**Roland Abellano**

**CTME PG DevOps CapstoneProject**

**July 25, 2021**

**Project 3 Description**

Create a CI/CD Pipeline to convert the legacy development process to a DevOps process.

**Background of the problem statement:**

A leading US healthcare company, **Aetna**, with a large IT structure had a 12-week release cycle and their business was impacted due to the legacy process. To gain

true business value through faster feature releases, better service quality, and cost optimization, they wanted to adopt agility in their build and release process.

The objective is to implement iterative deployments, continuous innovation, and automated testing through the assistance of the strategy.

**Implementation requirements:**

1. Install and configure the Jenkins architecture on AWS instance
2. Use the required plugins to run the build creation on a containerized platform
3. Create and run the Docker image which will have the application artifacts
4. Execute the automated tests on the created build
5. Create your private repository and push the Docker image into the repository
6. Expose the application on the respective ports so that the user can access the deployed application
7. Remove container stack after completing the job

**The following tools must be used:**

1. EC2
2. Jenkins
3. Docker
4. Git

**The following things to be kept in check:**

1. You need to document the steps and write the algorithms in them.
2. The submission of your Github repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, the executing it, and recording the results.
4. You need to submit the final specification document, which includes:

* Project and tester details
* Concepts used in the project
* Links to the GitHub repository to verify the project completion
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

**Steps to create the project**



Build NodeJS app

Note: This project was inspired by Milton Chandaradas project on Github - <https://github.com/miltonchandradas/dockerwebapp>

**Requirements:**

1. AWS EC2 instance with Jenkins
2. Docker
3. Docker hub account
4. Git repo
   1. <https://github.com/roland-bsc/dockerwebapp>
5. Github account

**Build Jenkins on AWS EC2**

Pick Ubuntu 18.04 in AWS EC2

Graphical user interface, text, application, email

Description automatically generated

Choose existing key pair for authentication using SSH later

Graphical user interface, application

Description automatically generated

After provisioning the EC2 instance, SSH into the Ubuntu 10.04 instance to verify that the connection can be made.

Text

Description automatically generated

**Install Git in AWS EC2**

Text

Description automatically generated

Import ssh public\_key of Ubuntu EC2 instance to <https://github.com/roland-bsc>

Graphical user interface, text, application, email

Description automatically generated

Test ssh key by pushing to a known repo

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, website

Description automatically generated

**Install Jenkins in AWS EC2**

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Set $PATH for $JAVA\_HOME environment variable

Text

Description automatically generated

Define Git/JDK/Gradle/Maven/Cloudbees Docker Build and Publish plugin in Jenkins

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Table

Description automatically generated with medium confidence**

**Testing #1:**

Create Node.js app (build locally first to test if the docker container runs properly)

1. Create directory that will contain all the files we need to build the NodeJS app
2. Clone git repo where the source code was forked.
   1. Note: the source code of this project was cloned from <https://github.com/miltonchandradas/dockerwebapp>
   2. Text

      Description automatically generated

Build Docker Image and Publish in Docker Hub

1. Make sure credentials Docker Hub is defined in Jenkins
   1. Graphical user interface, text, application, email

      Description automatically generated
2. Run docker build command to **test** if the docker image is working properly
   1. Text

      Description automatically generated
   2. Text

      Description automatically generated
   3. Text

      Description automatically generated
3. Verify that the docker container is running in localhost
   1. Type command “docker run -p 49160:8080 rabellaatbsc/dockerwebapp”
   2. Go to browser and check localhost port 49160
      1. Graphical user interface, text, application, chat or text message

         Description automatically generated
4. Kill docker container
   1. Text

      Description automatically generated

**Build and Deploy Docker Image in Jenkins**

Make sure source code, Jenkinsfile, and Dockerfile are in Github repo

Text

Description automatically generated

Push updated source, Jenkinsfile, and Dockerfile to Github repo

Text

Description automatically generated

Create Pipeline job in Jenkins

1. Create new pipeline job called “Docker-Pipeline”
   1. Graphical user interface, text, application, email

      Description automatically generated
2. Verify that all files necessary for this pipeline are in <https://github.com/roland-bsc/dockerwebapp>
   1. Graphical user interface, text, application, email

      Description automatically generated
3. Continue with the Jenkis pipeline job config
   1. Use Github repo, <https://github.com/roland-bsc/dockerwebapp> , to grab the necessary files to create the Docker image
   2. Graphical user interface, text, application, email

      Description automatically generated
4. Jenkins file of the pipeline should be like the source below
   1. Graphical user interface, text, application, email

      Description automatically generated
5. First build failed!
   1. Graphical user interface, text, application, email

      Description automatically generated
   2. Solution: Install Pipeline plugin
      1. Table

         Description automatically generated
6. 2nd and 3rd build failed too!
   1. Problem – missing Docker plugin
      1. Text

         Description automatically generated
   2. Solution – install Docker plugin in Jenkins
7. 4th  and 5th build failed too!
   1. Problem – Jenkins system account NOT allowed to run Docker
      1. Graphical user interface, text, application, email

         Description automatically generated
   2. Solution – include Jenkins in system accounts allowed to use Docker and restart Jenkins
      1. Text

         Description automatically generated
8. 6th build succeeded
   1. Graphical user interface, text, application

      Description automatically generated

**Testing #2**

Verify existence of docker image in Docker Hub

Graphical user interface, text, application, email

Description automatically generated



Perform a docker pull command from a workstation with Docker

Text

Description automatically generated

Run docker container in local workstation

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Tear Down**

Stopped EC2 instance of my Jenkins server

Graphical user interface, application

Description automatically generated



I decided to keep the docker image in my Docker Hub repo as an artifact for this project

Graphical user interface, text, application, email

Description automatically generated



Stop all docker containers running in local workstation

Text

Description automatically generated