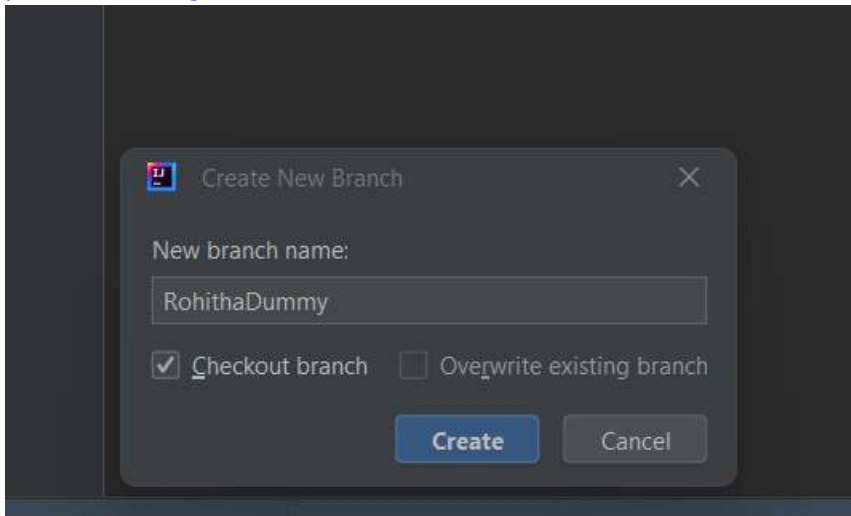


DevOps topics covered in the assignment

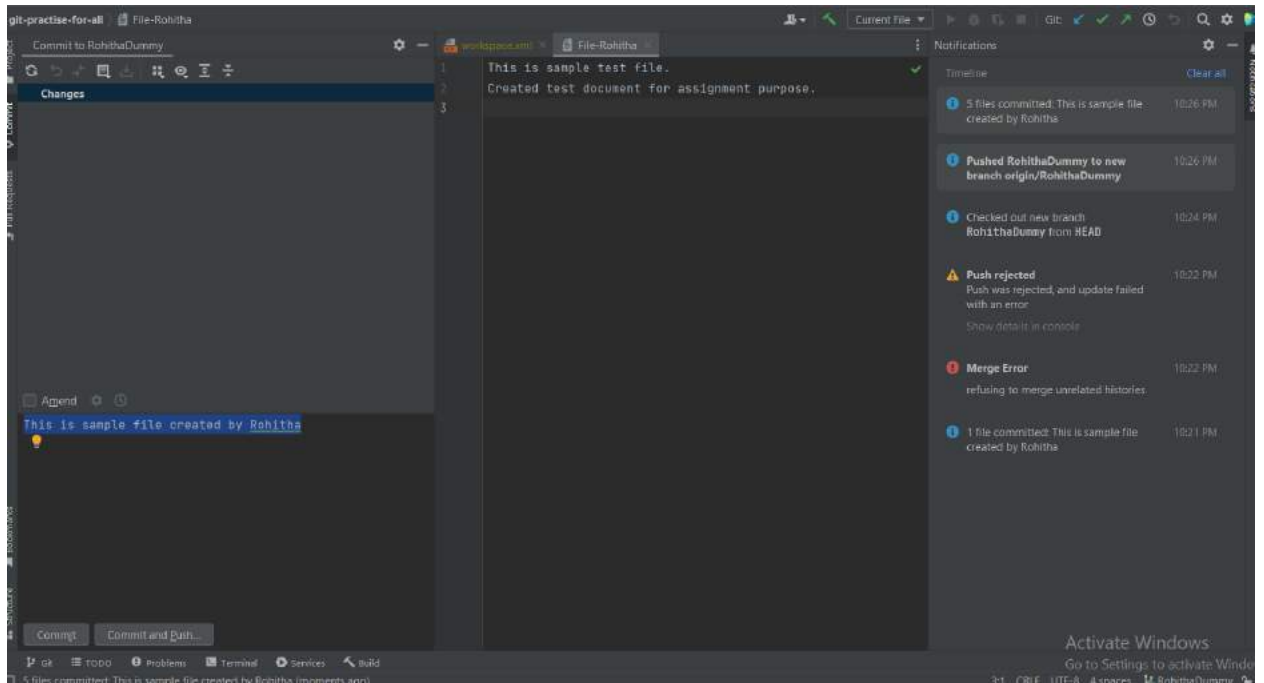
1. GitHub (Task-1 and Task-2)
2. Gradle
3. Java
4. Linux
5. Maven
6. Spring Boot
7. Jenkins
8. AWS SQS and Lambda (Task-1 and Task-2)

1. 1) GitHub (Task-1)

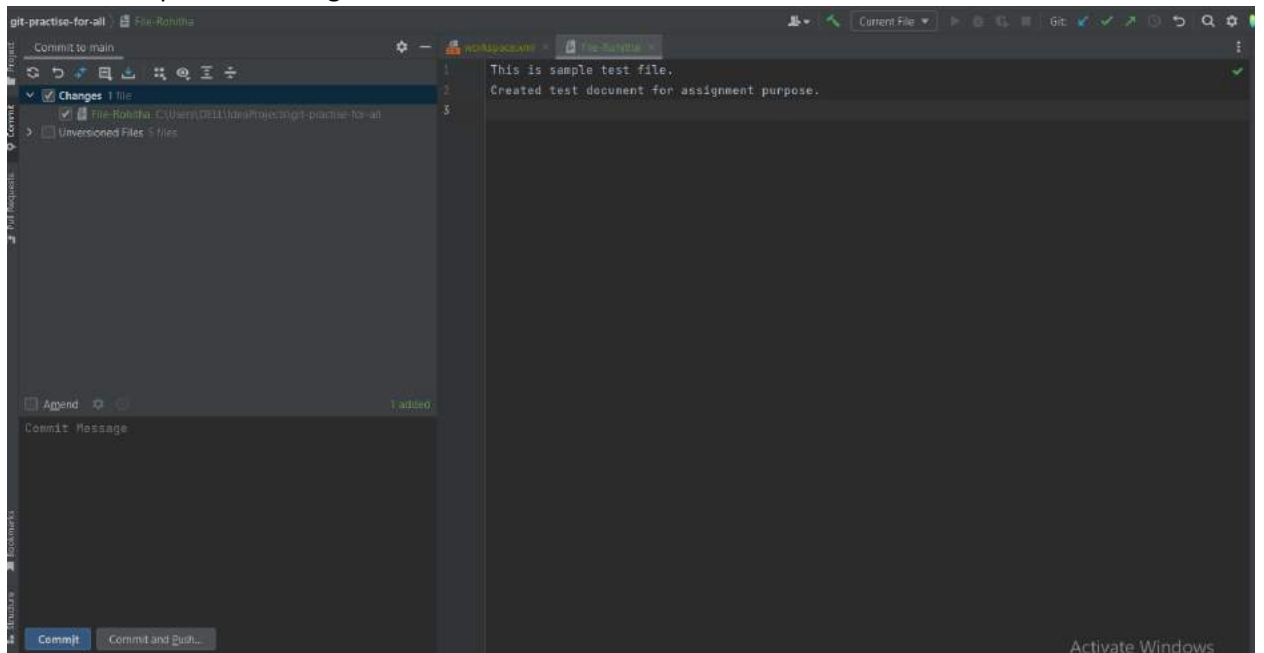
1. Install IntelliJ
2. Generate the token using below path
https://github.com/ -->settings-->Developer settings -->Personal access tokens -->Generate new token--> select all the scopes --> Generate token.
3. Open the project in IntelliJ, Clone the git repository (<https://github.com/columbuskvmk/git-practise-for-all.git>) and create a new branch

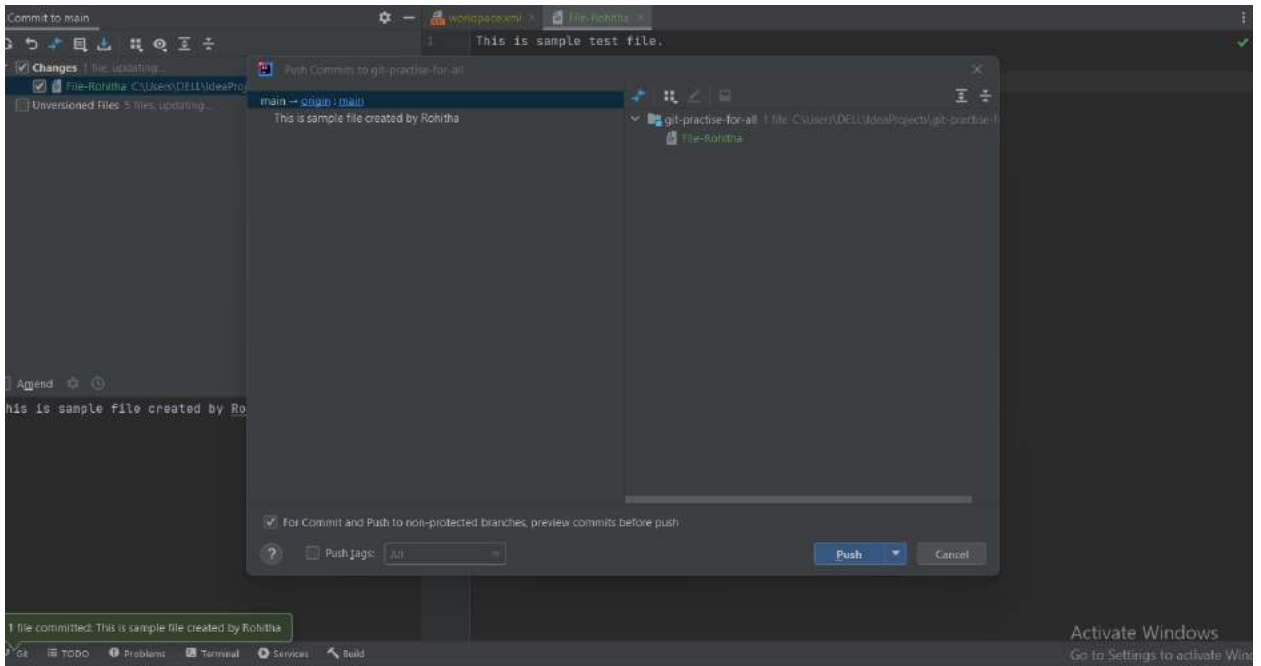


4. Add the new file as below:

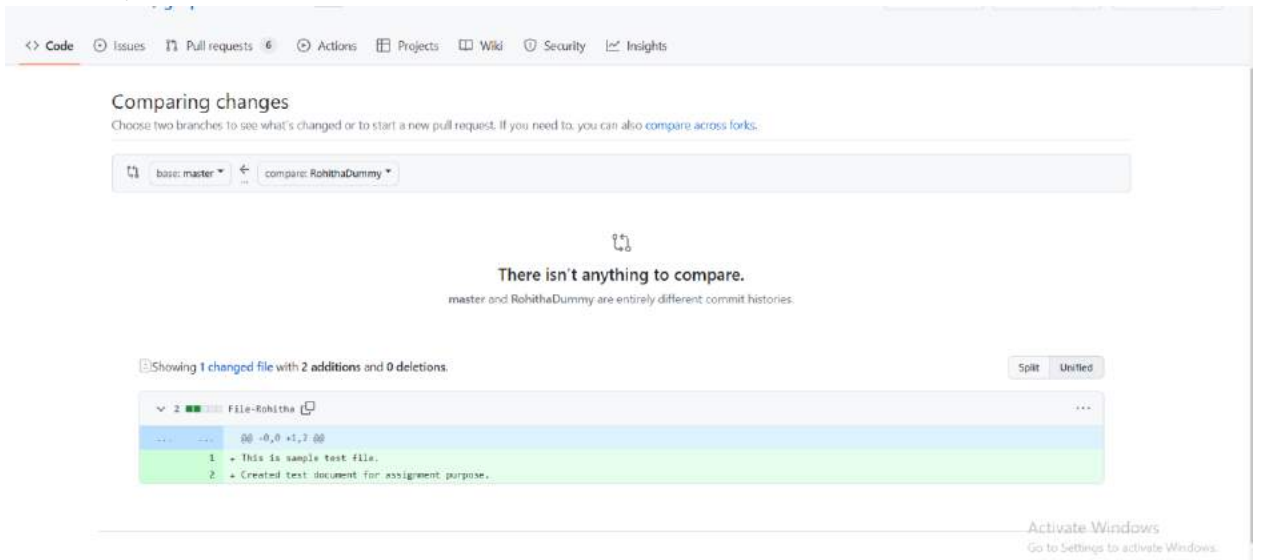


5. Commit and push the changes to master as below:



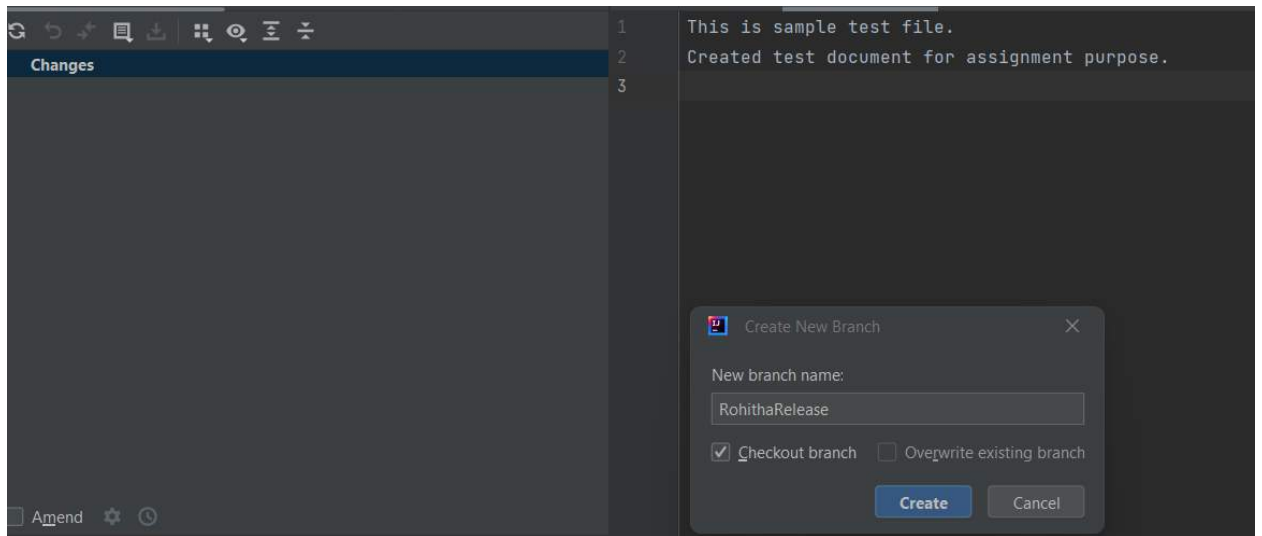


6. In Git Hub, we can see as below:

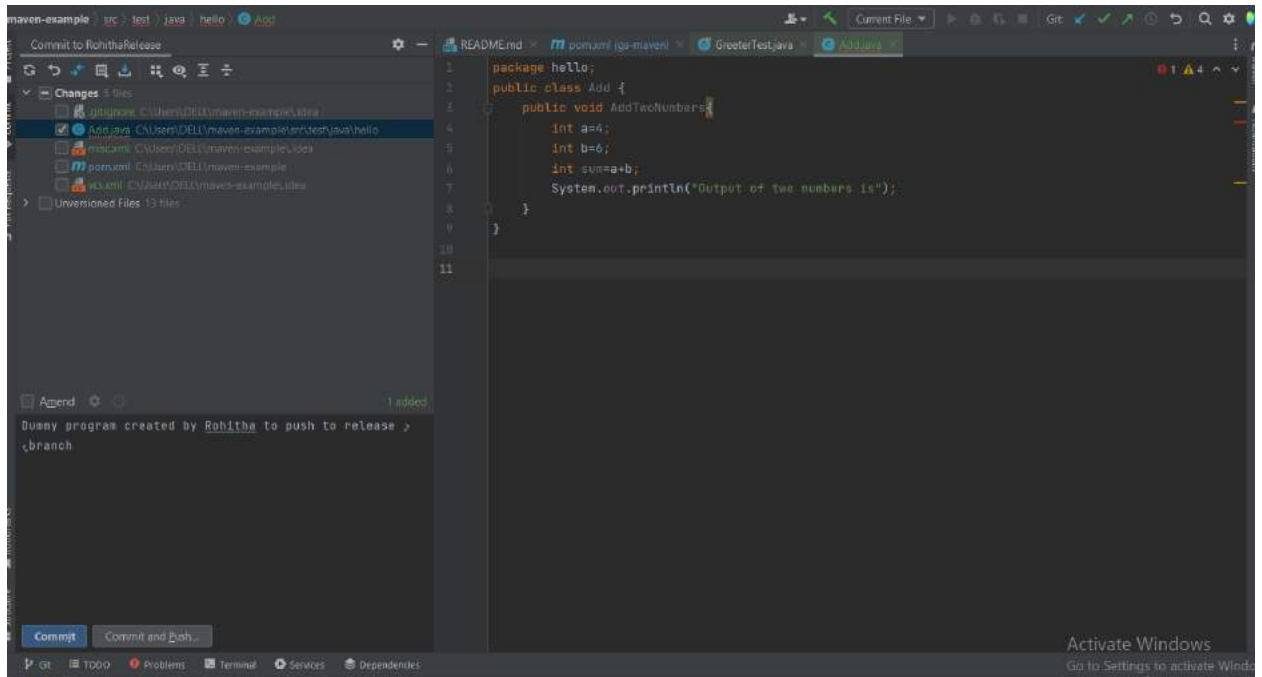


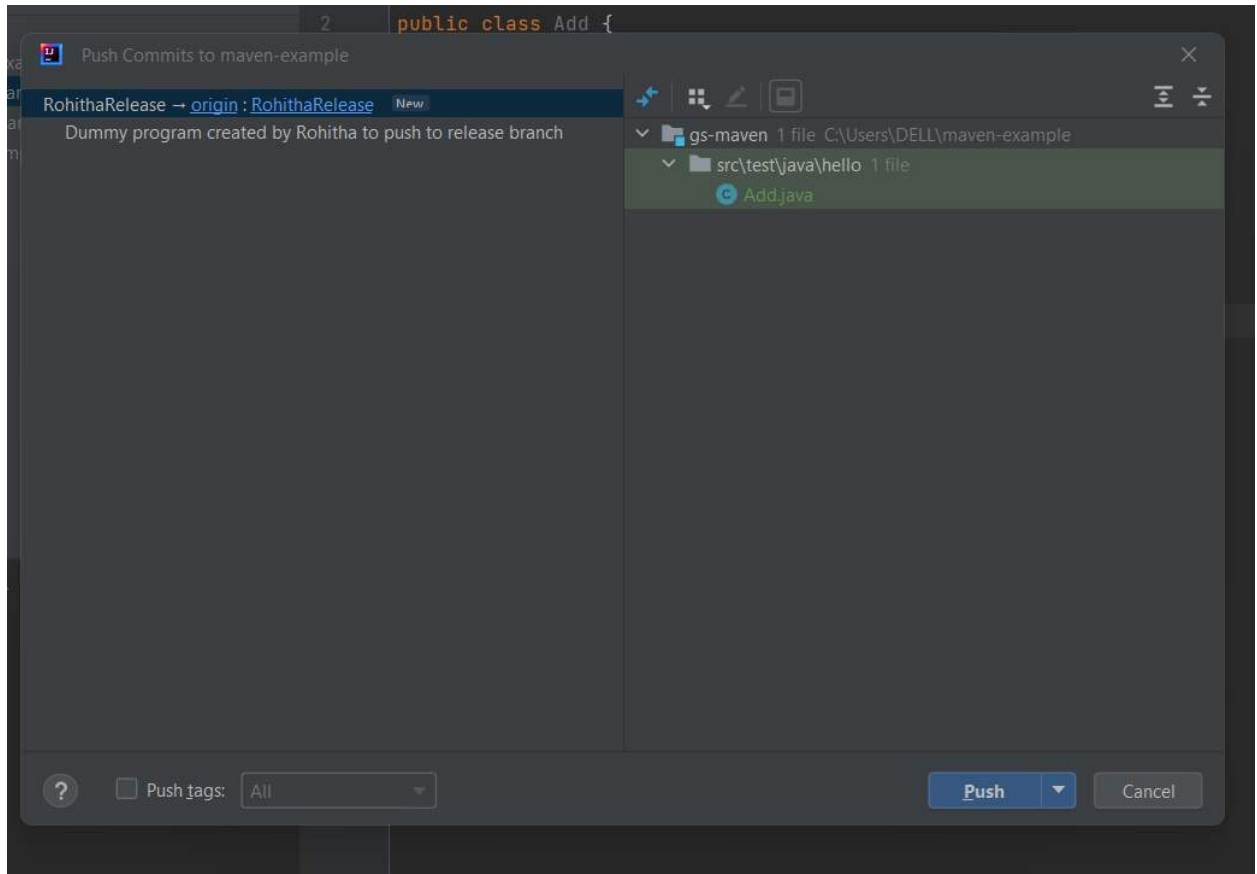
1. 2)GitHub (Task-2)

1. Create a new release branch from master as below:

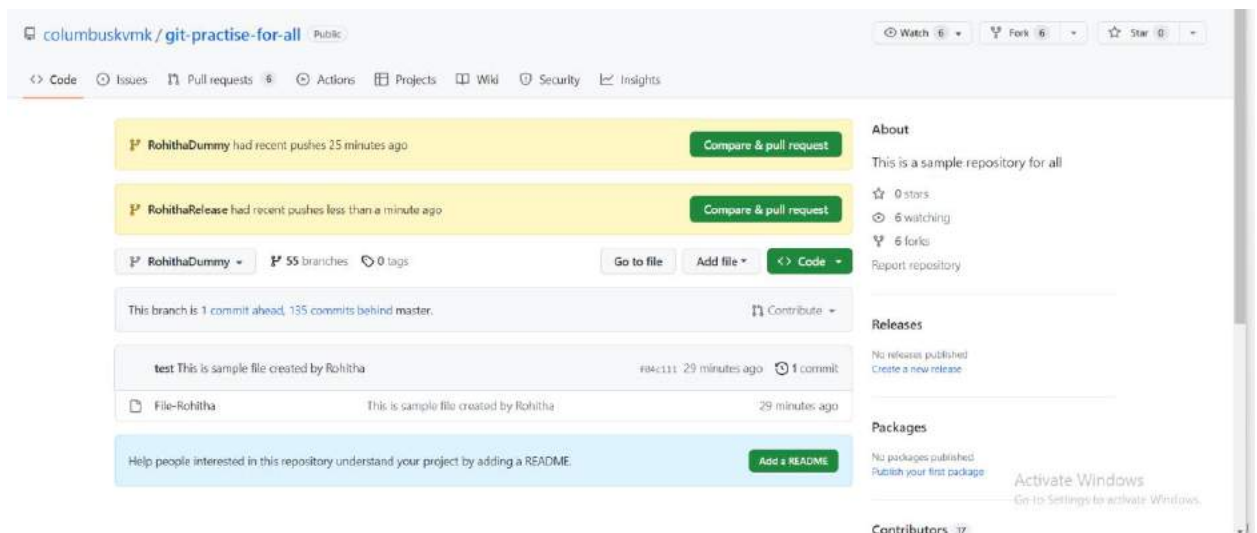


2. Make a change to the file in the Rohitha Release branch. Commit and push the above changes.





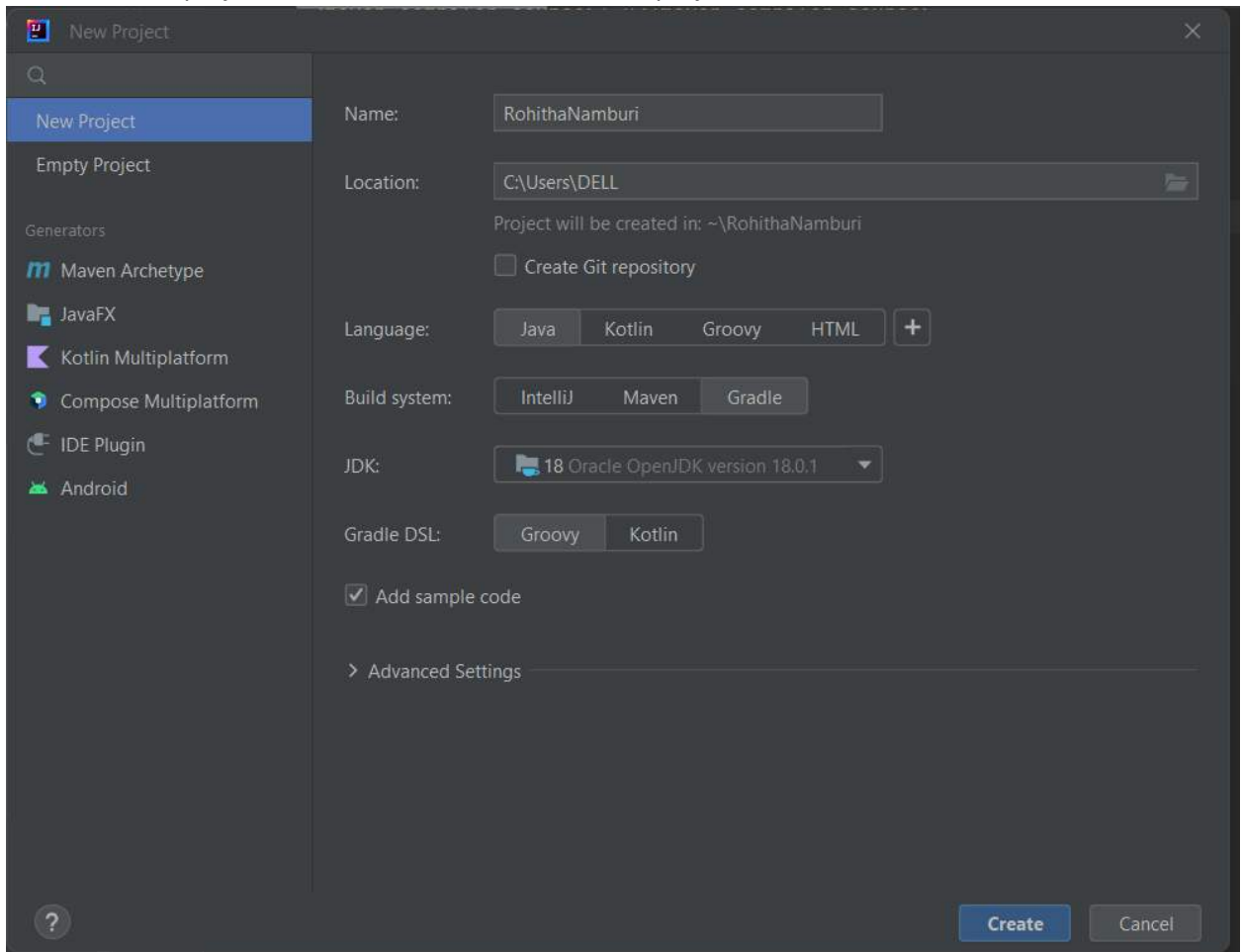
3. Create a pull request from Rohitha release branch to Rohitha Dummy branch from the GitHub repository.



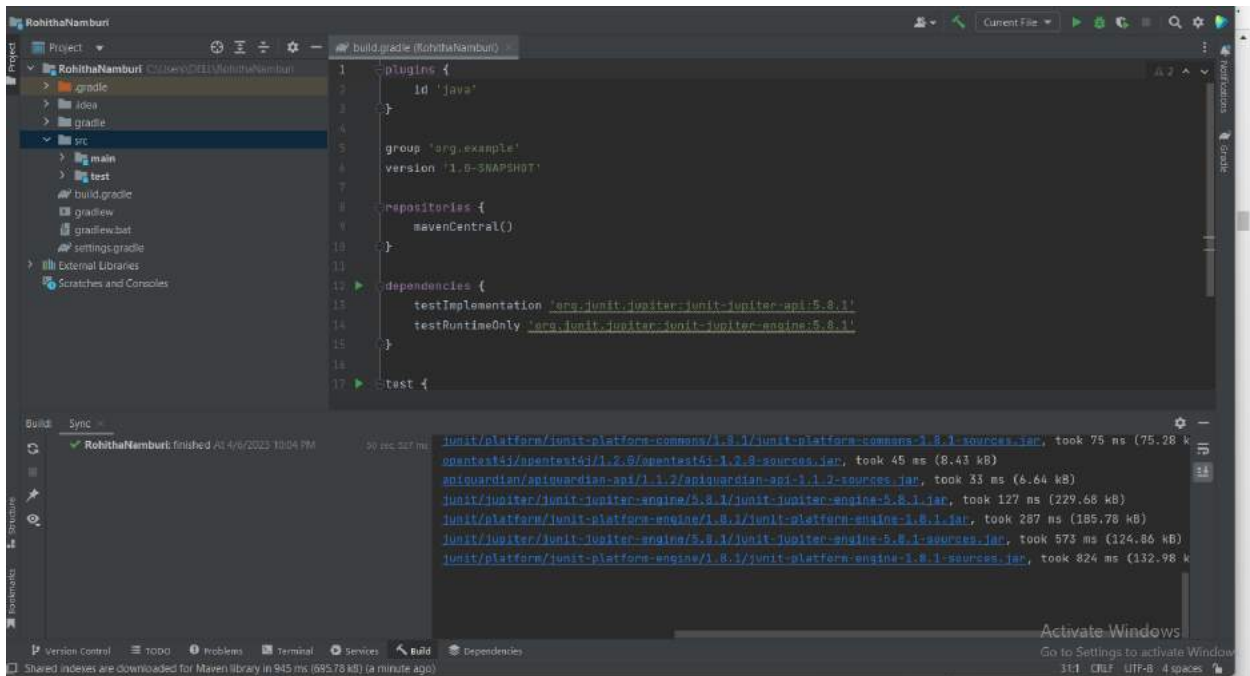
4. Once pull request is created, Click on Commit merge and then click on Merge pull request and Confirm merge.

2. Gradle :

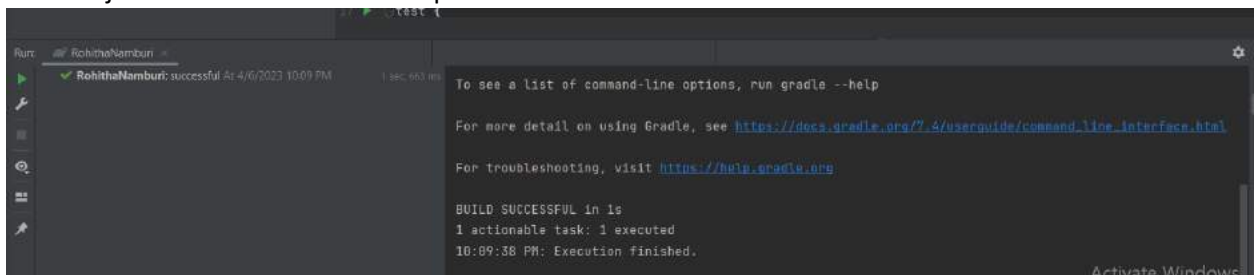
1. Create Gradle project in IntelliJ and click on File->New project as shown below:



2. Gradle project will be created and shown as below:

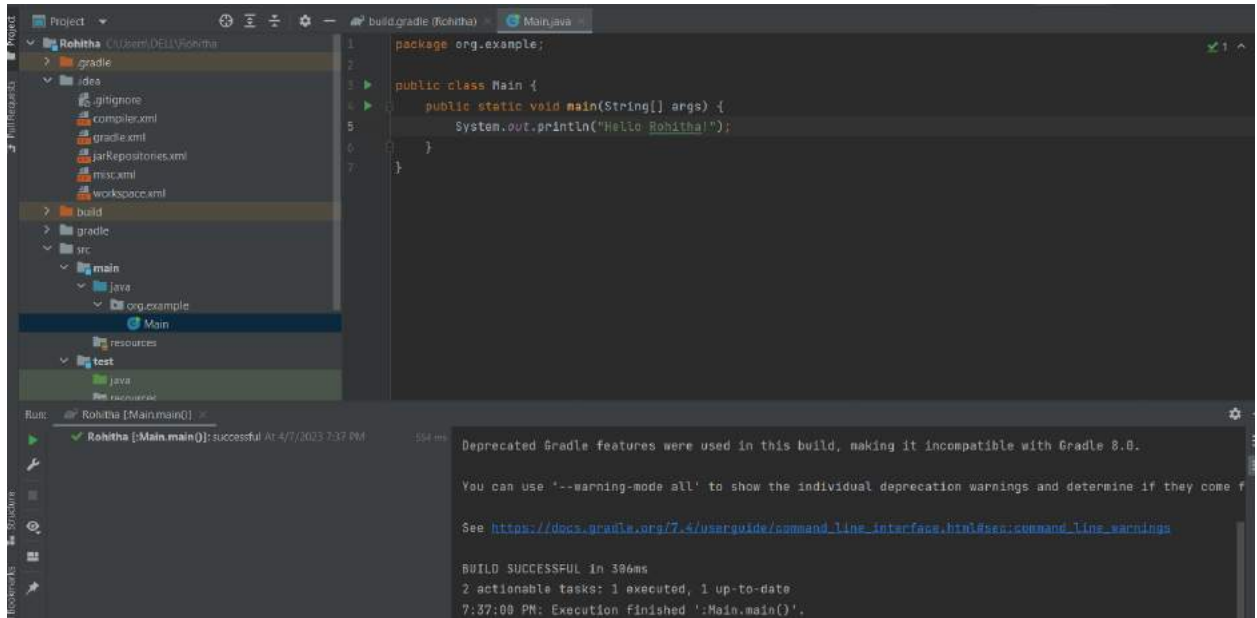


3. Run the jar as shown below and output is executed.

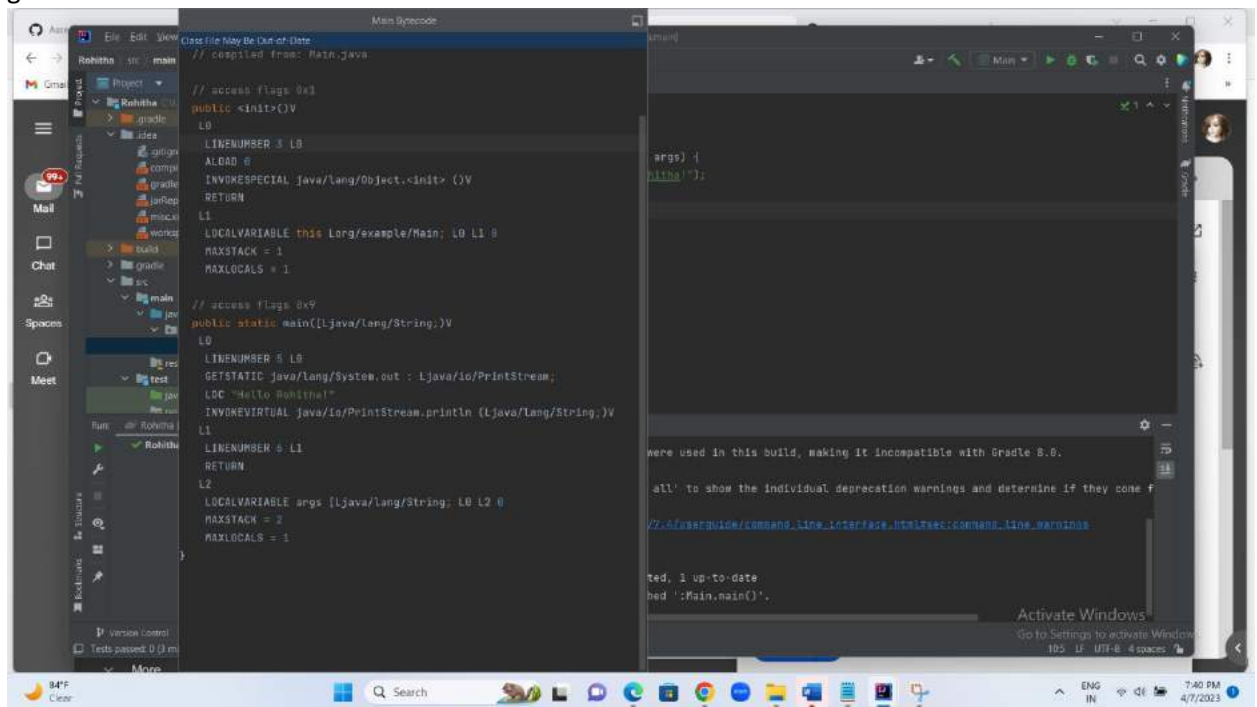


3. Java:

- 1) Task-1 Run a simple java application that prints the below statement Hello "your-name" !
- 2) Identify the Byte code for the same class and capture it.



- 3) In IntelliJ, click on the class name and from the View menu click on Show bytecode which generates as below:



4) Linux:

Log 1:

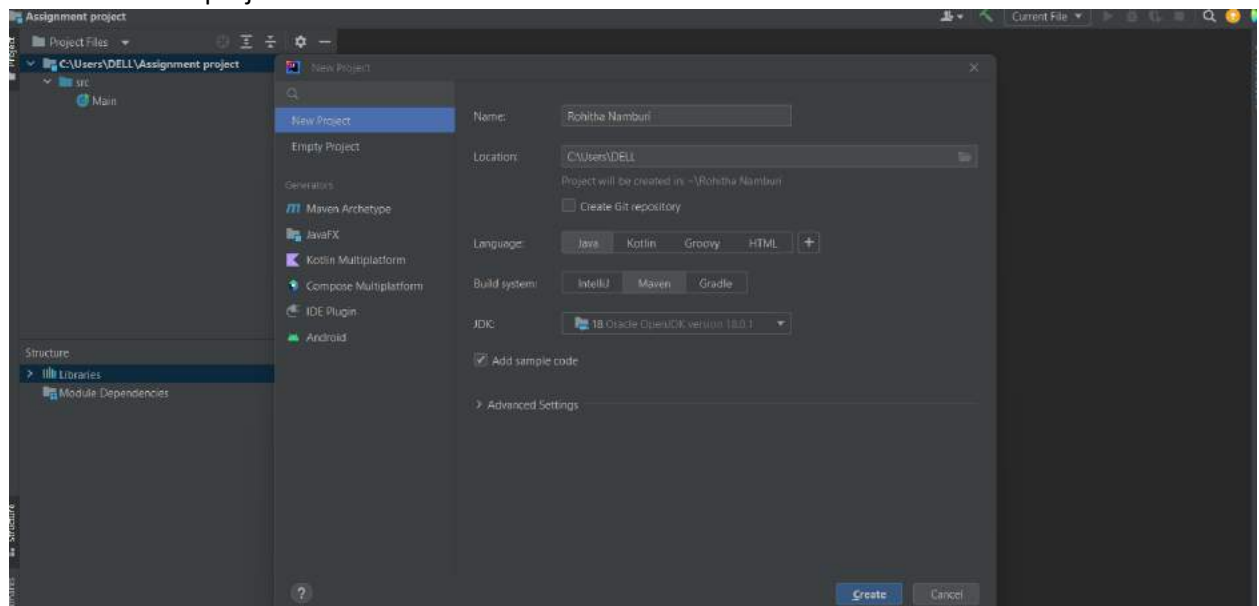
We need to run a command to get filter the log where ever we get an exception and line for the same and so that we can fix them.

1. Copy the application.log to a location on your drive, then use the command as shown in the screenshot to get all the errors that has the text "Exception"

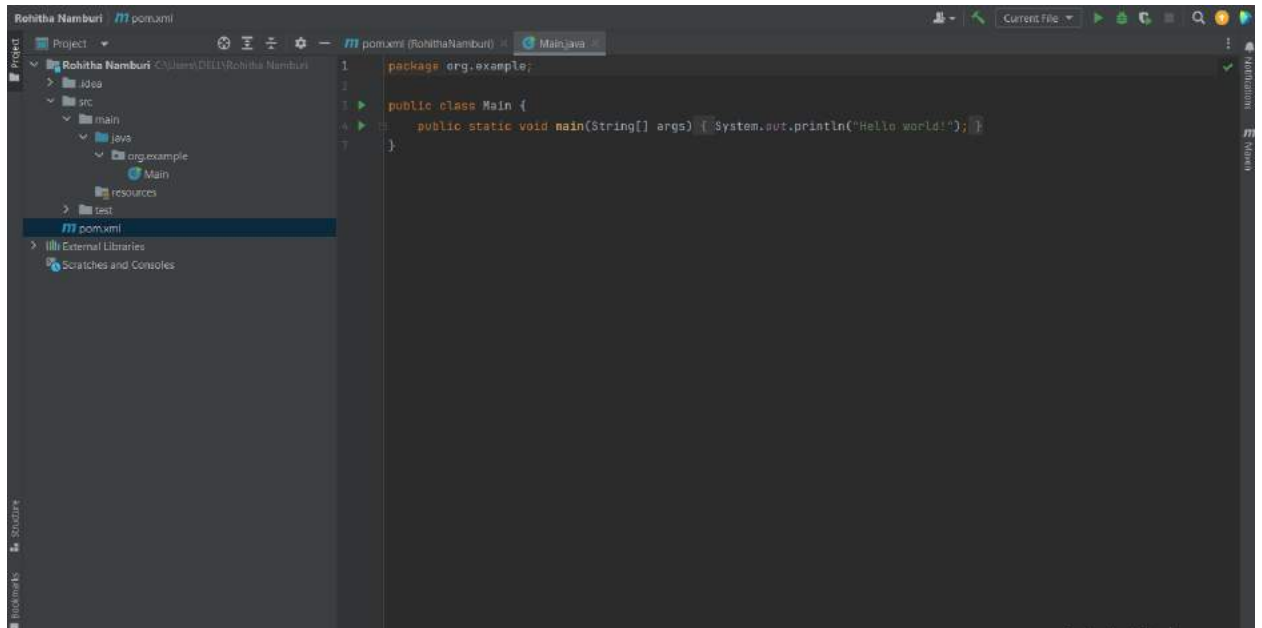

```
$ cat Application.log | grep "Exception"
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
Exception occurred connection time out(Connectivity.java line 53,121)
Exception occurred illegal sql grammer-(NotificationRepo.java insertNotification
method ,insert into table Notification line 23,54)
```

5. Maven

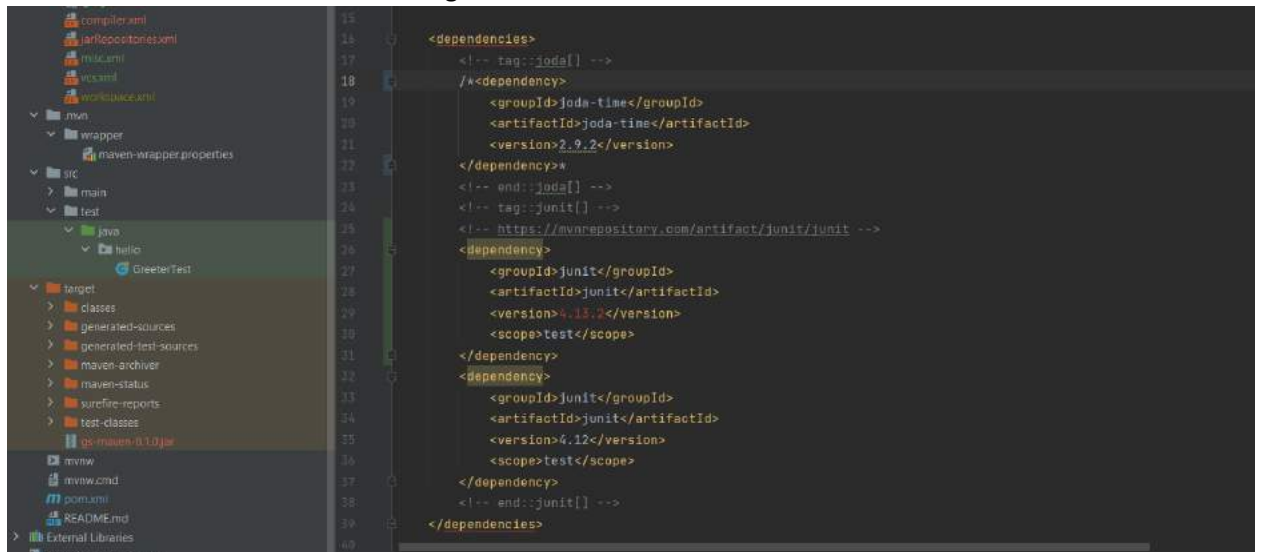
1. Create a Maven project as below:



2. To build the artifact from IntelliJ UI, open View->Tool Windows->Maven and you will see a menu like this: Click on package and hit the Run icon after expanding as below:



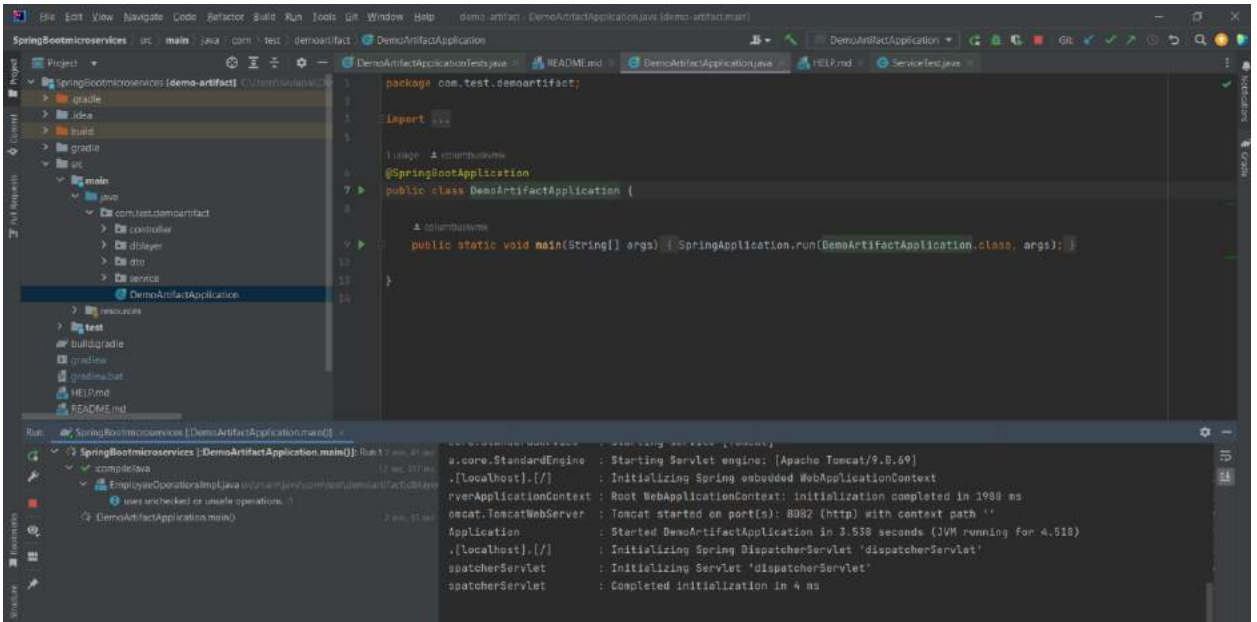
3. The artifact will be created under target folder as below:



6 Spring Boot Application:

Task-1 : Run the springBoot in the repo provided

1. Clone the repo <https://github.com/jntu-devops/SpringBootmicroservices> locally and open the project in IntelliJ. It will download all the dependencies. Now run the DemoArtifactApplication, it will start the service as shown in the screenshot below:

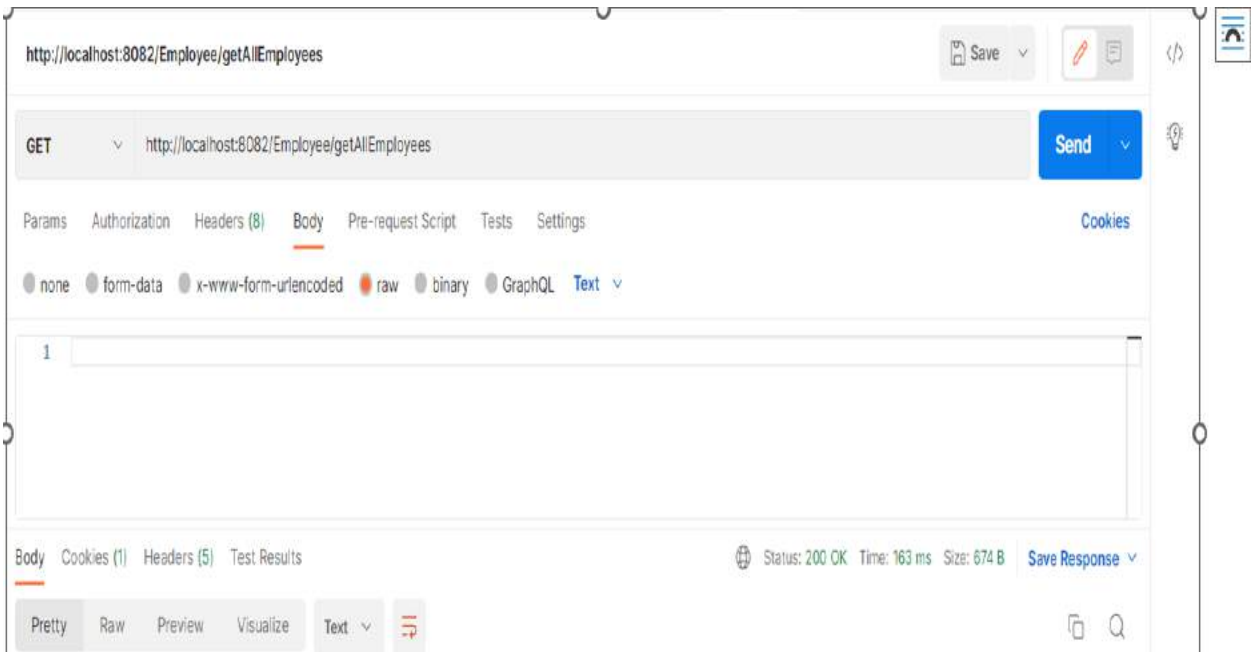


Task-2: Use postman to verify the results of the three api's

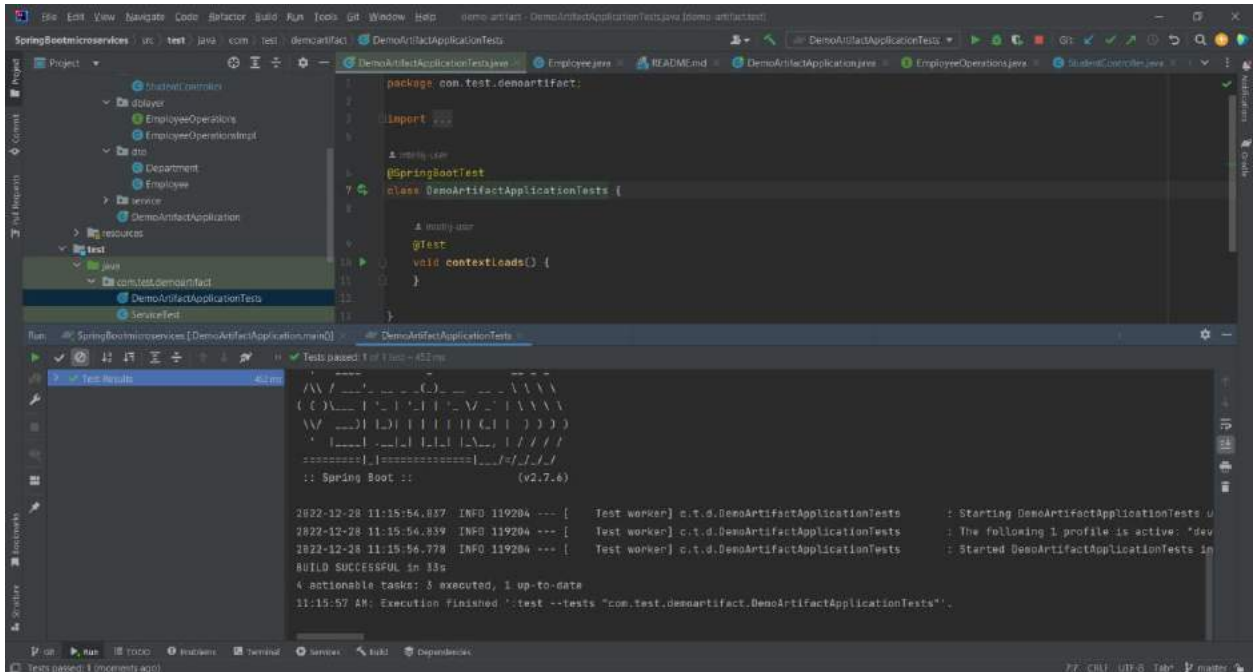
<http://localhost:8082/Employee/welcome>

<http://localhost:8082/Employee/getAllEmployees>

<http://localhost:8082/Employee/getEmployeeDetails>



Task-3 : Run any test-cases and capture the results



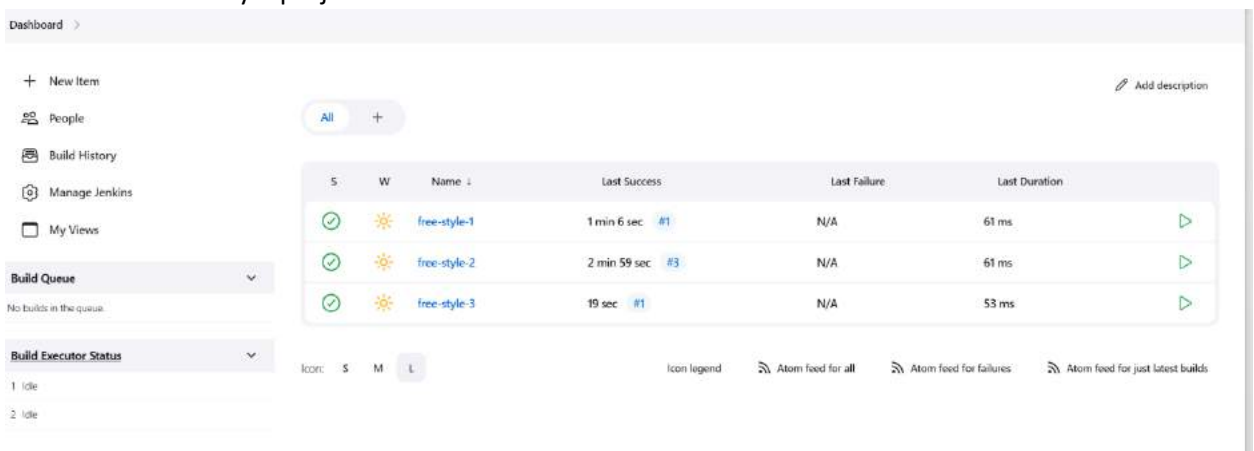
7. Jenkins

Task-1 : In Jenkins create 3 free style projects

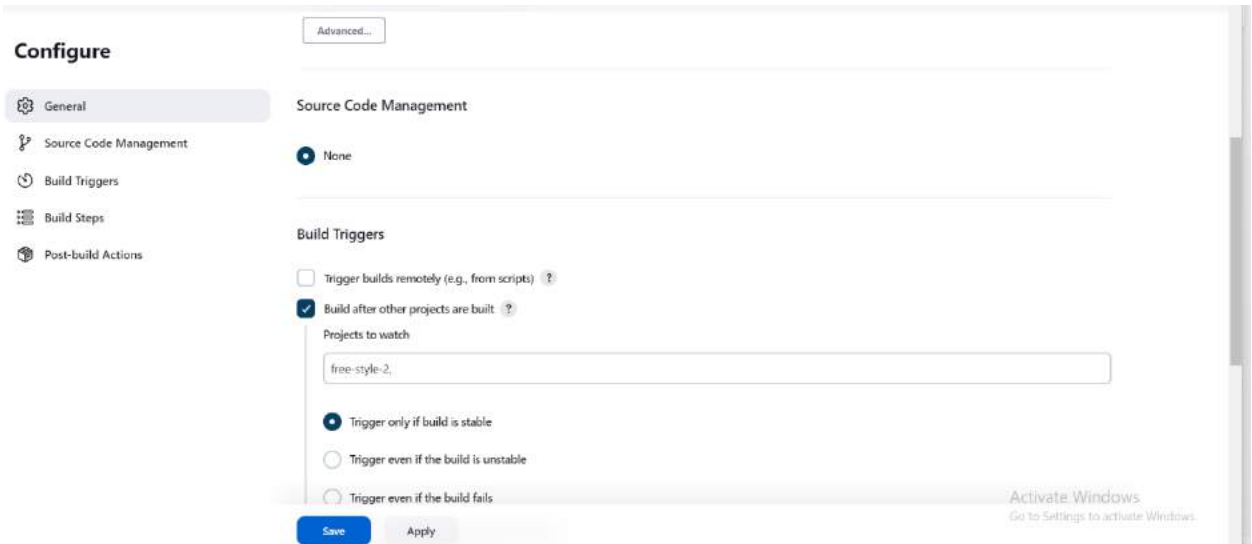
- free-style-1
- free-style-2
- free-style-3

once free-style-1 is build then free-style-2 should be build free-style-3

1. Create three free style projects as below:



2. Open FreeStyle-project2 project and configure it as below so it will be triggered once FreeStyle-project1 is built:



3. Similarly open FreeStyle-project3 and configure it as below so it will be triggered once FreeStyle-project2 is built.
4. Now view the build details of the first project as below, it indicates that a new build is triggered for second project: Similarly for all the projects it gives the console output.



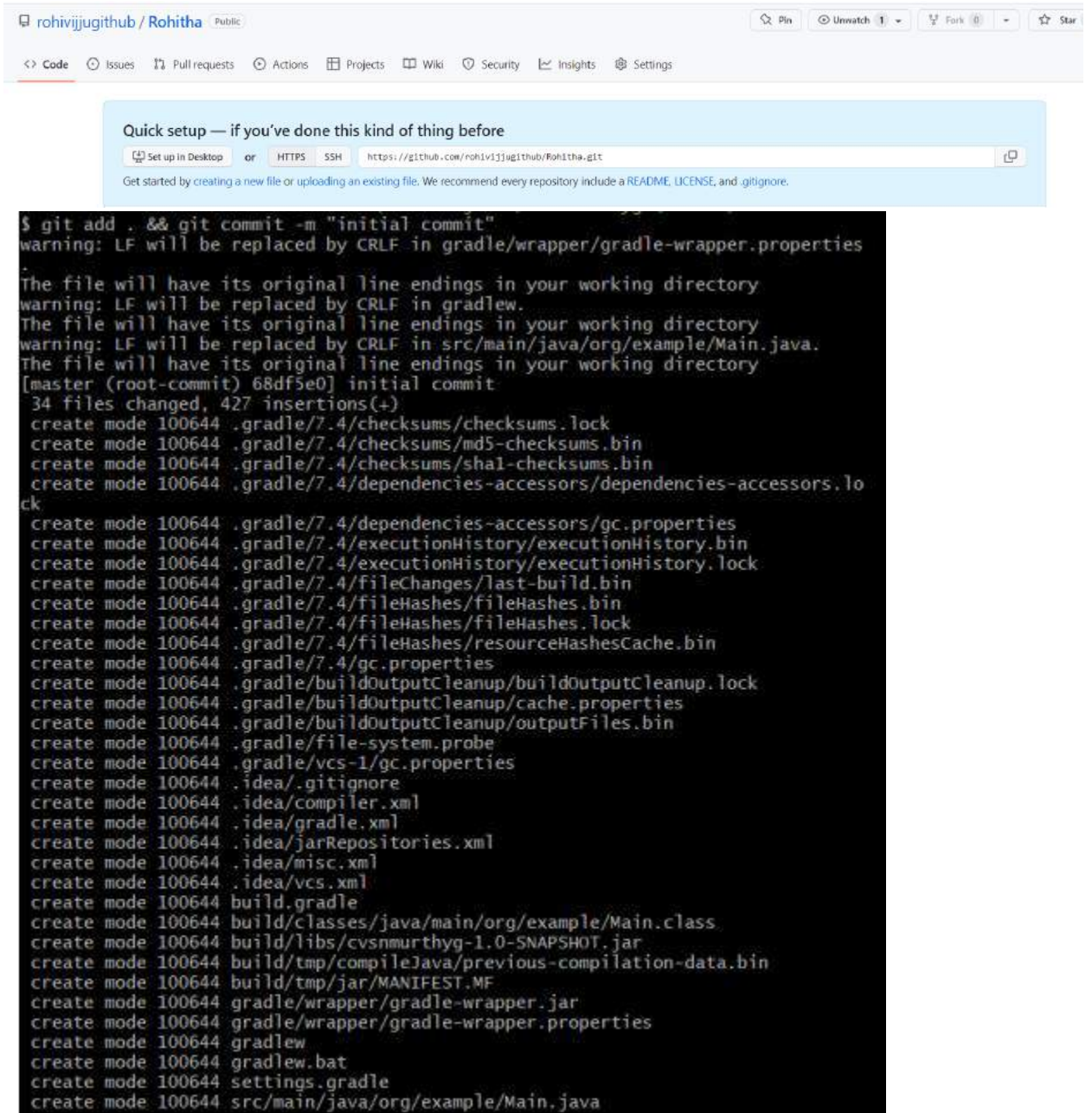
5. View the build details of the second project, it indicates that the build is triggered by the first project and a new build is triggered for third project.



Task-2 :

- Create a new repo
- Create a new Gradle / maven project and push into the newly create git repo
- Then configure in Jenkins the newly created project
- Whenever a new commit happens on the branch configure in Jenkins a new build should trigger.

1. Login to Git hub and create a new repo.



The screenshot shows a GitHub repository page for 'rohivijugithub/Rohitha'. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation is a 'Quick setup' section with options for 'Set up in Desktop', 'HTTPS', and 'SSH'. The SSH URL is 'https://github.com/rohivijugithub/Rohitha.git'. Below this is a terminal window showing the output of a 'git add' command. The terminal output includes warnings about line endings and a list of files that were created or changed during the commit.

```
$ git add . && git commit -m "initial commit"
warning: LF will be replaced by CRLF in gradle/wrapper/gradle-wrapper.properties
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in .
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/main/java/org/example/Main.java.
The file will have its original line endings in your working directory
[master (root-commit) 68df5e0] initial commit
34 files changed, 427 insertions(+)
 create mode 100644 .gradle/7.4/checksums/checksums.lock
 create mode 100644 .gradle/7.4/checksums/md5-checksums.bin
 create mode 100644 .gradle/7.4/checksums/sha1-checksums.bin
 create mode 100644 .gradle/7.4/dependencies-accessors/dependencies-accessors.lock
 create mode 100644 .gradle/7.4/dependencies-accessors/gc.properties
 create mode 100644 .gradle/7.4/executionHistory/executionHistory.bin
 create mode 100644 .gradle/7.4/executionHistory/executionHistory.lock
 create mode 100644 .gradle/7.4/fileChanges/last-build.bin
 create mode 100644 .gradle/7.4/fileHashes/fileHashes.bin
 create mode 100644 .gradle/7.4/fileHashes/fileHashes.lock
 create mode 100644 .gradle/7.4/fileHashes/resourceHashesCache.bin
 create mode 100644 .gradle/7.4/gc.properties
 create mode 100644 .gradle/buildOutputCleanup/buildOutputCleanup.lock
 create mode 100644 .gradle/buildOutputCleanup/cache.properties
 create mode 100644 .gradle/buildOutputCleanup/outputFiles.bin
 create mode 100644 .gradle/file-system.probe
 create mode 100644 .gradle/vcs-1/gc.properties
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/compiler.xml
 create mode 100644 .idea/gradle.xml
 create mode 100644 .idea/jarRepositories.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 build.gradle
 create mode 100644 build/classes/java/main/org/example/Main.class
 create mode 100644 build/libs/cvsnmurthyg-1.0-SNAPSHOT.jar
 create mode 100644 build/tmp/compileJava/previous-compilation-data.bin
 create mode 100644 build/tmp/jar/MANIFEST.MF
 create mode 100644 gradle/wrapper/gradle-wrapper.jar
 create mode 100644 gradle/wrapper/gradle-wrapper.properties
 create mode 100644 gradlew
 create mode 100644 gradlew.bat
 create mode 100644 settings.gradle
 create mode 100644 src/main/java/org/example/Main.java
```

2. Add the URL for the remote repository that is copied from git hub as below:

```

DELL@DESKTOP-IHTBG5V MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/DELL/.git/

DELL@DESKTOP-IHTBG5V MINGW64 ~ (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'
hint: Turn this message off by running
hint: "git config advice.addEmptyPaths false"

DELL@DESKTOP-IHTBG5V MINGW64 ~ (master)
$ git add new
fatal: pathspec 'new' did not match any files

DELL@DESKTOP-IHTBG5V MINGW64 ~ (master)
$ git remote add origin https://github.com/rohivijugithub/Rohitha.git

DELL@DESKTOP-IHTBG5V MINGW64 ~ (master)
$ .....

```

3. Push the project to remote repository.

```

$ git push -u origin main
Enumerating objects: 54, done.
Counting objects: 100% (54/54), done.
Delta compression using up to 12 threads
Compressing objects: 100% (32/32), done.
Writing objects: 100% (54/54), 64.04 KiB | 5.82 MiB/s, done.
Total 54 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.

```

4. Now go to GitHub remote repository in the browser and notice that the files are all added there as below:

📁 .gradle	initial commit	10 hours ago
📁 .idea	initial commit	10 hours ago
📁 build	initial commit	10 hours ago
📁 gradle/wrapper	initial commit	10 hours ago
📁 src/main/java/org/example	initial commit	10 hours ago
📄 build.gradle	initial commit	10 hours ago
📄 gradlew	initial commit	10 hours ago
📄 gradlew.bat	initial commit	10 hours ago
📄 settings.gradle	initial commit	10 hours ago

5. Download the executable for windows from <https://ngrok.com/download>
6. Execute "ngrok http 8080"
7. It will return a URL to use in the webhook in github as below:

```
C:\WINDOWS\system32\cmd.exe - ngrok.exe http 8080

ngrok

Add OAuth and webhook security to your ngrok (its free!): https://ngrok.com/free

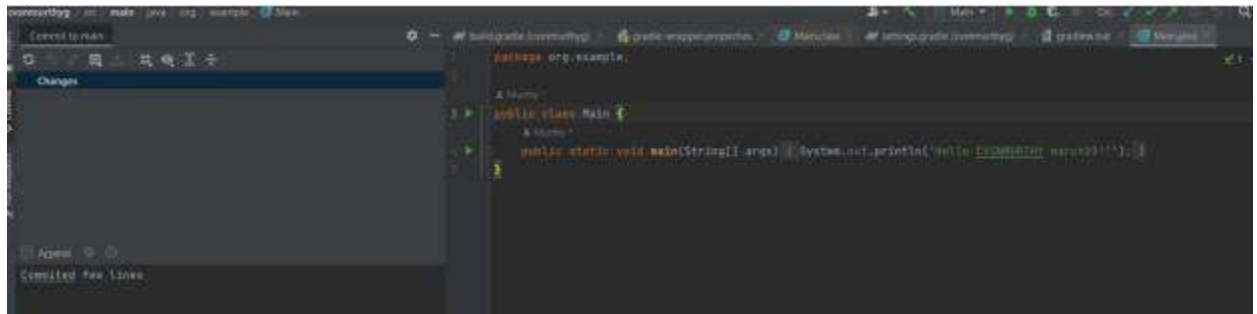
Session Status      online
Session Expires     1 hour, 59 minutes
Terms of Service     https://ngrok.com/tos
Version              3.2.2
Region              India (in)
Latency              -
Web Interface        http://127.0.0.1:4040
Forwarding           https://3bda-103-204-96-66.in.ngrok.io -> http://localhost:8080

Connections          ttl    opn    rt1    rt5    p50    p90
                    ---    ---    ---    ---    ---    ---
                    0      0      0.00  0.00  0.00  0.00
```

- Click on Settings on the repository on github and add a webhook for Jenkins as below:(You need to give the url from above, localhost will not work.
- Create a free style project in Jenkins and configure GIT configuration.



- Now make a change to code in the Gradle project, commit and push the changes:



- Now monitor Jenkins build and check the logs.

- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#4'
- Polling Log
- Git Build Data
- Previous Build

Console Output

Started by an SCM change

 Running as SYSTEM
 Building in workspace C:\Users\suvulupa1\.jenkins\workspace\SravaniVulupalaGitHubProject1
 The recommended git tool is: NONE
 using credential ffb0d31d-c4b1-42e3-e908-86de294a7728
 > git.exe rev-parse --resolve-git-dir C:\Users\suvulupa1\.jenkins\workspace\SravaniVulupalaGitHubProject1\.git # timeout=10
 Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/sravanireddy/SravaniVulupalaJenkinsRepo.git/ # timeout=10
 Fetching upstream changes from https://github.com/sravanireddy/SravaniVulupalaJenkinsRepo.git/
 > git.exe --version # timeout=10
 > git --version # 'git version 2.23.6.windows.1'
 using GIT_ASKPASS to set credentials Credentials for git
 > git.exe fetch --tags --force --progress -- https://github.com/sravanireddy/SravaniVulupalaJenkinsRepo.git/ +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
 Checking out Revision 73c604ccb04cf000002e1202a7b066d7ec1a310 (refs/remotes/origin/main)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 73c604ccb04cf000002e1202a7b066d7ec1a310 # timeout=10
 Commit message: "Update Main.java"
 > git.exe rev-list --no-walk 10040ccc05403f9c934c5e91076ad005f009ec04 # timeout=10
 [SravaniVulupalaGitHubProject1] \$ cmd /c call C:\Users\suvulupa1\AppData\Local\Temp\jenkins365780325701237413.bat

 C:\Users\suvulupa1\.jenkins\workspace\SravaniVulupalaGitHubProject1>gradlew clean
 > Task :clean

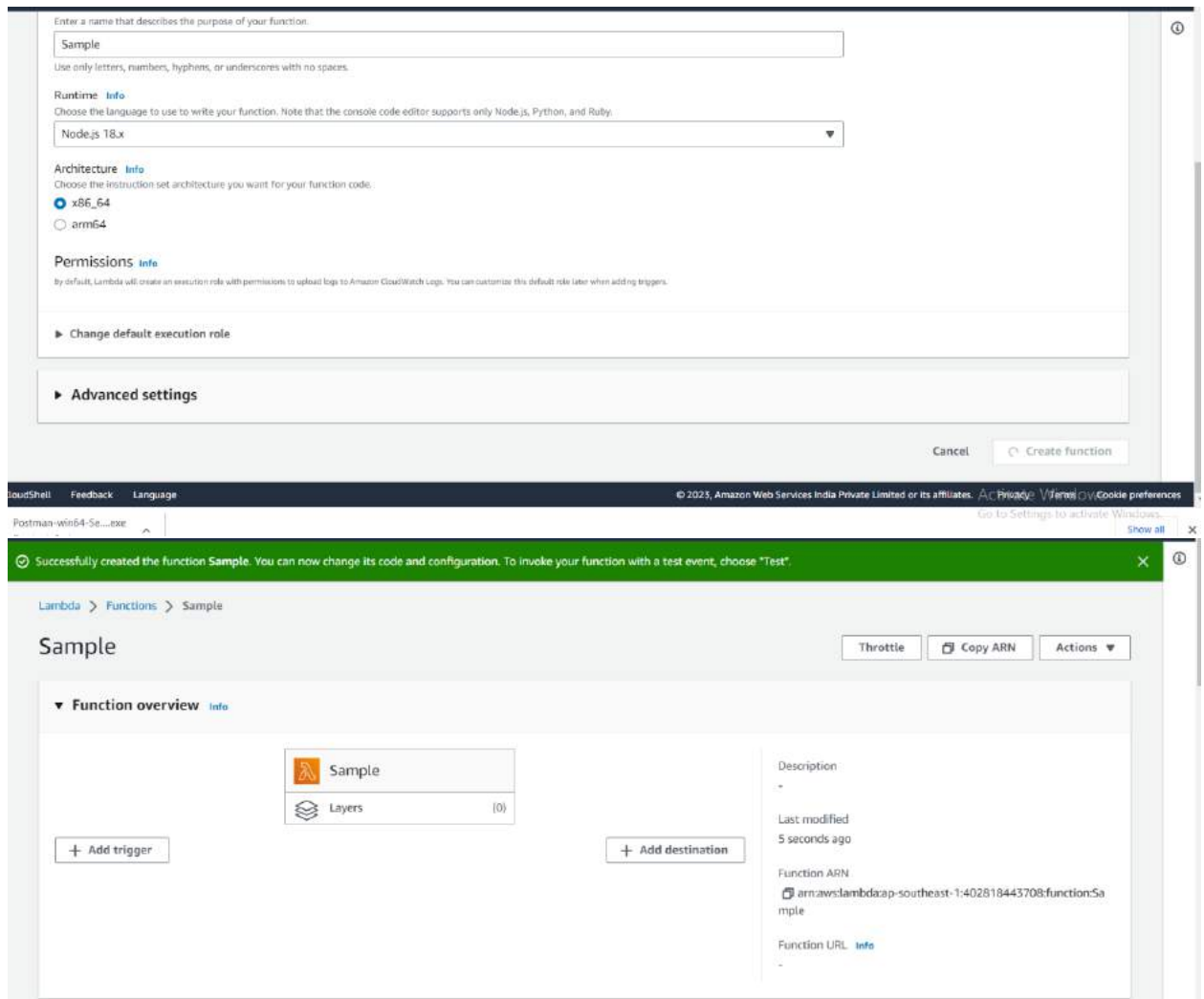
 BUILD SUCCESSFUL in 3s
 1 actionable task: 1 executed
 Finished: SUCCESS

8. AWS SQS and Lambda:

1. AWS Lambda:

Create a lambda function that prints sample as below:

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Services', 'Search', and '[Alt+S]'. The main content area is titled 'Resources for Asia Pacific (Singapore)' and includes a 'Create function' button. Below this, a summary card shows: 1 Lambda function(s), 310.0 byte (0% of 75.0 GB) Code storage, 10 Full account concurrency, and 10 Unreserved account concurrency. Underneath, there's an 'Account-level metrics' section with three charts: 'Error count and success rate (%)', 'Throttles', and 'Invocations'. All three charts display 'No data available. Try adjusting the dashboard time range.' The bottom of the screenshot shows a 'CloudShell' terminal window with a Postman request and a footer with copyright information for Amazon Web Services India Private Limited.



2. Create Lambda application as below:

Create a Lambda application

An AWS Lambda application is a combination of Lambda functions, triggers, and other resources that work together to perform tasks. You can create an application with a continuous integration and continuous delivery (CI/CD) pipeline or from a serverless application in the AWS Serverless Application Repository.

[CI/CD pipeline application](#) | [Serverless application](#)

From scratch

Create a repository and pipeline to use with your own application.

Runtime:
Node.js 14.x, Node.js 16.x, Node.js 18.x

Uses:
Lambda

Serverless API backend

A RESTful web API that uses DynamoDB to manage state.

Runtime:
Node.js 14.x

Uses:
API Gateway, DynamoDB, Lambda

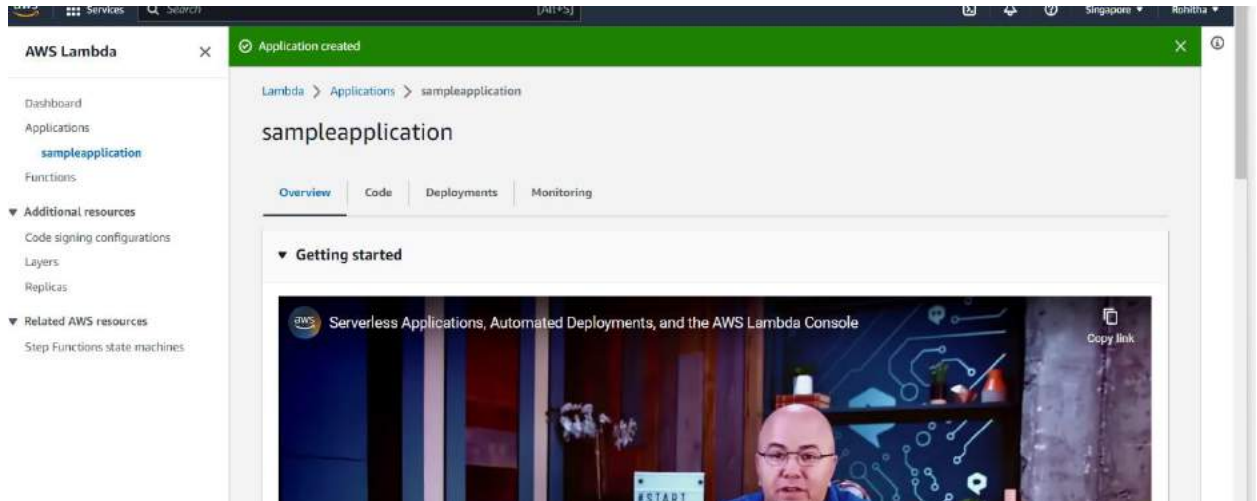
File processing

© 2025, Amazon Web Services India Private Limited or its affiliates. [Ac Privacy](#) [Terms of](#) [Cookie preferences](#)

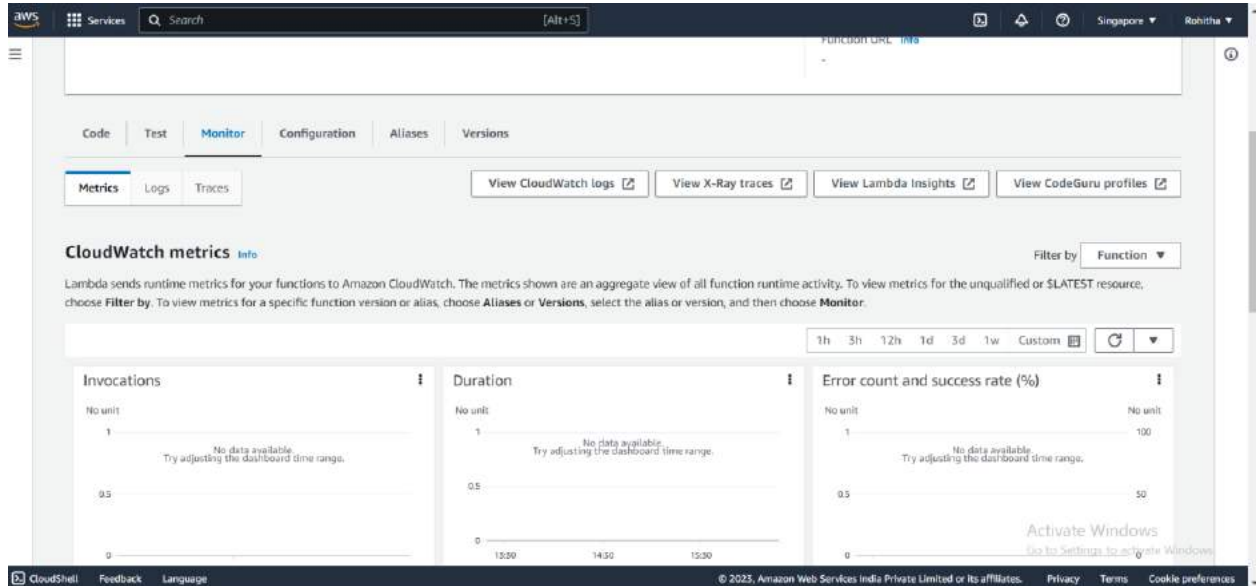
[Go to Settings to activate Windows](#)

[Show all](#)

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and the user's name 'Rohit'. Below this, a blue banner indicates 'Creating application resources. This takes a few minutes. Step 2 of 4 - Creating pipeline'. The main content area is titled 'sampleapplication' and has tabs for 'Overview', 'Code', 'Deployments', and 'Monitoring'. Under the 'Overview' tab, there's a 'Getting started' section with a video thumbnail titled 'Serverless Applications, Automated Deployments, and the AWS Lambda Console'. The video thumbnail shows a person speaking in front of a screen with technical diagrams. At the bottom of the console, there's a footer with '© 2025, Amazon Web Services India Private Limited or its affiliates. Ac Privacy Terms of Cookie preferences' and a 'Go to Settings to activate Windows' notification. The Windows taskbar at the very bottom shows 'CloudShell', 'Feedback', 'Language', and 'Postman-win64-Se..._exe'.



3. Capture the logs in cloud watch logs as below:



2. AWS SQS:

1. Create a SQS as below:

Amazon SQS > Queues > Create queue

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

Standard info
 At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO info
 First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

ⓘ You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens [-], and underscores [_].

Configuration

Queue Rohitha created successfully
You can now send and receive messages.

Amazon SQS > Queues > Rohitha

Rohitha

[Edit](#) [Delete](#) [Purge](#) [Send and receive messages](#) [Start DLQ redrive](#)

Details info

Name Rohitha	Type Standard	ARN arn:aws:sqs:ap-southeast-1:402818443708:Rohitha
Encryption Amazon SQS key (SSE-SQS)	URL https://sqs.ap-southeast-1.amazonaws.com/402818443708/Rohitha	Dead-letter queue -

[More](#)

[SNS subscriptions](#) [Lambda triggers](#) [Dead-letter queue](#) [Monitoring](#) [Tagging](#) [Access policy](#) [Encryption](#) [Dead-letter queue redrive tasks](#)

Subscription region
ap-southeast-1

2. Create Lambda function as below:

Successfully created the function Sample. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > Sample

Sample

[Throttle](#) [Copy ARN](#) [Actions](#)

Function overview info

Sample
 Layers (0)

[+ Add trigger](#) [+ Add destination](#)

Description
-

Last modified
5 seconds ago

Function ARN
arn:aws:lambda:ap-southeast-1:402818443708:function:Sample

Function URL [info](#)
-

3. Add subscription of the SQS to the lambda.

