COMPX341-22A Assignment 4
Quality Assurance via CI/CD on the cloud

Prepared by: Kevin Han 1521885

Table of Content

Contents

1. Overview

The purpose of the report is to document the progress of the steps to understand the importance of AWS and DevOps. The outcome of is to be able to utilise DevOps services on the cloud and apply DevOps and testing processes to deploy applications on the cloud and automate a continuous deployment pipeline. Also show that errors will not be push to repository stored on the cloud/system.

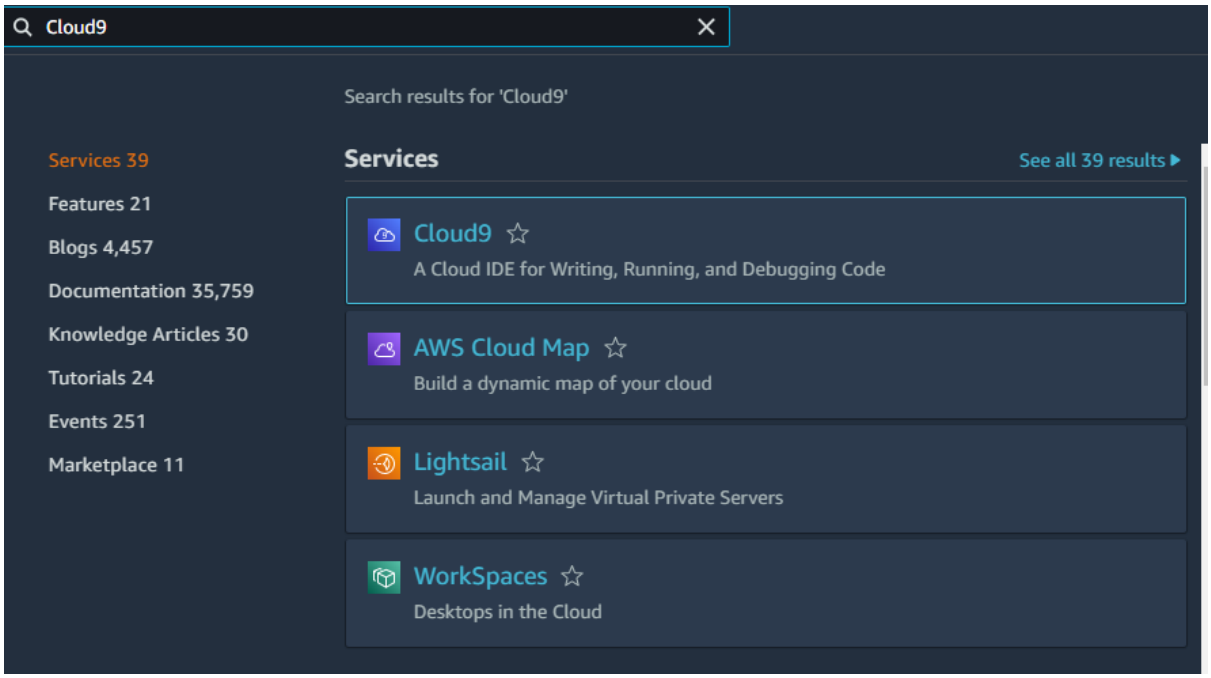
We will be using the AWS (https://awsacademy.instructure.com/) to demonstrate my understanding of the task provided.

2. Task 1: Preparing your Cloud9 IDE

The task is to initialise the environment for the Cloud9 IDE, to have access the IDE from any logged-in AWS console.

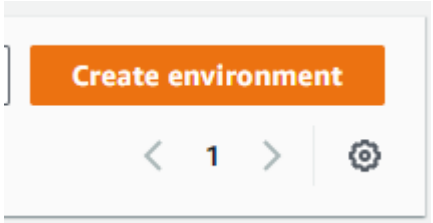
Step 1: Search for Cloud9 in the AWS Console as shown in Fig 1.

Fig 1



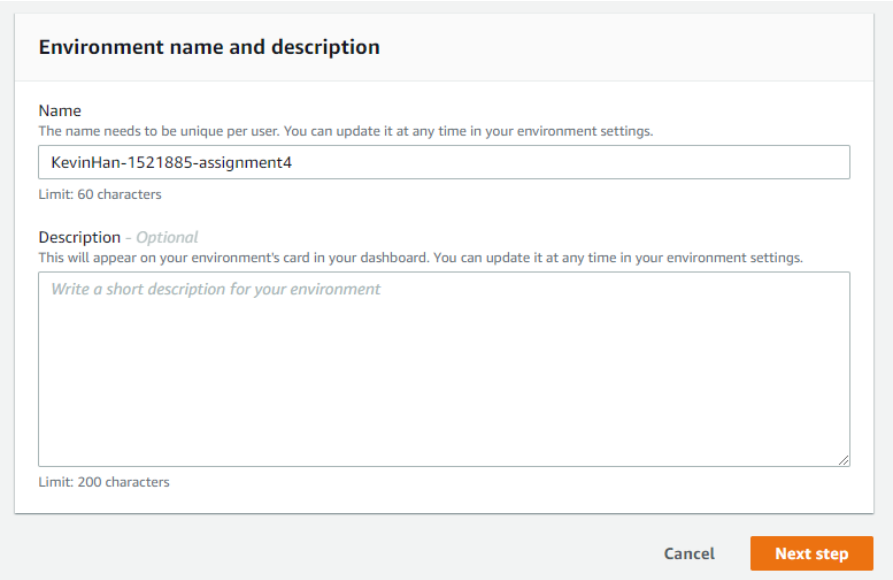
Step 2: Click on the create environment as shown in Fig 2.

Fig 2



Step 3: Set-up the environment name based on the assignment specification. The name should be like “your_name-studentID-assignment4”. It should look like Fig 3.

Fig 3



Step 4: The application will not be using any special services, so the setting can be left as default. It should look like Fig 4-5.

Fig 4

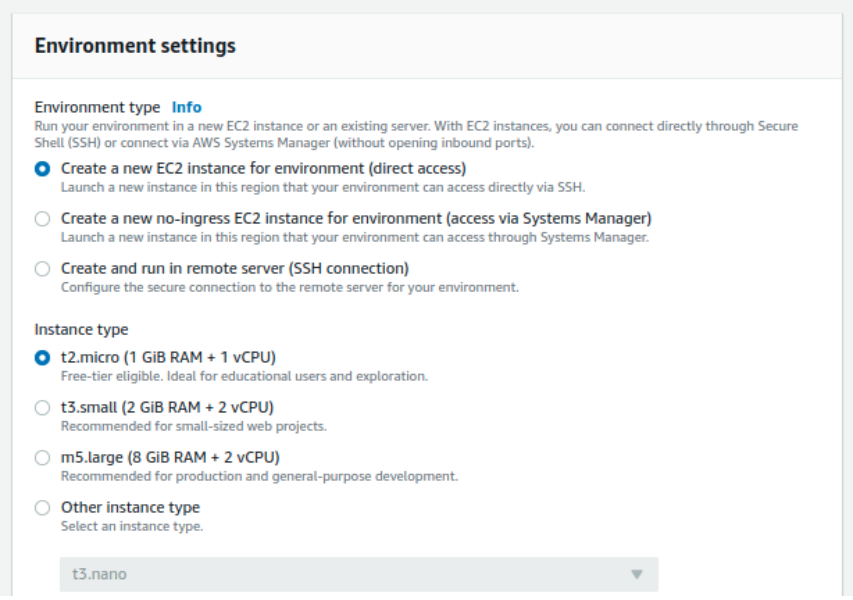
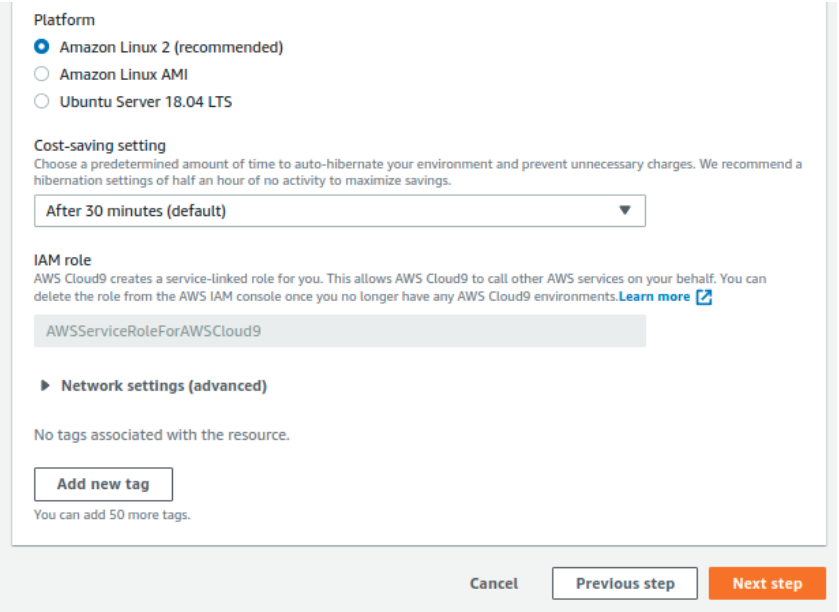
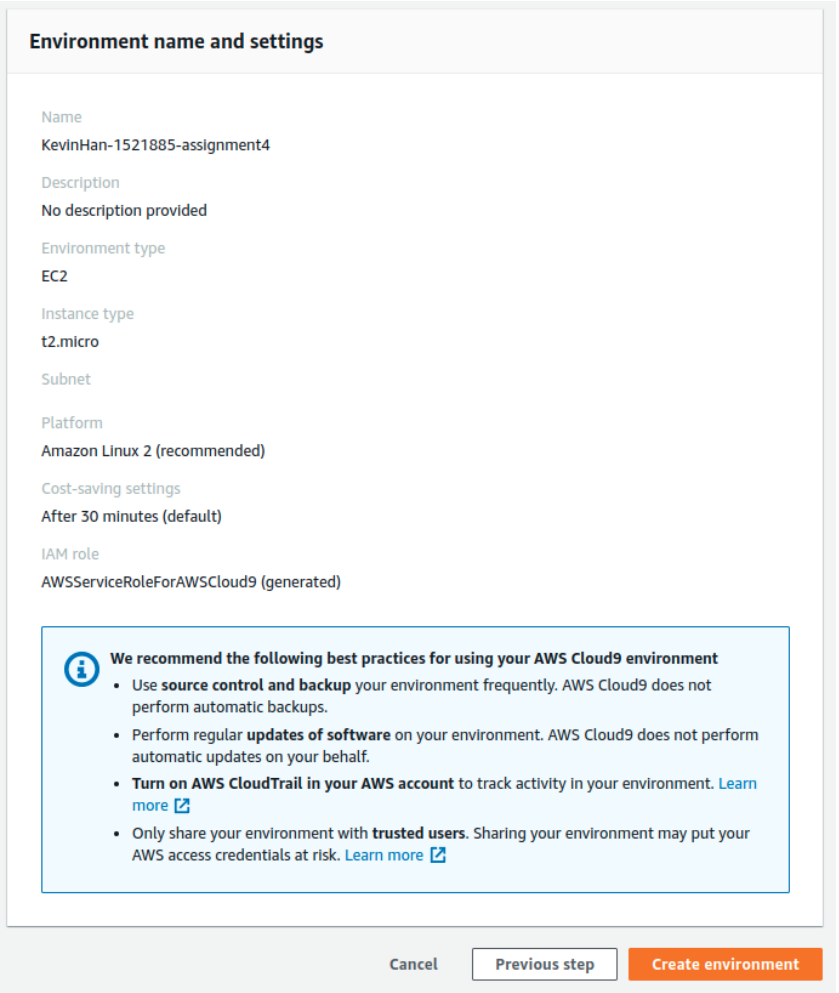


Fig 5



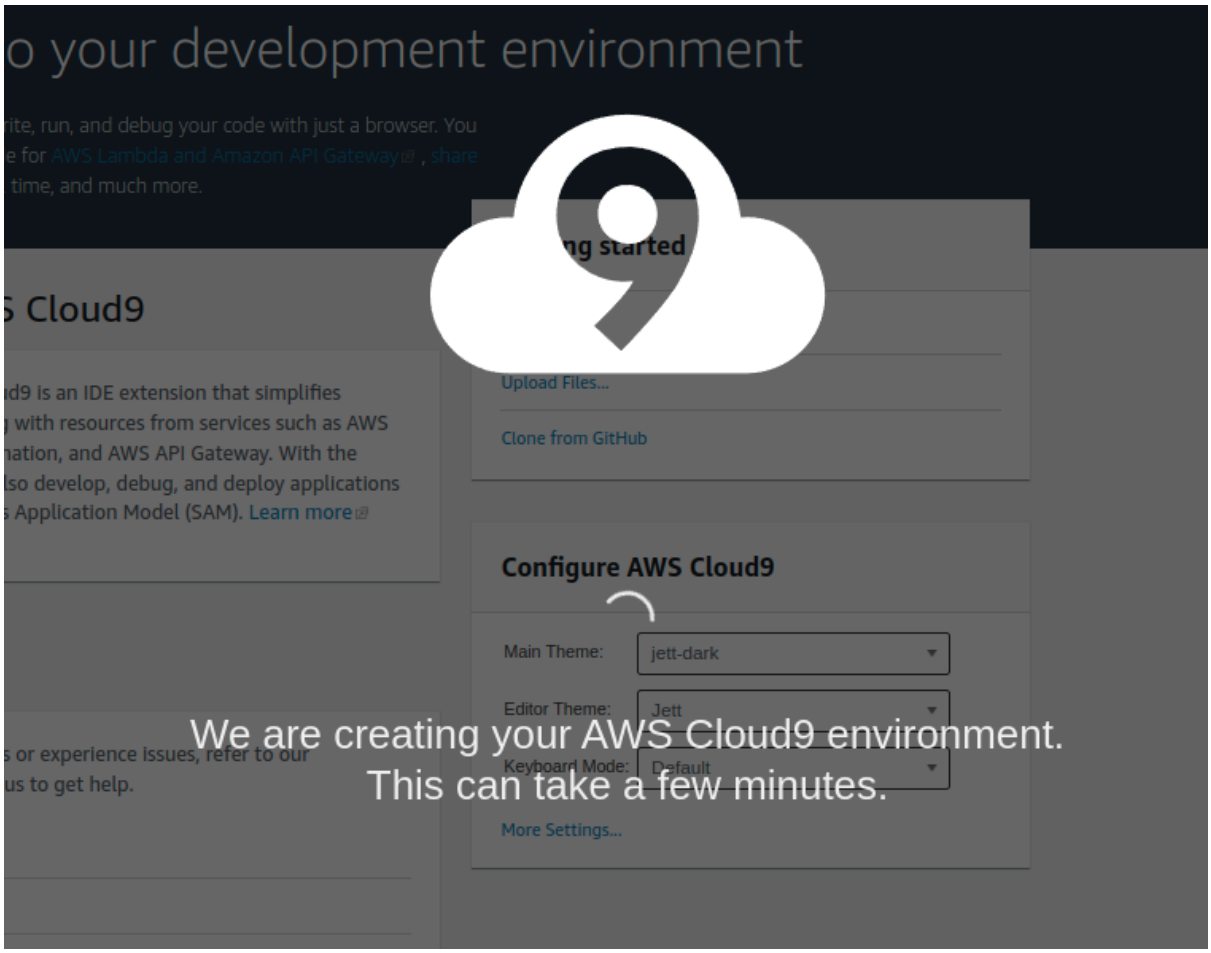
Step 5: Click on the create environment button to create the environment.

Fig 6



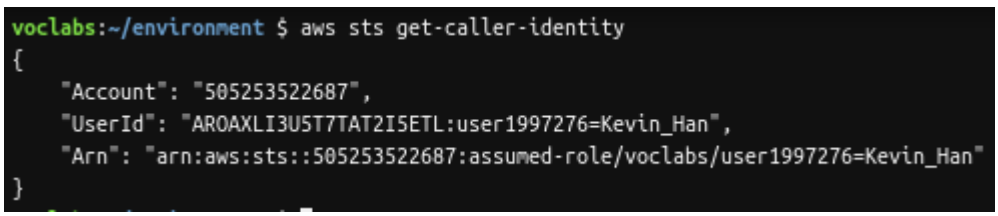
After clicking the button, it will show the screen based on the Fig 7. This shows that the AWS environment is currently being created.

Fig 7



Open the environment and type “aws sts get-caller-identity” in the console. This is to ensure that your account is signed in. An example of is shown in Fig 8.

Fig 8



3. Task 2: Initializing AWS Serverless Architecture Model

The task is to initialise the AWS Serverless Architecture Model to the latest version within the Cloud9 IDE. We will be using a preset template provided to initalise the “AWS Hello World Example”.

Step 1: Run the following commands.

wget https://cicd.serverlessworkshops.io/assets/bootstrap.sh

chmod +x bootstrap.sh

./bootstrap.sh

The following command is to get the latest version, then allowing the program to be executable, then run the bootstrap script. It should look like Fig 9-10 when you run the command.

Fig 9

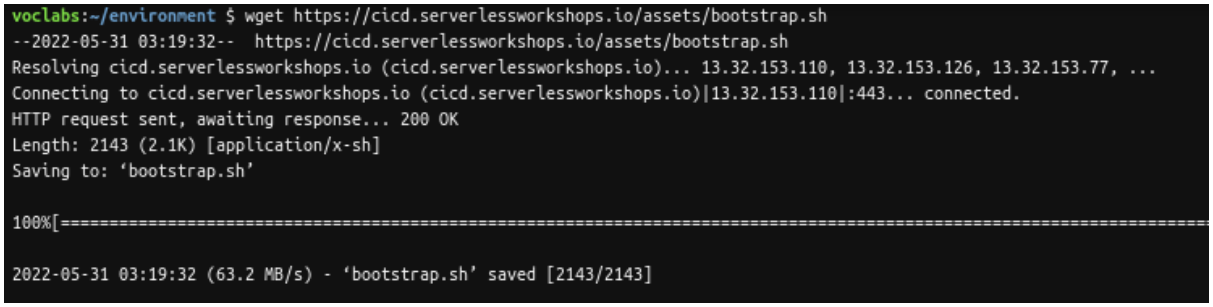
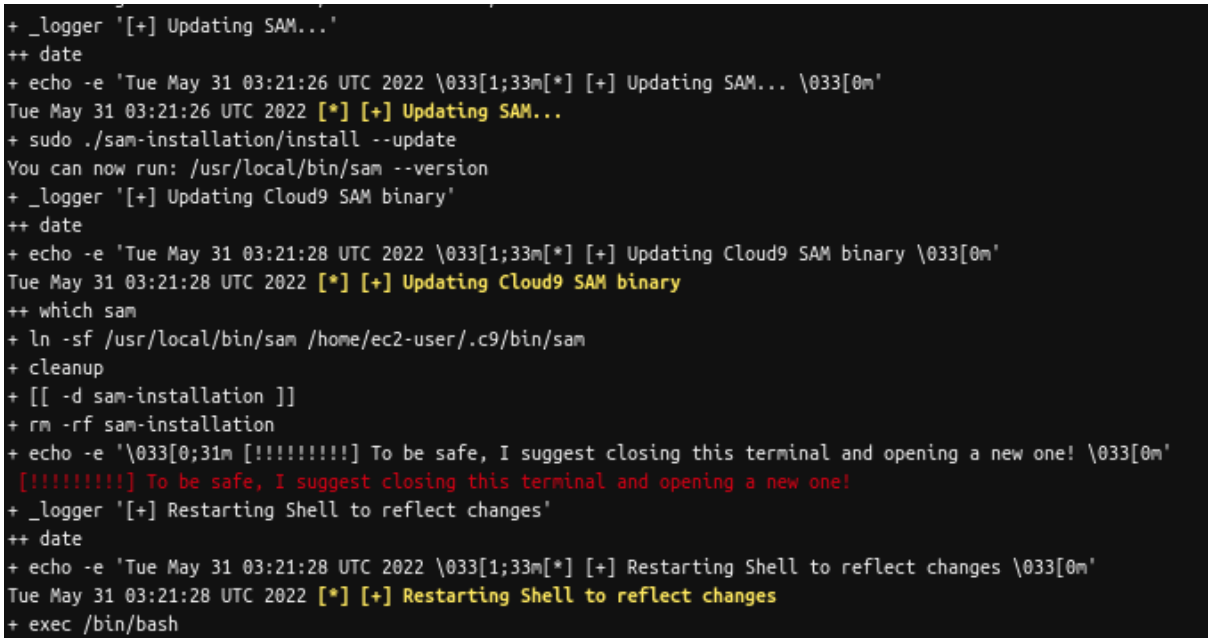


Fig 10



The following command “sam –version” shows the current version of the Serverless Architecture Model. At this current moment, the latest version is show in the Fig 11.

Fig 11

Step 2: Initialize the Serverless Architecture Model by running the command “sam init”.

This is to initialize the Serverless Architecture Model and we want to use the pre-set hello world template.

Follow the choice as shown in the Fig 12-13.

Fig 12

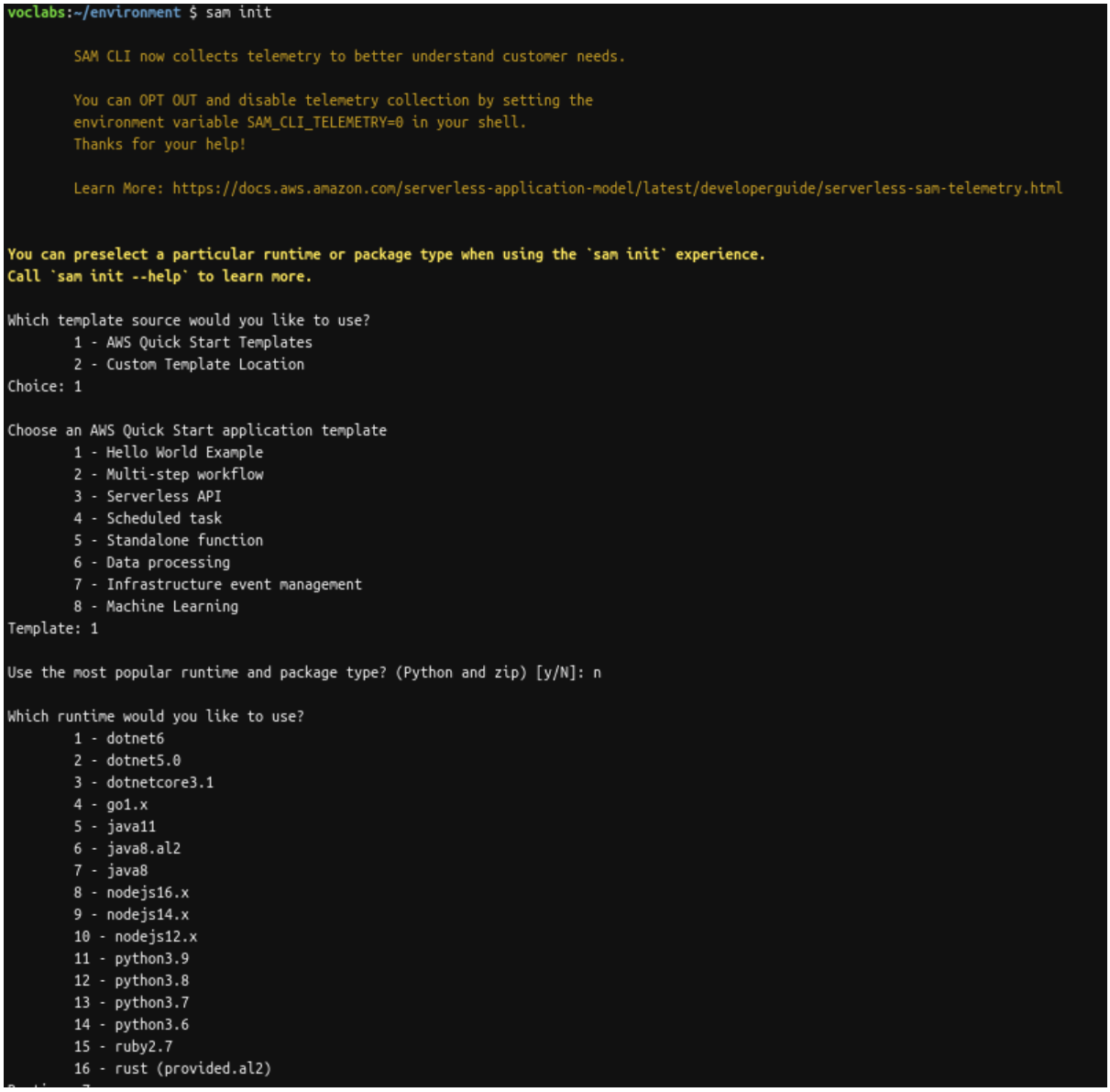
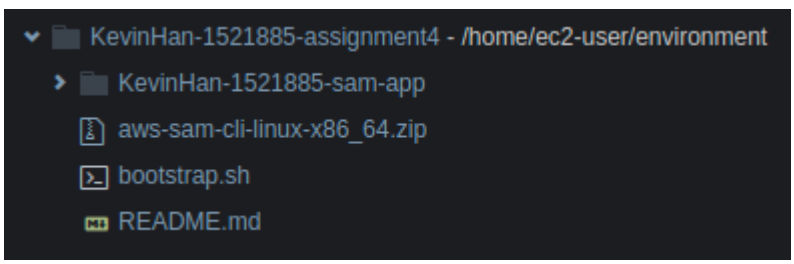


Fig 13

```
        9 - nodejs14.x
        10 - nodejs12.x
        11 - python3.9
        12 - python3.8
        13 - python3.7
        14 - python3.6
        15 - ruby2.7
        16 - rust (provided.al2)
Runtime: 7

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 1

Which dependency manager would you like to use?
        1 - gradle
        2 - maven
Dependency manager: 2

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/N]: n

Project name [sam-app]: KevinHan-1521885-sam-app

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a moment)

    -----------------------
    Generating application:
    -----------------------
    Name: KevinHan-1521885-sam-app
    Runtime: java8
    Architectures: x86_64
    Dependency Manager: maven
    Application Template: hello-world
    Output Directory: .

    Next steps can be found in the README file at ./KevinHan-1521885-sam-app/README.md


    Commands you can use next
    =========================
    [*] Create pipeline: cd KevinHan-1521885-sam-app && sam pipeline init --bootstrap
    [*] Validate SAM template: sam validate
    [*] Test Function in the Cloud: sam sync --stack-name {stack-name} --watch
```

After initializing the Serverless Architecture Model, you should see the directory of the project on the menu at the left side of the screen as shown in Fig 14.

Fig 14



4. Task 3: Deploy Locally

The task is to be able to run the project locally. To do this we need to install some dependency.

Step 1: Install the prebuilt OpenJDK packages, update the configs for the java and javac.

Run the following commands:

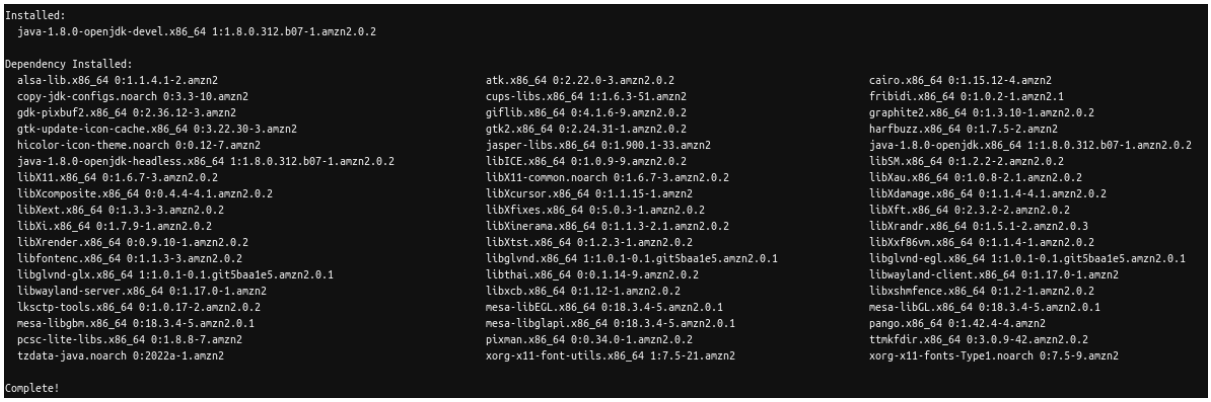
sudo yum -y install java-1.8.0-openjdk-devel

sudo update-alternatives --config java

sudo update-alternatives --config javac

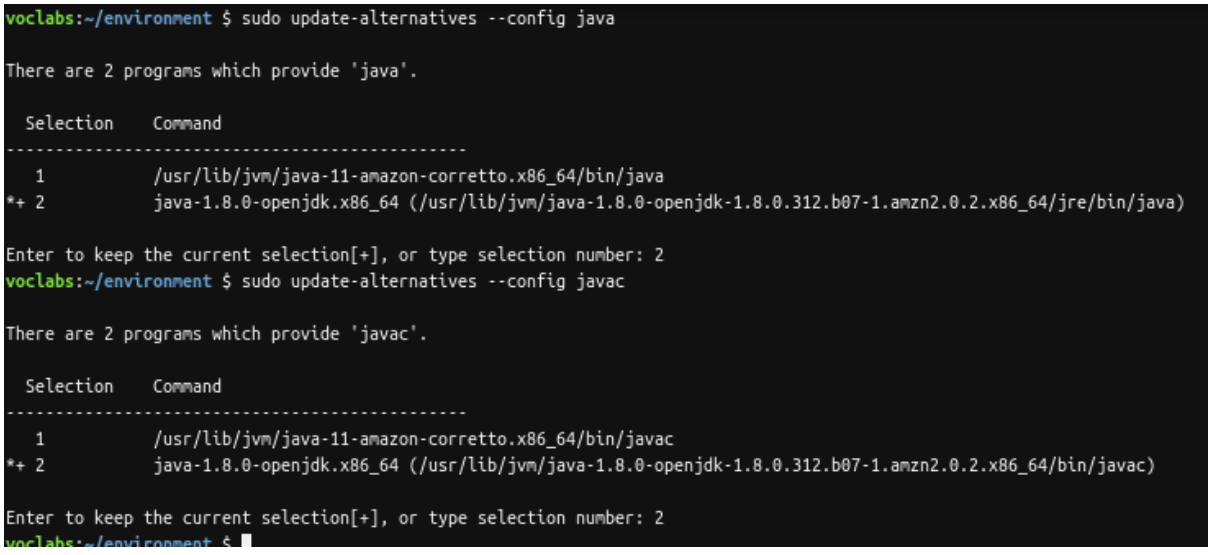
The Fig 15-16 shows the outcome as you enter the commands.

Fig 15



```
Installed:
  java-1.8.0-openjdk-devel.x86_64 1:1.8.0.312.b07-1.amzn2.0.2

Dependency Installed:
  alsa-lib.x86_64 0:1.1.4.1-2.amzn2                atk.x86_64 0:2.22.0-3.amzn2.0.2                cairo.x86_64 0:1.15.12-4.amzn2
  copy-jdk-configs.noarch 0:3.3-10.amzn2           cups-libs.x86_64 1:1.6.3-51.amzn2              fribidi.x86_64 0:1.0.2-1.amzn2.1
  gdk-pixbuf2.x86_64 0:2.36.12-3.amzn2             giflib.x86_64 0:4.1.6-9.amzn2.0.2              graphite2.x86_64 0:1.3.10-1.amzn2.0.2
  gtk-update-icon-cache.x86_64 0:3.22.30-3.amzn2   gtk2.x86_64 0:2.24.31-1.amzn2.0.2              harfbuzz.x86_64 0:1.7.5-2.amzn2
  hicolor-icon-theme.noarch 0:0.12-7.amzn2         jasper-libs.x86_64 0:1.900.1-33.amzn2          java-1.8.0-openjdk.x86_64 1:1.8.0.312.b07-1.amzn2.0.2
  java-1.8.0-openjdk-headless.x86_64 1:1.8.0.312.b07-1.amzn2.0.2  libICE.x86_64 0:1.0.9-9.amzn2.0.2  libSM.x86_64 0:1.2.2-2.amzn2.0.2
  libX11.x86_64 0:1.6.7-3.amzn2.0.2                libX11-common.noarch 0:1.6.7-3.amzn2.0.2       libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
  libXcomposite.x86_64 0:0.4.4-4.1.amzn2.0.2       libXcursor.x86_64 0:1.1.15-1.amzn2             libXdamage.x86_64 0:1.1.4-4.1.amzn2.0.2
  libXext.x86_64 0:1.3.3-3.amzn2.0.2               libXfixes.x86_64 0:5.0.3-1.amzn2.0.2           libXft.x86_64 0:2.3.2-2.amzn2.0.2
  libXi.x86_64 0:1.7.9-1.amzn2.0.2                 libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2       libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2           libXtst.x86_64 0:1.2.3-1.amzn2.0.2             libXxf86vm.x86_64 0:1.1.4-1.amzn2.0.2
  libfontenc.x86_64 0:1.1.3-3.amzn2.0.2            libglvnd.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1  libglvnd-egl.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1
  libglvnd-glx.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1  libthai.x86_64 0:0.1.14-9.amzn2.0.2       libwayland-client.x86_64 0:1.17.0-1.amzn2
  libwayland-server.x86_64 0:1.17.0-1.amzn2        libxcb.x86_64 0:1.12-1.amzn2.0.2               libxshmfence.x86_64 0:1.2-1.amzn2.0.2
  lksctp-tools.x86_64 0:1.0.17-2.amzn2.0.2         mesa-libEGL.x86_64 0:18.3.4-5.amzn2.0.1        mesa-libGL.x86_64 0:18.3.4-5.amzn2.0.1
  mesa-libgbm.x86_64 0:18.3.4-5.amzn2.0.1          mesa-libglapi.x86_64 0:18.3.4-5.amzn2.0.1      pango.x86_64 0:1.42.4-4.amzn2
  pcsc-lite-libs.x86_64 0:1.8.8-7.amzn2            pixman.x86_64 0:0.34.0-1.amzn2.0.2             ttmkfdir.x86_64 0:3.0.9-42.amzn2.0.2
  tzdata-java.noarch 0:2022a-1.amzn2               xorg-x11-font-utils.x86_64 1:7.5-21.amzn2      xorg-x11-fonts-Type1.noarch 0:7.5-9.amzn2

Complete!
```

Fig 16



```
voclabs:~/environment $ sudo update-alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           /usr/lib/jvm/java-11-amazon-corretto.x86_64/bin/java
*+ 2           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86_64/jre/bin/java)

Enter to keep the current selection[+], or type selection number: 2
voclabs:~/environment $ sudo update-alternatives --config javac

There are 2 programs which provide 'javac'.

  Selection    Command
-----------------------------------------------
   1           /usr/lib/jvm/java-11-amazon-corretto.x86_64/bin/javac
*+ 2           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86_64/bin/javac)

Enter to keep the current selection[+], or type selection number: 2
voclabs:~/environment $
```

Step 2: Install maven, it is used to by the project to build.

Run the following commands:

sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/7/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

The Fig 17-18 shows what it is expected when you run the commands to install maven.

Fig 17

Fig 18

Step 3: We can now build the project from the template file.

Ensure that you are in the right directory, run the command “sam build”. If done correctly, you should see that the console will output “Build Succeeded” as shown in the Fig 19. This shows the project has been successfully built.

Fig 19

Step 4: To deploy the project locally after the successful build.

To deploy the project, ensure that you are currently in yourname-studentid-sam-app folder. Then run the command “sam local start-api --port 8080” to deploy the project.

The Fig 20 shows the project has deploy.

Fig 20

```
voclabs:~/environment/KevinHan-1521885-sam-app $ sam local start-api --port 8080
Mounting HelloWorldFunction at http://127.0.0.1:8080/hello [GET]
You can now browse to the above endpoints to invoke your functions. You do not need to restart/reload SAM CLI while working on your functions, chan
ges will be reflected instantly/automatically. You only need to restart SAM CLI if you update your AWS SAM template
2022-05-31 09:22:15  * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
```

To test whether the project is working locally, we can send “curl http://localhost:8080/hello” in another terminal. The output should be the same as the Fig 21.

Fig 21

```
voclabs:~/environment $ curl http://localhost:8080/hello
{ "message": "hello world", "location": "184.72.143.166" }voclabs:~/environment $
```

The Fig 22 shows the request made by the curl command above.

Fig 22

```
Mounting /home/ec2-user/environment/KevinHan-1521885-sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegate
d inside runtime container
END RequestId: 9731497a-29b7-4414-b3f3-b65e31b27a1a
REPORT RequestId: 9731497a-29b7-4414-b3f3-b65e31b27a1a  Init Duration: 1.59 ms  Duration: 1054.19 ms    Billed Duration
: 1055 ms       Memory Size: 512 MB     Max Memory Used: 512 MB
2022-05-31 09:23:59 127.0.0.1 - - [31/May/2022 09:23:59] "GET /hello HTTP/1.1" 200 -
```

5. Task 4: Configuring your repository

The task is to have our latest version of deployed project to be stored in a cloud repository. For this assignment, we are storing the repository in aws’s git service.

Step 1: Creating the git repository

Run the command ‘aws codecommit create-repository --repository-name “yournamesamapp-repo”’ The output of the command is shown in Fig 23.

Fig 23

```
voclabs:~/environment/KevinHan-1521885-sam-app $ aws codecommit create-repository --repository-name "KevinHan-samapp-re
po"
{
    "repositoryMetadata": {
        "accountId": "505253522687",
        "repositoryId": "8017a9a5-d655-4083-a9e1-4ef9c7057fd8",
        "repositoryName": "KevinHan-samapp-repo",
        "lastModifiedDate": 1653989413.693,
        "creationDate": 1653989413.693,
        "cloneUrlHttp": "https://git-codecommit.us-east-1.amazonaws.com/v1/repos/KevinHan-samapp-repo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/KevinHan-samapp-repo",
        "Arn": "arn:aws:codecommit:us-east-1:505253522687:KevinHan-samapp-repo"
    }
}
```

Step 2: After successfully creating the repo, we need to configure the git repo with your details. To allow us to know who the commits are made by. Run the command shown in Fig 24.

Fig 24

```
voclabs:~/environment/KevinHan-1521885-sam-app $ git config --global user.name && --global user.email
```

Step 3: We need to config git to ignore some files. Create a .gitignore file, if it is not showing, enable hidden files. We do not want to store the built applications, so add the following as show in the fig 25.

Fig 25

```
1   .aws-sam/
2   packaged.yaml
3   pipeline.sh
```

Step 4: We can now population our repository with the project files. To do this we need to run the following commands “git init”, “git add .” and “git commit -m “Initial commit””.

Git init – to initialise a new local repository which store all the object and ref used in the project and creates a project’s history.

Git add . – add the updates made to the project

Git commit – it captures the snapshot of the current stage changes of the project.

When you run the commands, it should look similar to the Fig 26-27.

Fig 26

```
voclabs:~/environment/KevinHan-1521885-sam-app $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ec2-user/environment/KevinHan-1521885-sam-app/.git/
```

Fig 27

```
voclabs:~/environment/KevinHan-1521885-sam-app (master) $ git commit -m "Initial commit"
[master (root-commit) 7b096f2] Initial commit
 Committer: EC2 Default User <ec2-user@ip-172-31-30-42.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 7 files changed, 358 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 HelloWorldFunction/pom.xml
 create mode 100644 HelloWorldFunction/src/main/java/helloworld/App.java
 create mode 100644 HelloWorldFunction/src/test/java/helloworld/AppTest.java
 create mode 100644 README.md
 create mode 100644 events/event.json
 create mode 100644 template.yaml
```

Step 5: Now we can add the remote origin that we created in step 1 and push the repository to the “cloneUrlHttp”. Follow the commands show in Fig 28-29.

Fig 28

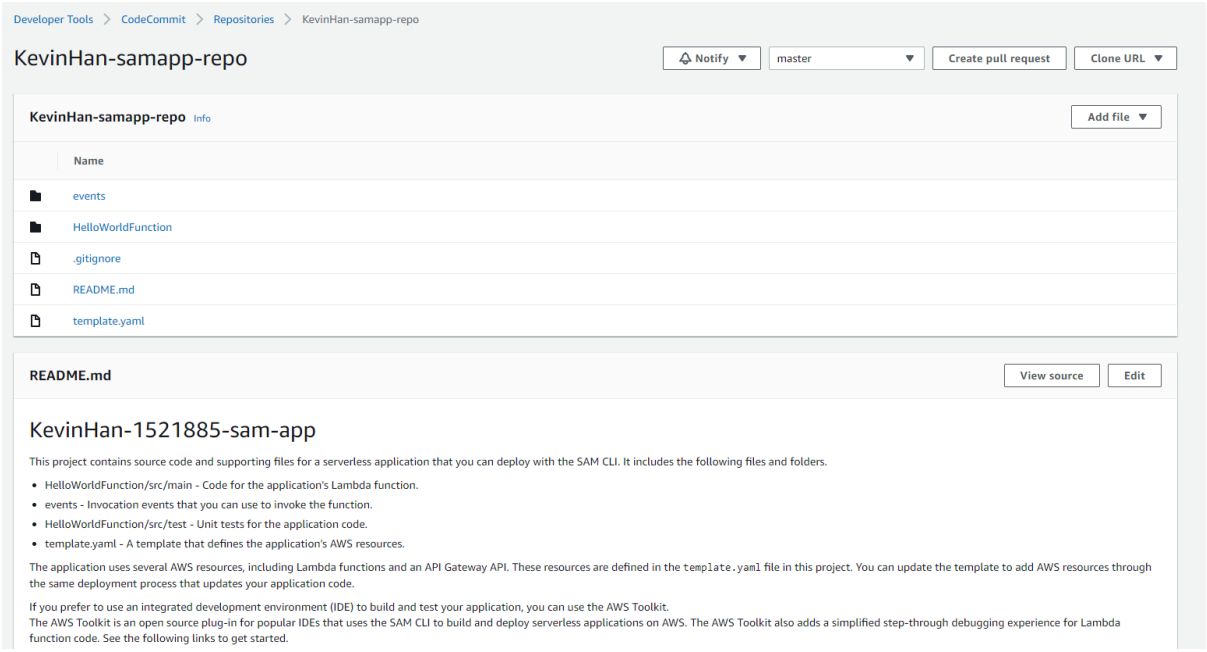
```
voclabs:~/environment/KevinHan-1521885-sam-app (master) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/KevinHan-samapp-repo
```

Fig 29

```
voclabs:~/environment/KevinHan-1521885-sam-app (master) $  git push --set-upstream origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (18/18), 6.91 KiB | 1011.00 KiB/s, done.
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/KevinHan-samapp-repo
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

After pushing the repository, you should be able to see your populated repository in the AWS codecommit console. Your populated repository should look like Fig 30.

Fig 30



6. Task 5: Implementing a CI/CD pipeline

The task is to create a bash script which automates the testing, building, committing, and pushing. If the tests are successful, it will attempt to build. If the build is successful, it will commit and push to the AWS Code Commit and ask the user if they want to deploy the project. If the build fails, it should not commit and push to AWS Code Commit. If the tests fails, it should not build, commit, and push to AWS Code Commit.

Fig 31

```
echo "(1) Test"
cd HelloWorldFunction
if mvn test ; then
    echo "Test Successful"
    cd ..
    echo "(2) Build"
    if sam build ; then
        read -p "Commit Message : " message
        git add .
        git commit -m "$message"
        git push
        while true; do
            read -p "Deploy Locally(Y/N): " deploy
            case $deploy in
                [yY] ) echo "Deploy locally";
                    sam local start-api --port 8080
                    break;;
                [nN] ) echo "Not deploying locally";
                    exit;;
                * ) echo "Please Enter Y or N";;
            esac
        done
    else
        echo "Build failed"
    fi
else
    echo "Test failed"
fi
```

I have setup the bash script as shown in Fig 31. The bash script first check if it can run the test successfully, then check if can build successfully. If it is build successful, it will ask the user for a commit message before committing and pushing the AWS Code Commit. Then ask the user whether they want to deploy locally. The user are given 2 choices Y or N. If other input are given, it will give an error and ask the user again. Error outputs are shown if the test or build are to fail. I decided to developed this way based on the assignment specification. The following Fig 32-36 shows my pipeline script running.

Fig 32

```
voclabs:~/environment/KevinHan-1521885-sam-app (master) $ bash pipeline.sh
(1) Test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building A sample Hello World created for SAM CLI. 1.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ HelloWorld ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ HelloWorld ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ HelloWorld ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ HelloWorld ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ HelloWorld ---
[INFO] Surefire report directory: /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running helloworld.AppTest
Downloading quotes.txt from S3 bucket kevinhan-bucket...
Done!
Successfully wrote to the file.
99
Success is walking from failure to failure with no loss of enthusiasm. ~Winston Churchill
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.258 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.919 s
[INFO] Finished at: 2022-06-15T09:55:16Z
```

Fig 33

```
bash - "ip-172-31-30-42.e ×
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.919 s
[INFO] Finished at: 2022-06-15T09:55:16Z
[INFO] Final Memory: 10M/24M
[INFO] ------------------------------------------------------------------------
Test Successful
(2) Build
Your template contains a resource with logical ID "ServerlessRestApi", which is a reserved logical ID in AWS SAM. It could result in unexpected behaviors and is not recommended.
Test the latest build changes for Java runtime 'SAM_CLI_BETA_MAVEN_SCOPE_AND_LAYER=1 sam build'. These changes will replace the existing flow on 1st of April 2022. Check https://github.com/aws/aws-sam-cli/issues/3639 for more information.
Building codeuri: /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction runtime: java8 metadata: {} architecture: x86_64 functions: ['HelloWorldFunction']
Running JavaMavenWorkflow:CopySource
Running JavaMavenWorkflow:MavenBuild
Running JavaMavenWorkflow:MavenCopyDependency
Running JavaMavenWorkflow:MavenCopyArtifacts

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {stack-name} --watch
[*] Deploy: sam deploy --guided

Commit Message : Implemented Task 6
[master e765503] Implemented Task 6
 Committer: EC2 Default User <ec2-user@ip-172-31-30-42.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 14 files changed, 292 insertions(+), 15 deletions(-)
 create mode 100644 .attach_pid14172
 create mode 100644 .attach_pid20746
```

Fig 34

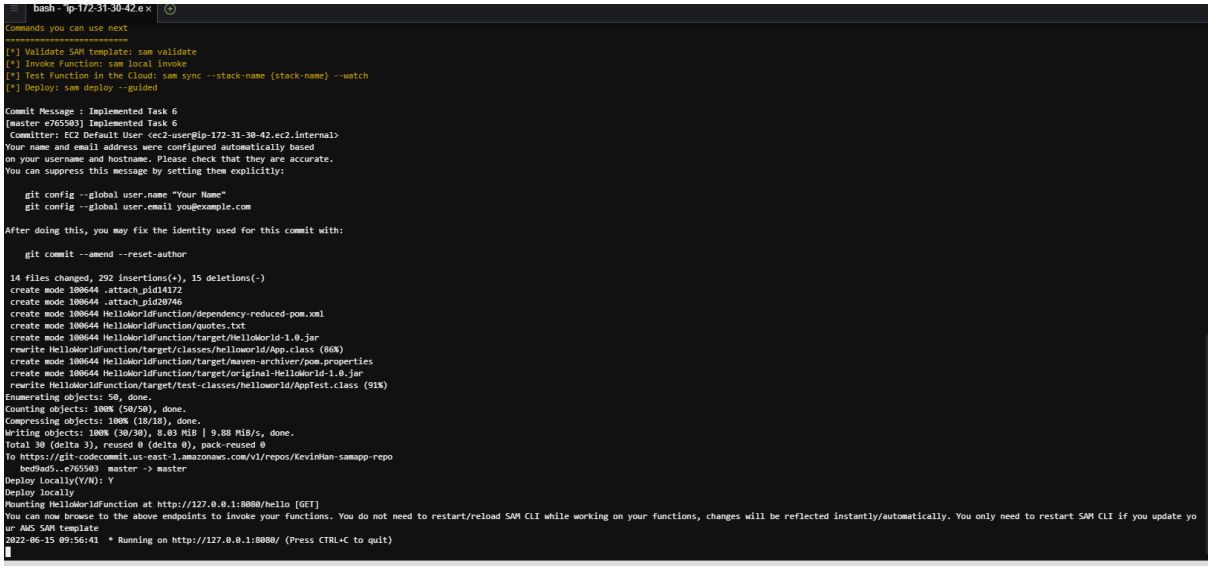
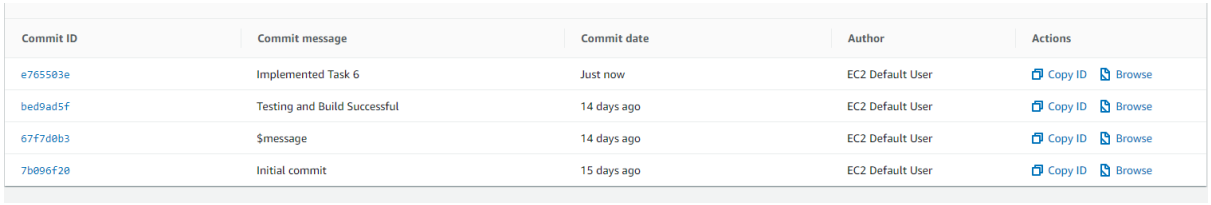


Fig 35



Commit ID	Commit message	Commit date	Author	Actions
e765503e	Implemented Task 6	Just now	EC2 Default User	Copy ID Browse
bed9ad5f	Testing and Build Successful	14 days ago	EC2 Default User	Copy ID Browse
67f7d0b3	$message	14 days ago	EC2 Default User	Copy ID Browse
7b096f20	Initial commit	15 days ago	EC2 Default User	Copy ID Browse

Fig 36

7. Task 6: Implementing Functionality and Testing

The task is to make some changes to the java application and testing the pipeline and application. The changes are to allow the program to access the S3 bucket and retrieve the quotes stored and randomly display a quote back when a curl is made. Testing must ensure the quote exists in the quotes.txt retrieve from the bucket.

Fig 37 shows the test failed when we made a change to the hello world message in the App.java.

Fig 37

```
Results :

Failed tests:   successfulResponse(helloworld.AppTest)

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.857 s
[INFO] Finished at: 2022-06-01T01:35:40Z
[INFO] Final Memory: 8M/20M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project HelloWorld: There are test failures.
[ERROR]
[ERROR] Please refer to /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/target/surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
Test failed
```

The following Fig 38-39 show the bucket that has been created and the storing the quotes.txt file in the bucket.

Fig 38

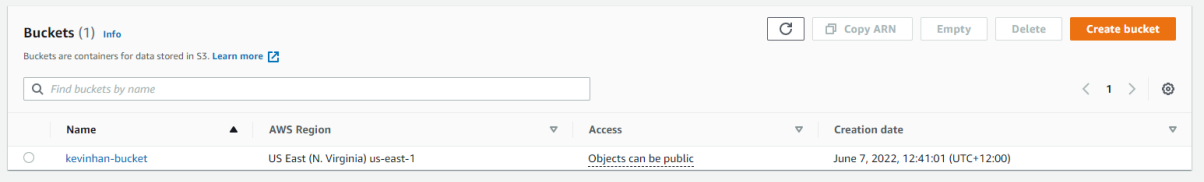


Fig 39

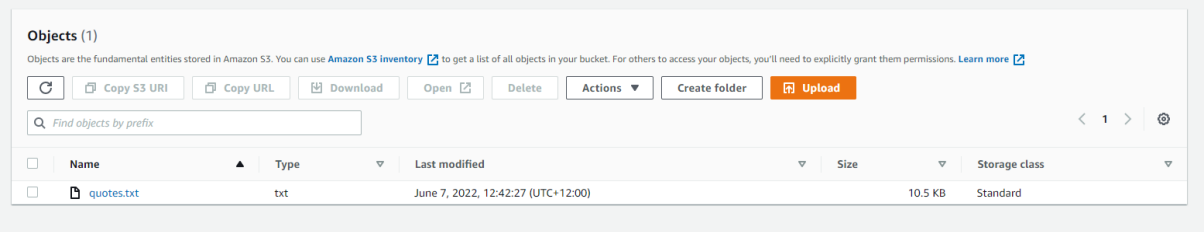


Fig 40 show the imports of the java and aws service that I will be using for the changes made in this task.

Fig 41-42 shows the dependency needed for the App.java to retrieve the contents of the buckets. We are getting the particular dependency instead of the whole Maven SDK Modules, so it doesn't take a long time building the application. The dependency are added to the pom.xml.

Fig 40

```
package helloworld;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.net.URL;
import java.util.concurrent.ThreadLocalRandom;
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.ArrayList;
import java.nio.charset.StandardCharsets;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;
/**
```

Fig 41

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Fig 42

```
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-dynamodb</artifactId>
    </dependency>
</dependencies>
```

Fig 43

```java
public class AppTest {
  @Test
  public void successfulResponse() {
    App app = new App();
    APIGatewayProxyResponseEvent result = app.handleRequest(null, null);
    assertEquals(200, result.getStatusCode().intValue());
    assertEquals("application/json", result.getHeaders().get("Content-Type"));
    ArrayList<String> quoteList = new ArrayList<String>();
    BufferedReader reader;
    try {
      reader = new BufferedReader(new FileReader("/home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/quotes.txt"));
      String line = reader.readLine();
      while (line != null) {
        // System.out.println(line);
        // read next line
        line = reader.readLine();
        quoteList.add(line);
      }
      reader.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
    String content = result.getBody();
    // System.out.println(content);
    String quote = result.getBody();
    quote = quote.substring(14);
    quote = quote.split("\"")[0];
    // System.out.println(quote);
    assertNotNull(content);
    assertTrue(content.contains("\"message\""));
    assertTrue(quoteList.contains(quote));
    assertTrue(content.contains("\"location\""));
  }
}
```

Fig 43 show the AppTest.java that has been modified to check whether the quote parse contains within the quotes.txt file.

Fig 44

```java
// Code to parse bucket data
String key_name = "quotes.txt";
String bucket_name = "kevinhan-bucket";
System.out.format("Downloading %s from S3 bucket %s...\n", key_name, bucket_name);
String result = "";
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
try {
    S3Object o = s3.getObject(bucket_name, key_name);
    S3ObjectInputStream s3is = o.getObjectContent();
    result = new BufferedReader(new InputStreamReader(s3is)).lines().parallel().collect(Collectors.joining("\n"));
    // System.out.println(result);
    // FileOutputStream fos = new FileOutputStream(new File(key_name));
    // byte[] read_buf = new byte[1024];
    // int read_len = 0;
    // while ((read_len = s3is.read(read_buf)) > 0) {
    //     fos.write(read_buf, 0, read_len);
    // }
    s3is.close();
    // fos.close();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
} catch (IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Done!");

try {
    FileWriter myWriter = new FileWriter(key_name);
    myWriter.write(result);
    myWriter.close();
    System.out.println("Successfully wrote to the file.");
} catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
}
```

Fig 44 show the code that parse in from the bucket. Make sure that the region of the bucket is based on the region you created your bucket. Since the SAM runs on the cloud, it is not able to read and write 2 files. So, I decided to convert s3is input stream to a string as shown

in the Fig 44. I added a file write which will write the content of the string to a file which is used by the AppTest.java to check whether the quote exists.

Fig 45

```
//Code to setup quote
// System.out.println(System.getProperty("user.dir"));
ArrayList<String> quoteList = new ArrayList<String>();
   BufferedReader reader;
try {
    reader = new BufferedReader(new FileReader("/home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/quotes.txt"));
    String line = reader.readLine();
    while (line != null) {
        // System.out.println(line);
        // read next line
        line = reader.readLine();
        quoteList.add(line);
    }
    reader.close();
} catch (IOException e) {
    e.printStackTrace();
}
String[] tempList = result.split(System.lineSeparator());
for(int i = 0; i < tempList.length; i++) {
    quoteList.add(tempList[i]);
}
System.out.println(quoteList.size());
```

Fig 45 shows the string receive from the bucket being separated by the line separator and stored in an array list called quoteList.

Fig 46, the output has been modified to output the quote instead of the original hello message.

Fig 46

```
String output = String.format("{ \"message\": \""+quote+"\", \"location\": \"%s\" }", pageContents);
```

Fig 47

```
voclabs:~/environment/KevinHan-1521885-sam-app/HelloWorldFunction (master) $ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building A sample Hello World created for SAM CLI. 1.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ HelloWorld ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ HelloWorld ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ HelloWorld ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ HelloWorld ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ HelloWorld ---
[INFO] Surefire report directory: /home/ec2-user/environment/KevinHan-1521885-sam-app/HelloWorldFunction/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running helloworld.AppTest
Downloading quotes.txt from S3 bucket kevinhan-bucket...
Done!
Successfully wrote to the file.
99
Knowledge is being aware of what you can do. Wisdom is knowing when not to do it. ~Anonymous
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.143 sec

Results :
```

Fig 48

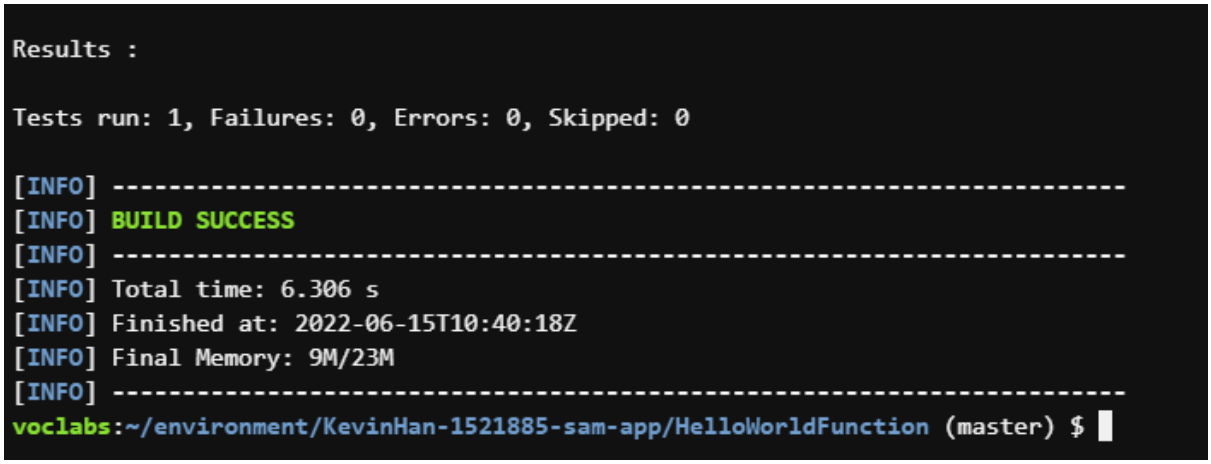


Fig 49

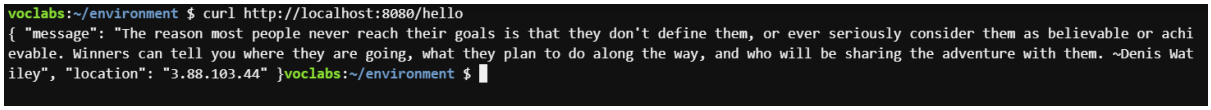


Fig 50

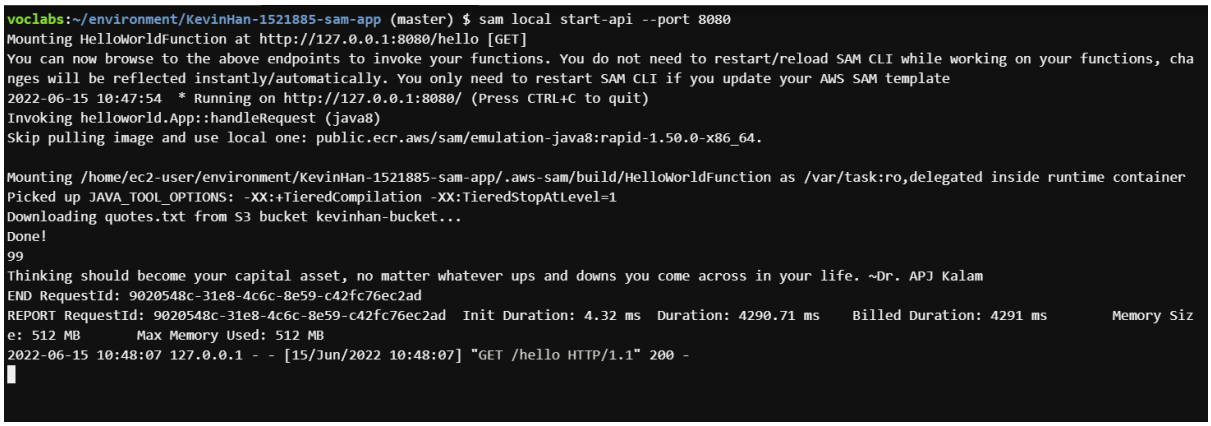


Fig 47-50 shows the test being conducted after the changes has been made to the assignment brief. We can see the message response with a quote from the quotes.txt from the bucket list. For the mvn test, we are testing whether the quote exist within the quote.txt, the http status code is 200.

8. Conclusion

The purpose of the project is to show our understanding of implementation of functionality and testing of the changes made. Allowing automate scripts allow us as software engineers to efficiently test and push the latest working deploy project to the repository.