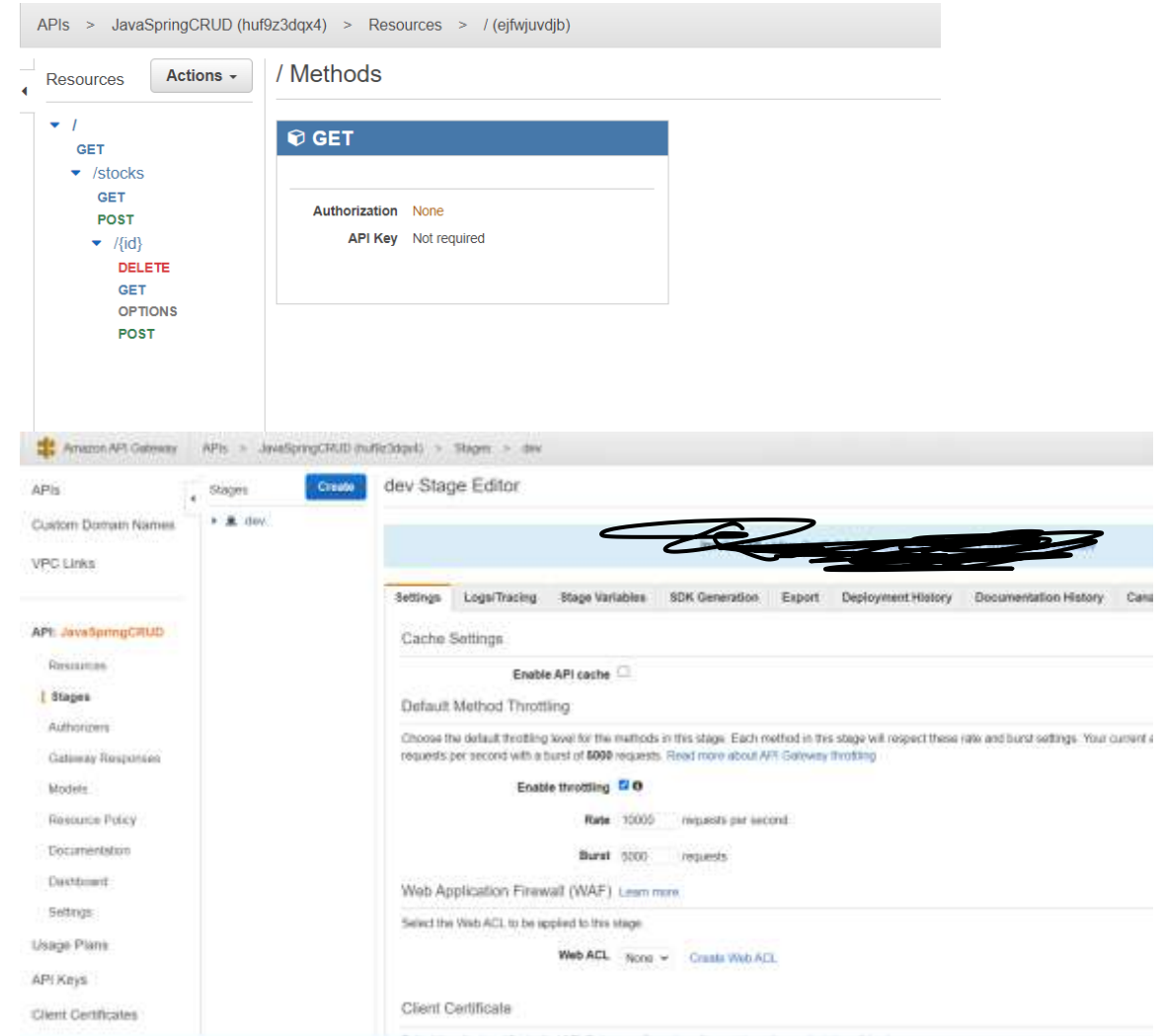
```
    });
```

```
  });
```

AWS Lambda Serverless



Your AWS Lambda Serverless endpoint should
Set the AWS API gateway for Java Spring Elastic BeanStalk



Frontend Last Methodology

- Create vuejs
- Vue create frontendproject

Choose router & vuex



Frontend Deploy (AWS Amplify)



Amplify Hosting



Host your web app

Connect your Git repository to continuously deploy your frontend and backend. Host it on a globally available CDN.



[Get started](#)

Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

☐ GitHub



☐ Bitbucket



☐ GitLab



☐ AWS CodeCommit



☐ Deploy without Git provider



[Continue](#)