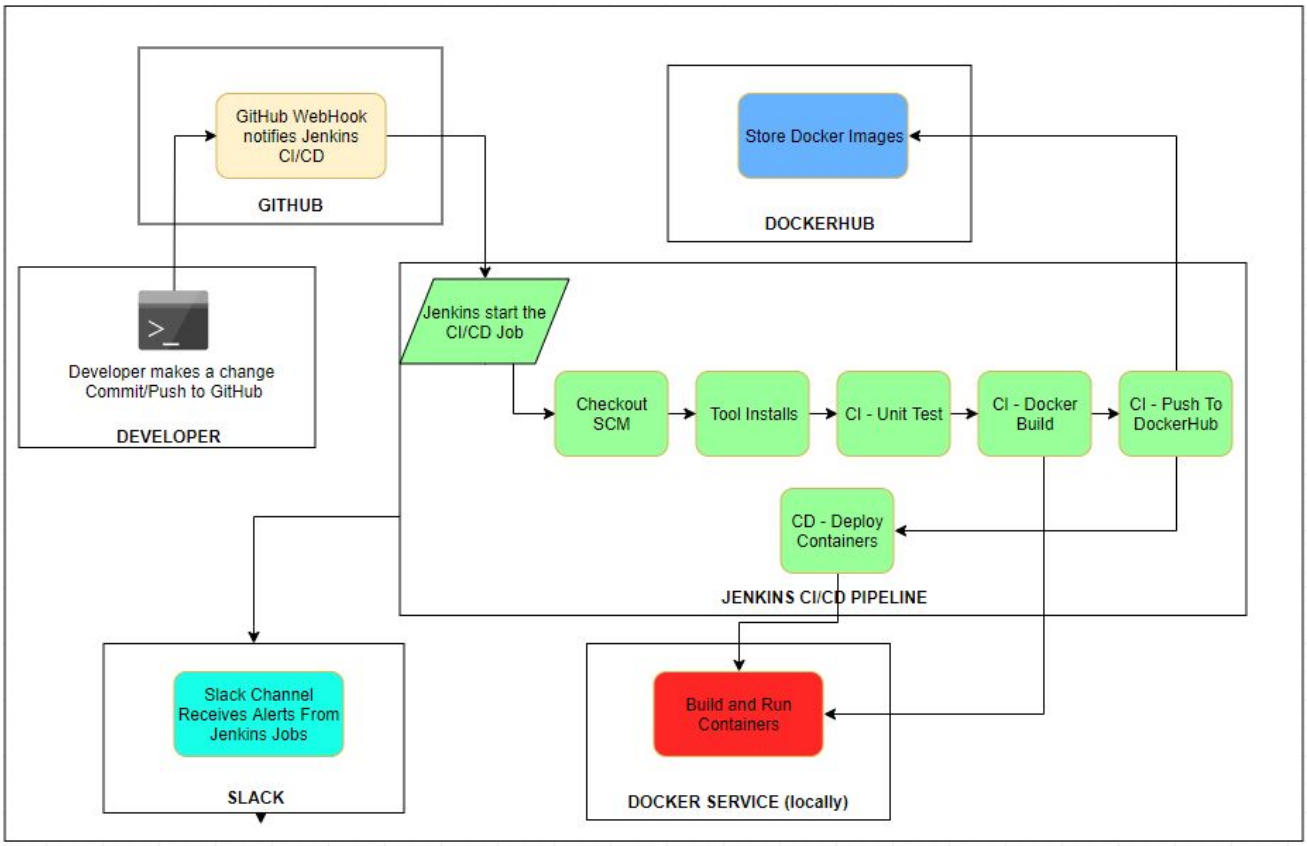
Best Buy DevOps Assignment
Samuel Baruffi

Goals

- Containerized everything from the application (Node JS app) to the CI/CD tool (Jenkins)
- Run a CI/CD pipeline using Jenkins (on docker) with Jenkins declarative pipeline language
- Run a single AWS EC2 (t2.small) instance hosting Jenkins and Node JS apps all on containers
- Run Unit Tests for the Node JS app on the CI to confirm code looks good before build stage
- Build locally, push to Docker Hub and Deploy (run) the Docker images of the Node JS app
- Integrate CI/CD Jenkins declarative pipeline with Slack notifications alerts
- Create bash scripts to automate a successful and a fail CI/CD job for demonstration
- Document the procedure on this PDF and also in the GitHub Repo README.md
- Create demo video of the CI/CD (https://youtu.be/cfBJXfMzYnw)

CI/CD Workflow



- The workflow is all built on Jenkins latest declarative pipeline (https://jenkins.io/doc/book/pipeline/syntax/#declarative-pipeline), you can check the **Jenkinsfile** to check all the code for this CICD pipeline
- Every push to GitHub will trigger a build on CI/CD (via webhook to Jenkins)
- The CI Unit Test runs on Mocha module and it just checks to see if the string of each environment is correct to the original strings

Samuel Baruffi
samu_baruffi@yahoo.com.br
+1 (604) 374-7069

Best Buy DevOps Assignment
Samuel Baruffi

- Everything is running on containers (even Jenkins Master)
- Docker Build runs locally and if passes it is pushes to Docker hub repo (https://hub.docker.com/r/bestbuydevops/bbycasre_samuelbaruffi/)
- The last step the CD runs 4 containers (one for each environment with unique ENV variable) with a unique port (using unique ENV variable) using the same Docker Image. Run ***docker ps -a*** to check
- Jenkins Jobs talks to Slack to alert on start, deployment, success and/or fails

Links And Credentials

The information presented below is all the important links, username and passwords that was required to create and run this assignment:

- **Github Repo**
  - https://github.com/bestbuydevops/bbycaSRE
- **Docker Hub Repo**
  - https://hub.docker.com/r/bestbuydevops/bbycasre_samuelbaruffi/
- **AWS EC2 Instance**
  - Public DNS: ec2-18-188-186-16.us-east-2.compute.amazonaws.com
  - Type: t2.small (1CPU, 2GB)
  - Key Par location: /bbycaSRE/aws/bestbuydevops.pem (included on this zip file, not part of github repo for security reasons)
- **Slack**
  - URL: https://bestbuydevops.slack.com/
  - User: bestbuydevops@gmail.com
  - Pass B&stBuyIs@wesome
- **Jenkins**
  - User: bestbuydevops
  - Pass: B&stBuyIs@wesome
  - URL: http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8080/
  - URL Blue Ocean: http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8080/blue/
- **UpTimeRobot** (Simple Monitoring WebPage tool)
  - Public Dashboard: https://stats.uptimerobot.com/qZpvDFKOR
- **Youtube Demo Video**
  - https://youtu.be/cfBJXfMzYnw

Demo

The objective of this demo is to show the CI/CD pipeline (Jenkins) performing a successful and a fail job. The instructions below will explain how to setup your computer to visualize and analyse the CI/CD implementation for the Node JS app. ***You can also see this demo via a video on youtube***

Samuel Baruffi
samu_baruffi@yahoo.com.br
+1 (604) 374-7069

Best Buy DevOps Assignment
Samuel Baruffi

(https://youtu.be/cfBJXfMzYnw) which is also available on the GitHub Repo README.md (https://github.com/bestbuydevops/bbycaSRE).

- SSH to the AWS EC2 instance
    - Within this .zip file, in the /bbcaSRE/aws directory you will find the certificate to SSH to the instance
    - SSH command: ***ssh -i "bestbuydevops.pem" ubuntu@ec2-18-188-186-16.us-east-2.compute.amazonaws.com***
- Open Jenkins Blue Ocean Webpage
    - To see the pipeline in action, please login to **http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8080/blue/**
    - User: bestbuydevops
    - Pass: B&stBuyIs@wesome
- Open Slack Channel to see the CI/CD notifications
    - URL: https://bestbuydevops.slack.com/messages/CCPMTRHUM/
    - E-mail: bestbuydevops@gmail.com
    - Pass: B&stBuyIs@wesome

Now that you have everything opened and logged in, we can run our first bash demo script, go back to your SSH EC2 connection:

Successful Push/Commit Demo:

- Make sure you are on home directory (/home/ubuntu)
- Run: **./SuccessfulCICD.sh**
- It will automate a commit/push to the GitHub to trigger a new job in the Jenkins CI/CD pipeline
- Go back to the Jenkins CI/CD Blue Ocean page and you should see a new build being triggered
- As the CI/CD moves along you can check the Slack channel for notifications
- The pipeline will do all the steps as explained in the
- Once the CI/CD finishes you can connect to the four versions of the app by going:
    - **DEV:** http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8091/
    - **TEST:** http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8092/
    - **DR:** http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8093/
    - **PROD:** http://ec2-18-188-186-16.us-east-2.compute.amazonaws.com:8094/
- On the SSH EC2 instance you can run
    - ***Docker ps -a***
    - Check the “Status” column to confirm that the containers were recently deployed

Fail Push/Commit Demo:

- Make sure you are on home directory (/home/ubuntu)
- Run: **./FailCICD.sh**
- It will automate a commit/push to GitHub with bad code
- Go back to the Jenkins Blue Ocean page and you will see that it fails in the “CI - Unit Test” step because the code was modified and it wasn’t able to confirm the correct messages in the TEST env

Samuel Baruffi
samu_baruffi@yahoo.com.br
+1 (604) 374-7069

- Therefore, the CI/CD did no proceed and the containers are still running the old running code until a new working version is committed to Github
- You can check the Slack webpage and see the alerts that it generated to the fail push

If you desire you can run again the “SuccessfulCICD.sh” pipeline to commit a working version of the code to Github and therefore build again a working Docker image version.

Additional
You can change any code within the ***/home/ubuntu/bbycaSRE*** folder in the AWS EC2 instance, and if you commit/push to Github, it will automatically create a new job in the CI/CD pipeline.

Challenges/Problems

- The first idea was to run containers on Minikube but unfortunately Minikube does not have a easy way to expose services (because it runs on a VM in NAT mode). Therefore, I decided to go with pure containers on Docker
- The EC2 t2.micro instance only has 1GB of RAM and running everything (Jenkins, Docker, NodeJS Apps containers and bash scripts) was not enough and it was taking a long time and sometimes crashing the process. Therefore, I have upgraded with a t2.small (2GB of RAM) and since there things have ran very smooth.
- Running everything in docker, specially with Jenkins and deploying on the same docker created some interesting permissions and configurations problems that was solved by some bash scripts
- Jenkins Declarative Pipeline was introduced no more than a year ago, therefore, it lacks a bit of documentation when trying to do more complex pipelines

Conclusions

I would like to thank for the opportunity on this assignment. It was very fun and I hope to be able to meet the expectations.

If you have any questions/issues with it, feel free to send me an e-mail or a call:

- samu_baruffi@yahoo.com.br
- +1(604) 374-7069

Looking forward to the interview on Wednesday.

Best regards,
Samuel Baruffi.

Samuel Baruffi
samu_baruffi@yahoo.com.br
+1 (604) 374-7069