



Deploy a highly available Sprint application

- 1.  Revision History
- 2.  Resources
- 3. Pipeline Implementations steps
 - 1. Repository creation
 - 2. AWS Pipeline implementation
 - 1. Implementations steps
 - 1. The Pipeline stages should be as follows:
 - 2. Fetch repos and configure buildspec.yaml
- 4. Architecture diagram
- 5. Pre-Code Preparation
- 6. Gradle application build steps
- 7. Python application build steps

Revision History

Version	Date	Description	Author
1.0	03/10/2024	Initial version	Mohamed El Eraki
1.2	03/10/2024	Append pre-code section	Mohamed El Eraki
1.3	04/10/2024	Append Gradle build steps	Mohamed El Eraki
1.4	05/10/2024	Append Python build steps	Mohamed El Eraki

Resources

-  Architecture Diagram.drawio
-  Mohamed-Eleraki/spring-boot-app

Pipeline Implementations steps

Repository creation

Create a GitHub public repository called `spring-boot-app` as follows *Screenshot-01*

Create a new repository

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

Required fields are marked with an asterisk ().*

Owner *

Mohamed-Eleraki


Repository name *


spring-boot-app

✔ spring-boot-app is available.

Great repository names are short and memorable. Need inspiration? How about **solid-fiesta** ?

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.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)


Choose a license


License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

Grant your Marketplace apps access to this repository
You are subscribed to 1 Marketplace app.

☒  **Azure Pipelines** *(auto-installed)*
Continuously build, test, and deploy to any platform and cloud

 You are creating a public repository in your personal account.

Create repository

Screenshot-01

AWS Pipeline implementation

Craft an AWS pipeline integrated with The GitHub repository created, the pipeline will be triggered automatically based on tags assigned with the pushed commits that include "release*" value.

The pipeline will include multiple stages as the fetching source code stage, Plan stage, Manual approval stage, and Apply stage.

Implementations steps

- Open-up AWS console, in the search bar search for **CodePipeline**.
- From the CodePipeline main page press **Create Pipeline**.
- Set **pipeline name** as `eraki_springApp_us1_pipeline`.
- Set the **Execution mode** as **Superseded** avoiding overlapping deployments, leave all as defaults then **Next**.
- For **Source provider** specify **GitHub version 2**, Then **Connect to GitHub**.
- on the Pop-up page, for **Connection name** set `eraki_springApp_us1_connection`, Then **Connect to GitHub**.

- press **Install a new app**, This will open-up the GitHub configuration page, specify the created branch, Or you can set it for all repositories.

Repository access

☐ **All repositories**
This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

☒ **Only select repositories**
Select at least one repository.
Also includes public repositories (read-only).

Select repositories ▾

Selected 2 repositories.

Mohamed-Eleraki/spring-boot-app	X
Mohamed-Eleraki/terraform	X

Save Cancel

Screenshot-02

- Press **Save**, Will revert you back to the connection creation page, press **connect**.

i You should see **Ready to connect** as follows

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codeconnections:us-east-1:891377122503:connection/51c42945-9a X **or** **Connect to GitHub**

✓ Ready to connect
Your GitHub connection is ready for use.

Screenshot-03

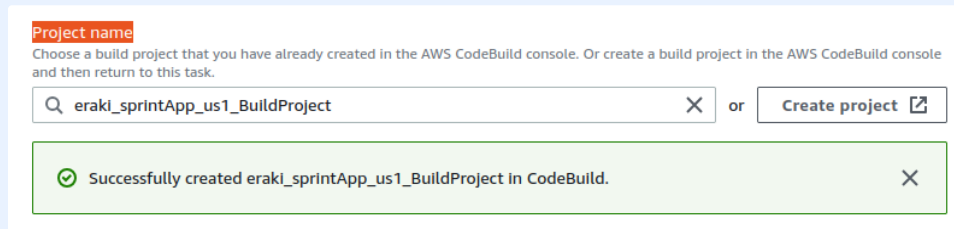
- on the list of **Repository name**, specify “spring-boot-app” repository.
- on the list of **Default branch**, specify “main”.
- For **Triggers**, Set **Trigger type** as Specify filter, **Event type** as push, and **Filter type** as Tags and for **Include** type “release-*” Then **Next**.

i this ensures that the pipeline will fireup when a commit include **release-*** instead of setup it for a specific branch

- In **Build Stage** for **Build provider** Specify AWS **CodeBuild**.
- For **Project name** Press **Create project**.
- On the Pop-up page, for **Project name** set “eraki_sprintApp_us1_BuildProject”
- Leave all as default, and for **Build spec** specify “Use a builds spec file” Then **continue to pipeline**.

i here we’re specifying the machine specs that the pipeline stages will run on it.

The result should be as follows:



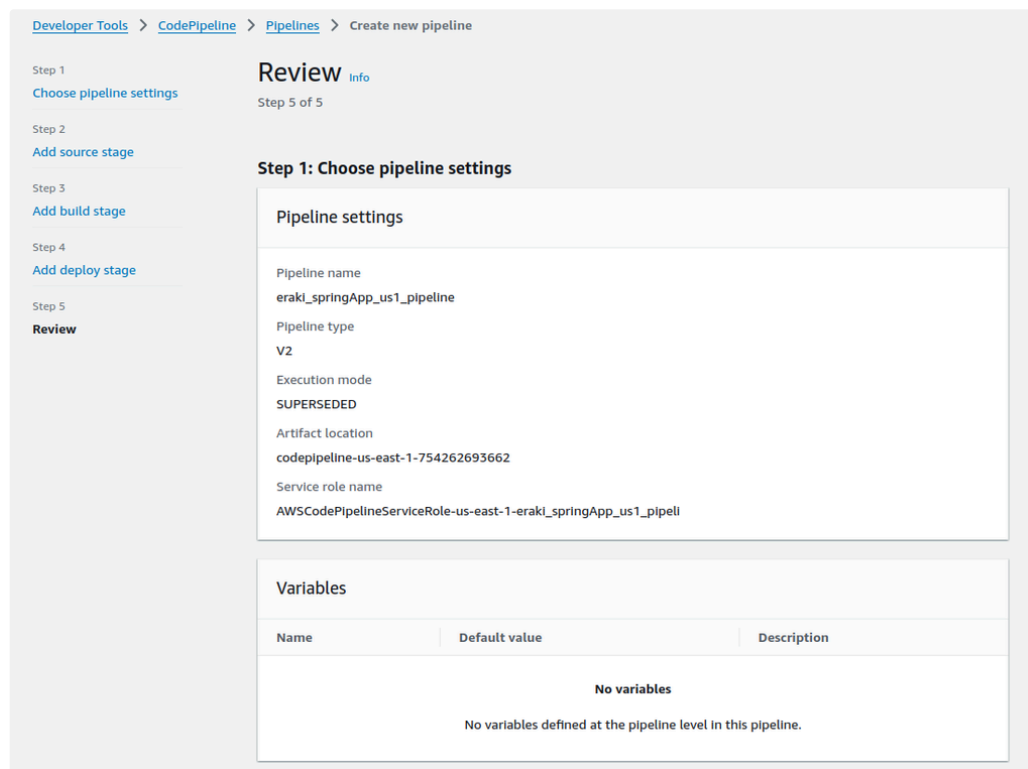
Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

Q eraki_sprintApp_us1_BuildProject X or Create project

✓ Successfully created eraki_sprintApp_us1_BuildProject in CodeBuild. X

Screenshot-04

- Press **Next**, and **skip deploy stage**.
- Review and Press **Create pipeline**, The pipeline configurations should be as follows *Screenshot-05*, *Screenshot-06*, and *Screenshot-07*



Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1
Choose pipeline settings

Step 2
Add source stage

Step 3
Add build stage

Step 4
Add deploy stage

Step 5
Review

Review Info

Step 5 of 5

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name
eraki_springApp_us1_pipeline

Pipeline type
V2

Execution mode
SUPERSEDED

Artifact location
codepipeline-us-east-1-754262693662

Service role name
AWSCodePipelineServiceRole-us-east-1-eraki_springApp_us1_pipeli

Variables

Name	Default value	Description
No variables		
No variables defined at the pipeline level in this pipeline.		

Screenshot-05

Step 2: Add source stage

Source action provider

Source action provider

GitHub (Version 2)

OutputArtifactFormat

CODE_ZIP

DetectChanges

false

ConnectionArn

arn:aws:codeconnections:us-east-1:891377122503:connection/51c42945-9a8b-466e-b1b8-aa0c11689607

FullRepositoryId

Mohamed-Eleraki/spring-boot-app

Default branch

main

Trigger configuration

You can add additional pipeline triggers after the pipeline is created.

Trigger type

Specify filter

Event type

Push

Filter type

Tags

Include tags

release-*

Exclude tags

-

Screenshot-06

Step 3: Add build stage

Build action provider

Build action provider

AWS CodeBuild

ProjectName

eraki_sprintApp_us1_BuildProject

Step 4: Add deploy stage

Deploy action provider

Deployment stage

No deploy

Cancel

Previous

Create pipeline

Screenshot-07

Now, that we have created the pipeline, let's create the pipeline stages in the following steps.

- Open your created pipeline, Then **Edit**.
- Edit **Build stage** as follows.

Create a variable for your stages so you can differentiate them into your `buildspec.yml` file. The plan stage will include **terraform plan** command.

The screenshot shows the 'Edit action' dialog for a Build stage. The 'Action name' is 'PlanStage'. The 'Action provider' is 'AWS CodeBuild'. The 'Region' is 'US East (N. Virginia)'. The 'Input artifacts' section shows 'SourceArtifact'. The 'Project name' is 'eraki_sprintApp_vst_BuildProject'. The 'Environment variables - optional' section shows a table with 'Name' as 'STAGE_TYPE', 'Value' as 'plan', and 'Type' as 'Plaintext'. The 'Build type' is 'Single build'. The 'Variable namespace - optional' section shows 'BuildVariables'. The 'Output artifacts' section is empty.

Screenshot-08

- Under build stage, **Add stage**.

The screenshot shows the 'Edit: Build' stage configuration. The 'Conditions' section shows 'Entry: Not configured', 'Success: Not configured', and 'Failure: Not configured'. The 'Build' section shows 'AWS CodeBuild'. The '+ Add stage' button is highlighted with a red box.

Screenshot-09

- For **stage name** set Manual_Approval.

The screenshot shows the 'Edit: Manual_Approval' stage configuration. The 'Add entry condition', 'Add success condition', and 'Add failure condition' buttons are visible. The '+ Add action group' button is highlighted with a red box. The 'Configure automatic rollback on stage failure' checkbox is unchecked. The '+ Add stage' button is visible at the bottom.

Screenshot-10

Edit action

Action name
Choose a name for your action
Manual_Approval

Action provider
Manual approval

Configure the approval request.

SNS topic ARN - optional
arn:aws:sns:us-east-1:1091377122503:erak_topic_us1_pipelineMailNotifications

URL for review - optional
Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

Comments - optional
Comments you type here display for the reviewer in email notifications or the console.
Please review and approve this action

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel Done

Screenshot-11

i I have configured an SNS topic holds my e-mail for reciving notifications from pipeline

- Add a new stage for applying terraform code as follows.

Create a variable for your stages so you can differentiate them into your `buildspec.yml` file. The apply stage will include **terraform** **apply** command.

Edit action

Action name
Choose a name for your action
Apply_stage

Action provider
AWS CodeBuild

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)
BuildArtifact

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
erak_sprintApp_us1_BuildProject or Create project

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Name	Value	Type
STAGE_TYPE	apply	Plaintext

Add environment variable

Build type
☒ Single build
☐ Batch build

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts
Choose a name for the output of this action.

Screenshot-12

The Pipeline stages should be as follows:

Developer Tools > CodePipeline > Pipelines > eraki_springApp_us1_pipeline

eraki_springApp_us1_pipeline

Pipeline type: V2 Execution mode: SUPERSEDED

🟢 **Source** Succeeded

Pipeline execution ID: [bf803420-57a2-49e2-8d7d-a08abcb5bf1f](#)

Source

[GitHub \(Version 2\)](#)

🟢 Succeeded - 39 minutes ago

[45bc3b90](#)

View details

[45bc3b90](#) Source: Initial commit

Disable transition

🔴 **Build** Failed

Pipeline execution ID: [bf803420-57a2-49e2-8d7d-a08abcb5bf1f](#)

Plan_Stage

[AWS CodeBuild](#)

⏸ Didn't Run

No executions yet

[45bc3b90](#) Source: Initial commit

Disable transition

⏸ **Manual_Approval** Didn't Run

Manual_Approval

Manual approval

⏸ Didn't Run

No executions yet

Disable transition

⏸ **Apply_stage** Didn't Run

Apply_stage

[AWS CodeBuild](#)

⏸ Didn't Run

No executions yet

Fetch repos and configure `buildspec.yaml`

Now, let's configure the pipeline line YAML file `buildspec.yaml` as follows.

- Clone down the repo into your local by running the following command:

```
1 git clone https://github.com/Mohamed-Eleraki/spring-boot-app.git
```

- Open the repo into your editor, and create the `buildspec.yaml` file as follows:

```
1 version: 0.2
```



```
2 env:
3   variables:
4     TERRAFORM_VERSION: "1.5.6"
5     #key: "value"
6   #
7   # parameter-store:
8   #   key: "value"
9   #
10  # secrets-manager:
11  #   key: "value"
12
13  phases:
14    install:
15      runtime-versions:
16        python: 3.12
17
18      on-failure: ABORT
19      commands: |
20        echo "Installing terraform"
21        yum install -y wget unzip
22        yum clean all
23        tf_version=$TERRAFORM_VERSION
24        wget
25        https://releases.hashicorp.com/terraform/"$TERRAFORM_VERSION"/terraform_"$TERRAFORM_VERSION"_linux_amd64.zip
26        unzip terraform_"$TERRAFORM_VERSION"_linux_amd64.zip
27        chmod 775 terraform
28        mv terraform /usr/local/bin/
29        terraform --version
30        rm terraform_"$TERRAFORM_VERSION"_linux_amd64.zip
31        ls -al /usr/local/bin/terraform
32
33  pre_build:
34    on-failure: ABORT
35    commands: |
36      if [ "$STAGE_TYPE" = "plan" ]; then
37        echo "Filter on Plan stage";
38        echo terraform plan started on `date`
39        #cd "CODEBUILD_SRC_DIR/AWS_Demo/37-build-test-pipeline";
40        cd "infra";
41        ls -lathr;
42        terraform init;
43        terraform validate;
44        terraform plan -out tfplan;
45      else
46        echo "No Plan stage";
47      fi
48
49  build:
50    on-failure: ABORT
51    commands: |
52      if [ "$STAGE_TYPE" = "apply" ]; then
53        echo "Filter on Apply stage";
54        echo terraform execution started on `date`;
55        ls -lathr;
56        #cd "$CODEBUILD_SRC_DIR/AWS_Demo/37-build-test-pipeline";
57        cd "infra";
58        ls -lathr;
59        terraform apply tfplan;
```

```

59     else
60         echo "No Apply stage";
61     fi
62
63     post_build:
64         on-failure: CONTINUE
65         commands: |
66             echo "Fetching provisioning details"
67             terraform show -json tfplan > tfplan.json
68             yum install -y jq
69             echo "print out terraform version and json format version"
70             jq '.terraform_version, .format_version' tfplan.json
71             echo ""
72             echo "print out provider config"
73             jq '.configuration.provider_config' tfplan.json
74             echo ""
75             echo "print out resource config"
76             jq '.configuration.root_module.resources' tfplan.json
77             echo ""
78             echo "print out outputs"
79             jq '.outputs' tfplan.json
80             echo ""
81             echo "print out resource changes"
82             jq '.resource_changes' tfplan.json
83             echo ""
84             echo "print out resource config"
85             jq '.configuration.root_module.resources' tfplan.json
86             echo ""
87             echo "print out provider config"
88             jq '.configuration.provider_config' tfplan.json
89             echo ""
90             echo "print out provider config"
91             jq '.configuration.provider_config' tfplan.json
92             echo ""
93             echo "print out lock file configuration"
94             jq '.configuration.lock_version' tfplan.json

```

now the pipeline is ready, and will be triggered by setting a tag on commit using the following example command:

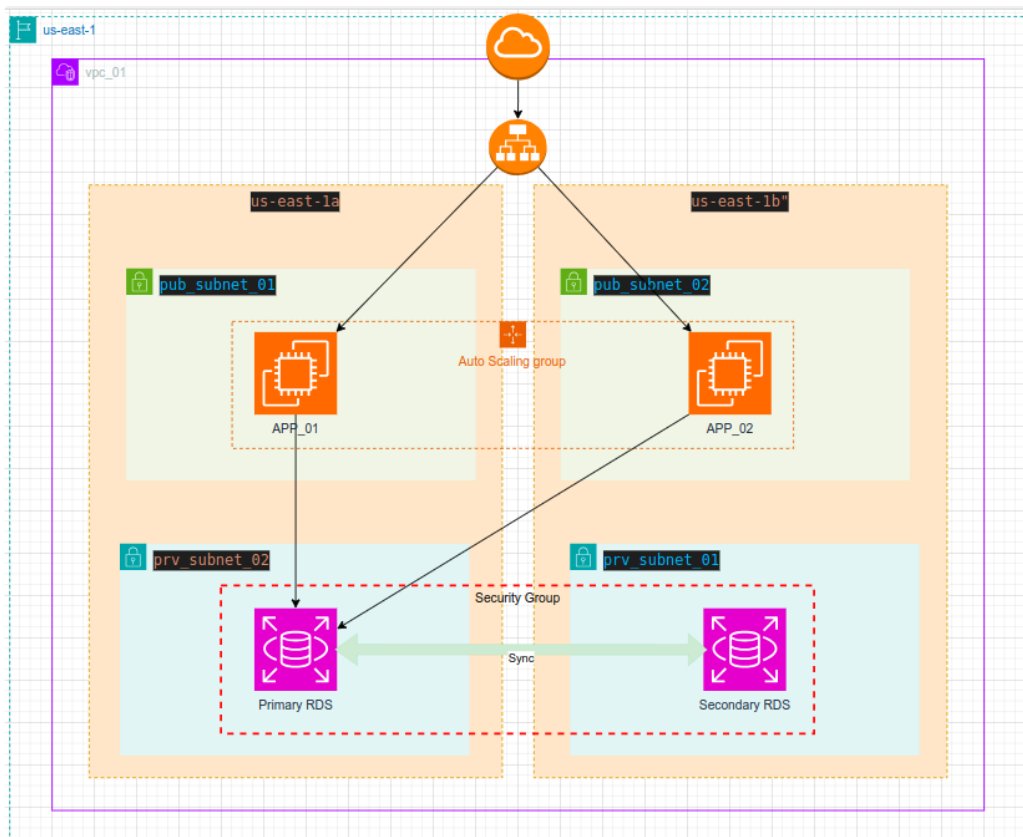
```

1 git log # to get the commit hash
2 git tag release-1 1234 # 1234 refer to commit has
3 git push origin release-test01 # push the applied tag

```

Architecture diagram

build a highly available application using autoscaling group and load balancer besides building a PostgreSQL RDS that is highly available as an RDS Multi-az instance.



Pre-Code Preparation

Here we go, will build up the entire infrastructure as the diagram above in the following repository: <https://github.com/Mohamed-Eleraki/spring-boot-app>

- Create a Dev Branch, deploying the infrastructure components.

```

/spring-boot-app$ git push origin release-test01
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Mohamed-Eleraki/spring-boot-app.git
 * [new tag]         release-test01 -> release-test01

/spring-boot-app$ git checkout -b Dev
Switched to a new branch 'Dev'

/spring-boot-app$ git branch -a
* Dev
  main
remotes/origin/HEAD -> origin/main
remotes/origin/main

/spring-boot-app$

```

Gradle application build steps

To build Gradle first you need to install the compatible Java jdk version using the following commands:

Check the compatible version of Gradle by viewing the following file

```
1 $ tree
2 .
3 ├── build.gradle
4 ├── gradle
5 │   └── wrapper
6 │       ├── gradle-wrapper.jar
7 │       └── gradle-wrapper.properties
8 ├── gradlew
9 ├── gradlew.bat
10 ├── HELP.md
11 ├── settings.gradle
12 └── src
13     ├── main
14     │   ├── java
15     │   │   └── com
16     │   │       └── example
17     │   │           └── demo
18     │   │               └── DemoApplication.java
19     │   └── resources
20     │       └── application.properties
21     └── test
22         ├── java
23         │   ├── com
24         │   │   ├── example
25         │   │   └── demo
26         │   └── DemoApplicationTests.java
27
28 14 directories, 10 files
29
```

```
1 grep "sourceCompatibility" build.gradle
2 sourceCompatibility = '11'
```

Install the required version

Ubuntu

```
1 sudo apt-get install -y openjdk-11-jdk
```

Amazon linux 2

```
1 sudo yum install -y java-11-openjdk-devel
```

ensure the required Java version is set successfully

```
1 java --version
```

if not set to 11, alternative by utilizing:

```
1 sudo update-alternatives --config java
```

ensure the JAVE_HOME variable is configured to use the Jave compatible version

- Check the variable value

```
1 $JAVE_HOME
```

- Adjust it to use Java 11 version

```
1 export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Now, you're ready to build your Gradle application as follows:

```
1 ./gradlew build
```

Run and test your application:

```
1 java -jar build/libs/<your-jar-file>.jar
2
3 Or with a specific port
4 java -jar build/libs/demo-0.0.1-SNAPSHOT.jar --spring.profiles.active=dev
5
6 Or
7 ./gradlew bootRun
```

Check application default port configuration:

```
1 cat src/main/resources/application.properties
2
3 # Server configuration
4 server.port=8080
5 server.address=0.0.0.0
6
7 # Database configuration
8 spring.datasource.url=jdbc:postgresql://<rds-endpoint>:5432/<database-name>
9 spring.datasource.username=<your-username>
10 spring.datasource.password=<your-password>
11 spring.datasource.driver-class-name=org.postgresql.Driver
12
13 # JPA / Hibernate settings (optional, but commonly used)
14 spring.jpa.hibernate.ddl-auto=update
15
16 spring.profiles.active=default
```

you should rebuild after adjusting this file!

To stop Gradle, kill the process

```
1 ps aux | grep -i gradle
2 kill -9 <PID>
```

- ✗ the application has run successfully as below screenshots. However, the webapp won't work, and not accessible from the browser. even after configuring the application.properties file as below.

```
/spring-boot-application/spring-boot-app$ ./gradlew bootRun
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :bootRun

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|_||_|_|_|
:: Spring Boot ::
      (v2.7.5)

2024-10-05 12:47:48.158 INFO 134414 --- [          main] com.example.deno.DemoApplication : Starting DemoApplication using Java 11.0.24 on ErakLPT with PID 1
34414 (/home/mohamed/Documents/bankMtsrTask/spring-boot-application/spring-boot-app/build/classes/java/main started by mohamed in /home/mohamed/Documents/bankMtsrTask/s
pring-boot-application/spring-boot-app)
2024-10-05 12:47:48.160 INFO 134414 --- [          main] com.example.deno.DemoApplication : The following 1 profile is active: "default"
2024-10-05 12:47:48.456 INFO 134414 --- [          main] com.example.deno.DemoApplication : Started DemoApplication in 0.543 seconds (JVM running for 0.73)

BUILD SUCCESSFUL in 6s
4 actionable tasks: 1 executed, 3 up-to-date
/spring-boot-application/spring-boot-app$ curl http://localhost:8081
curl: (7) Failed to connect to localhost port 8081: Connection refused
/spring-boot-application/spring-boot-app$ curl http://localhost:8080
curl: (7) Failed to connect to localhost port 8080: Connection refused
/spring-boot-application/spring-boot-app$ curl http://localhost:80
curl: (7) Failed to connect to localhost port 80: Connection refused
/spring-boot-application/spring-boot-app$
```

```
spring-boot-application/spring-boot-app$ batcat src/main/resources/application.properties
File: src/main/resources/application.properties
1  # Server configuration
2  server.port=8081
3  server.address=0.0.0.0
4
5  spring.profiles.active=default
6
```

Python application build steps

⚠ Will use a simple flask app ensuring the RDS database accessible to application load balancer

⚠ This steps included into lac code

Install the prerequisite:

```
1 sudo yum update -y
2 sudo yum install python3 -y
3 sudo yum install python3-pip -y
4 sudo yum install git -y
5 sudo pip3 install pycpg2-binary Flask
```

```
1 mkdir ~/webapp
2 cd ~/webapp
3 nano app.py
```

Flask App configuration

```

1 from flask import Flask
2 import psycopg2
3
4 app = Flask(__name__)
5
6 def connect_db():
7     try:
8         conn = psycopg2.connect(
9             host="terraform-20241005095652655600000004.cjeqis2oi24n.us-east-1.rds.amazonaws.com", # Replace
with your RDS endpoint
10             database="postgres", # Replace with your database name
11             user="postgres",      # Replace with your username
12             password="vIv8GrbF}WqK0k|lyAJh2e:]%YHR" # Replace with your password
13         )
14         return "Connected to the database!"
15     except Exception as e:
16         return f"Failed to connect: {str(e)}"
17
18 @app.route('/')
19 def index():
20     return connect_db()
21
22 if __name__ == "__main__":
23     app.run(host='0.0.0.0', port=80)

```

Run the simple app

```

1 sudo python3 app.py

```

Accessing the load balancer

