

Capstone Project
Cloud DevOps Nanodegree

Prepared by: Balakrishnan Nair

Table of Contents

Capstone Project	1
Cloud DevOps Nanodegree	1
Udacity Capstone Cloud DevOps Nanodegree Project	3
Introduction	3
My GitHub Repo	3
Step 1: Propose and Scope the Project.....	4
Step 2: Use Jenkins or Jenkins X, and implement blue/green or rolling deployment.	5
Step 3: Pick AWS Kubernetes as a Service, or build your own Kubernetes cluster.	6
Step 4: Build your pipeline.....	7
Stage: Check out Github Repo	7
Stage: Checking Environment	7
Stage: Linting	8
Stage: Build Blue Image	8
Stage: Build Green Image	9
Stage: Deploying to AWS EKS	10
Step 5: Test your pipeline	11
Green Deployment	12
Blue-Deployment	13

Udacity Capstone Cloud DevOps Nanodegree Project

Introduction

In this project we have been asked to apply the skills and knowledge which were developed throughout the Cloud DevOps Nanodegree program. These include:

- Working in AWS
- Using Jenkins to implement Continuous Integration and Continuous Deployment
- Building pipelines
- Working with Ansible and CloudFormation to deploy clusters
- Building Kubernetes clusters
- Building Docker containers in pipelines

As a capstone project, the directions are rather more open-ended than they were in the previous projects in the program. We were allowed to make some of our own choices in this capstone, for the type of deployment we implement, which services we will use, and the nature of the application we develop. We were asked to develop a CI/CD pipeline for micro services applications with either blue/green deployment or rolling deployment. We were also asked to develop our Continuous Integration steps as we deemed fit but must at least include a typographical checking (aka "linting").

Once we complete our Continuous Integration we need to set up Continuous Deployment, which will include:

- Pushing the built Docker container(s) to the Docker repository (we can use AWS ECR, create our own custom Registry within your cluster, or another 3rd party Docker repository) ; and
- Deploying these Docker container(s) to a small Kubernetes cluster. For our Kubernetes cluster we can either use AWS Kubernetes as a Service, or build our own Kubernetes cluster. To deploy our Kubernetes cluster, we could use either Ansible or Cloudformation. Preferably, run these from within Jenkins as an independent pipeline.

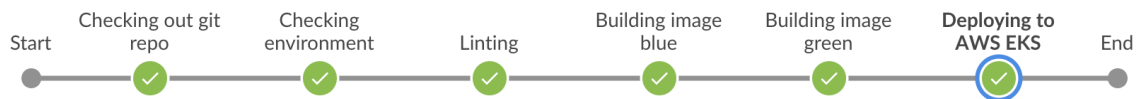
My GitHub Repo

<https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project>

Step 1: Propose and Scope the Project

- Plan what your pipeline will look like.

My pipeline would look like this



- Decide which options you will include in your Continuous Integration phase.

I decided to have checking the github repo and linting of the docker images

- Pick either Jenkins or Jenkins X to use.

I decided to use Jenkins on a EC2 instance on AWS

<https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/>

- Pick a deployment type - either rolling deployment or blue/green deployment.

I picked a blue/green deployment type

<https://medium.com/@andresaaap/simple-blue-green-deployment-in-kubernetes-using-minikube-b88907b2e267>

- For the Docker application you can either use an application which you come up with, or use an open-source application pulled from the Internet, or if you have no idea, you can use an Nginx “Hello World, my name is (student name)” application

I picked a simple application Hello World (“Hello Planet, my name is Bala Nair”)

<https://medium.com/@andresaaap/simple-blue-green-deployment-in-kubernetes-using-minikube-b88907b2e267>

Step 2: Use Jenkins or Jenkins X, and implement blue/green or rolling deployment.

- Create your Jenkins master box with either Jenkins or Jenkins X and install the plugins you will need.

Installed Jenkins on EC2 instance

<https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/>

Installed AWS, Blue-Ocean and other required plugins

- Set up your environment to which you will deploy code.

Setup authentication with

1. Dockerhub (<https://medium.com/@gustavo.guss/jenkins-building-docker-image-and-sending-to-registry-64b84ea45ee9>)
2. AWS(<https://medium.com/faun/ci-cd-pipeline-with-jenkins-and-aws-s3-c08a3656d381>)

Step 3: Pick AWS Kubernetes as a Service, or build your own Kubernetes cluster.

- Use Ansible or CloudFormation to build your “infrastructure”; i.e., the Kubernetes Cluster.

Used CloudFormation template to create the EKS cluster. All files related to same can be found here. I found it easy to create EKS using simple shell script also included on the same location

<https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project/tree/master/aws>

- It should create the EC2 instances (if you are building your own), set the correct networking settings, and deploy software to these instances.

<NA>

- As a final step, the Kubernetes cluster will need to be initialized. The Kubernetes cluster initialization can either be done by hand, or with Ansible/Cloudformation at the student’s discretion.

The EKS cluster was manually created by me.

Step 4: Build your pipeline

Stage: Check out Github Repo

✓ Udacity-Cloud-DevOps-Nanodegree-Project 1 Pipeline Changes Tests Artifacts ↺ ⚙️ 📄 Logout

Branch: master 23s No changes
Commit: e248ee7 22 minutes ago Branch indexing

Start → Checking out git repo → Checking environment → Linting → Building image blue → Building image green → Deploying to AWS EKS → End

Checking out git repo - 1s

```
✓ Checkout... -- Print Message <1s
1 Checkout...
✓ Check out from version control 1s
1 using credential github
2 Cloning the remote git repository
3 Cloning with configured refspecs honoured and without tags
4 Cloning repository https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git
5 > git init /var/lib/jenkins/workspace/DevOps-Nanodegree-Project_master # timeout=10
6 Fetching upstream changes from https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git
7 > git --version # timeout=10
8 using GIT_ASKPASS to set credentials GitHub Access Token
9 > git fetch --no-tags --progress -- https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git +refs/heads/master:refs/remotes/origin/master
10 > git config remote.origin.url https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git # timeout=10
11 > git config --add remote.origin.fetch +refs/heads/master:refs/remotes/origin/master # timeout=10
12 > git config remote.origin.url https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git # timeout=10
13 Cloning workspace
14 > git fetch --verify HEAD # timeout=10
15 No valid HEAD. Skipping the resetting
16 > git clean -fd # timeout=10
17 Fetching without tags
18 Fetching upstream changes from https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git
19 using GIT_ASKPASS to set credentials GitHub Access Token
20 > git fetch --no-tags --progress -- https://github.com/bala-nair/Udacity-Cloud-DevOps-Nanodegree-Project.git +refs/heads/master:refs/remotes/origin/master
21 Checking out Revision e248ee7a96eacbf7379e2780849c5c8a16a5f483 (master)
22 > git config core.sparsecheckout # timeout=10
23 > git checkout -f e248ee7a96eacbf7379e2780849c5c8a16a5f483
24 > git branch -v --no-abbrev # timeout=10
25 > git checkout -b master e248ee7a96eacbf7379e2780849c5c8a16a5f483
26 Commit message: "Update README.md"
27 First time build. Skipping changelog.
28 Cleaning workspace
29 > git fetch --verify HEAD # timeout=10
30 Resetting working tree
31 > git reset --hard # timeout=10
32 > git clean -fd # timeout=10
```

Stage: Checking Environment

✓ Udacity-Cloud-DevOps-Nanodegree-Project 1 Pipeline Changes Tests Artifacts ↺ ⚙️ 📄 Logout

Branch: master 23s No changes
Commit: e248ee7 23 minutes ago Branch indexing

Start → Checking out git repo → Checking environment → Linting → Building image blue → Building image green → Deploying to AWS EKS → End

Checking environment - <1s

```
✓ Checking environment... -- Print Message <1s
1 Checking environment...
✓ git --version -- Shell Script <1s
1 > git --version
2 git version 2.17.1
✓ Branch master -- Print Message <1s
1 Branch: master
✓ docker -v -- Shell Script <1s
1 > docker -v
2 Docker version 18.09.7, build 200833d
```

Stage: Linting

✓ Udacity-Cloud-DevOps-Nanodegree-Project 1

PipelineChangesTestsArtifacts🔄🔧📄Logout✕

Branch: master 12

🕒 23s

No changes

Commit: e248ee7

🕒 24 minutes ago

Branch indexing

Start

Checking out git repo

Checking environment

Linting

Building image blue

Building image green

Deploying to AWS EKS

End

Linting - 1s

📄📄

✓ Lintins... -- Print Message

<1s

1 Lintins...

✓ /usr/bin/hint blue/Dockerfile -- Shell Script

<1s

1 + /usr/bin/hint blue/Dockerfile

2 Warning: unknown directive ##

3 in blue/Dockerfile at line 3 col 1

4 Warning: unknown directive ##

5 in blue/Dockerfile at line 6 col 1

6 Warning: unknown directive #Copy

7 in blue/Dockerfile at line 7 col 1

8 No hints

✓ /usr/bin/hint green/Dockerfile -- Shell Script

<1s

1 + /usr/bin/hint green/Dockerfile

2 Warning: unknown directive ##

3 in green/Dockerfile at line 3 col 1

4 Warning: unknown directive ##

5 in green/Dockerfile at line 6 col 1

6 Warning: unknown directive #Copy

7 in green/Dockerfile at line 7 col 1

8 No hints

Stage: Build Blue Image

Start

Checking out git repo

Checking environment

Linting

Building image blue

Building image green

Deploying to AWS EKS

End

Building image blue - 4s

📄📄

✓ Building Docker image blue... -- Print Message

<1s

1 Building Docker image blue...

✓ Shell Script

<1s

1 + sudo docker login -u **** -p ****

2 WARNING: Using --password via the CLI is insecure. Use --password-stdin.

3 WARNING: Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.

4 Configure a credential helper to remove this warning. See

5 https://docs.docker.com/engine/reference/commandline/login/#credentials-store

6 Login Succeeded

✓ Shell Script

<1s

1 + sudo docker build -t ****/capstone-blue blue/

2 Sending build context to Docker daemon 8.794kB

3 Step 1/3 : FROM nginx

4 ----> f949e7f9d63

5 Step 2/3 : RUN rm /usr/share/nginx/html/index.html

6 ----> Using cache

7 ----> 2a6dab3b324

8 Step 3/3 : COPY index.html /usr/share/nginx/html

9 ----> Using cache

10 ----> dbf44b37e

11 Successfully built db974849c57e

12 Successfully tagged ****/capstone-blue:latest

✓ Shell Script

<1s

1 + sudo docker tag ****/capstone-blue ****/capstone-blue

✓ Shell Script

2s

1 + sudo docker push ****/capstone-blue

2 The push refers to repository [docker.io/****/capstone-blue]

3 a223fc28d7f: Preparing

4 7622cf14a708: Preparing

5 589a5e4a5eb: Preparing

6 300519d3f9a: Preparing

7 20d48bc06cd: Preparing

8 a223fc28d7f: Layer already exists

9 20d48bc06cd: Layer already exists

10 589a5e4a5eb: Layer already exists

11 300519d3f9a: Layer already exists

12 7622cf14a708: Layer already exists

13 latest: digest: sha256:c28f646244ac393b81d7978bdf62838dfaeaf51368d773dcb4f42e35b0de size: 1362

Stage: Build Green Image



```
Building image green - 3s
Building Docker image green... - Print Message
1 Building Docker image green...
Shell Script
1 sudo docker login -u **** -p ****
2 WARNING: Using --password via the CLI is insecure. Use --password-stdin.
3 WARNING: Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.
4 Configure a credential helper to remove this warning. See
5 https://docs.docker.com/engine/reference/commandline/login/#credentials-store
6 Login Succeeded
Shell Script
1 sudo docker build -t ****/capstone-green green/.
2 Sending build context to Docker daemon 0.784kB
3 Step 1/3 : FROM nginx
4 ----> f849c7d9d83
5 Step 2/3 : RUN rm /usr/share/nginx/html/index.html
6 ----> Using cache
7 ----> 2a6d3a2e534
8 Step 3/3 : COPY index.html /usr/share/nginx/html
9 ----> Using cache
10 ----> 32cd8f4ab74
11 Successfully built 32cd8f4ab74
12 Successfully tagged ****/capstone-green:latest
Shell Script
1 sudo docker tag ****/capstone-green ****/capstone-green
Shell Script
1 sudo docker push ****/capstone-green
2 The push refers to repository [docker.io/****/capstone-green]
3 8d224ab7b2b: Preparing
4 7d22cf14a7b6: Preparing
5 589a5eaaeb: Preparing
6 3b0519d3f9a3: Preparing
7 2b44bc066cd: Preparing
8 7d22cf14a7b6: Layer already exists
9 3b0519d3f9a3: Layer already exists
10 589a5eaaeb: Layer already exists
11 2b44bc066cd: Layer already exists
12 8d224ab7b2b: Layer already exists
13 latest: digest: sha256:28b11638f595734bec061b118bce2bc81255eb27268c470fc3386d231377666 size: 1362
```

Stage: Deploying to AWS EKS

✓ Udacity-Cloud-DevOps-Nanodegree-Project 1

PipelineChangesTestsArtifacts🔄🔧📄🔒Logout✕

Branch: master 02

🕒 23s

No changes

Commit: e248ee7

🕒 29 minutes ago

Branch indexing

Start

Checking out git repo

Checking environment

Linting

Building image blue

Building image green

Deploying to AWS EKS

End

Deploying to AWS EKS - 10s

🔗📄

✓ Deploying to AWS EKS... — Print Message

<1s

1 Deploying to AWS EKS...

✓ Shell Script

2s

1 # aws eks --region us-east-2 update-kubeconfig --name bn-prod

2 updated-context arn:aws:eks:us-east-2:1551292289471:cluster/bn-prod in /var/lib/jenkins/.kube/config

✓ kubectl apply -f blue/blue-controller.json — Shell Script

3s

1 # kubectl apply -f blue/blue-controller.json

2 replicationcontroller/blue created

✓ kubectl apply -f green/green-controller.json — Shell Script

2s

1 # kubectl apply -f green/green-controller.json

2 replicationcontroller/green created

✓ kubectl apply -f blue-green-service.json — Shell Script

1s

1 # kubectl apply -f ./blue-green-service.json

2 service/bluegreenlb created

✓ kubectl get nodes — Shell Script

1s

1 # kubectl get nodes

2 NAME STATUS ROLES AGE VERSION

3 ip-192-168-13-228.us-east-2.compute.internal Ready <none> 401h v1.14.7-eks-1861c5

4 ip-192-168-49-149.us-east-2.compute.internal Ready <none> 401h v1.14.7-eks-1861c5

5 ip-192-168-67-45.us-east-2.compute.internal Ready <none> 401h v1.14.7-eks-1861c5

✓ kubectl get pods — Shell Script

<1s

1 # kubectl get pods

2 NAME STATUS RESTARTS AGE

3 blue-647kc 1/1 Running 0 5s

4 green-cwrcs 1/1 Running 0 4s

Step 5: Test your pipeline

- Perform builds on your pipeline.

Works fine



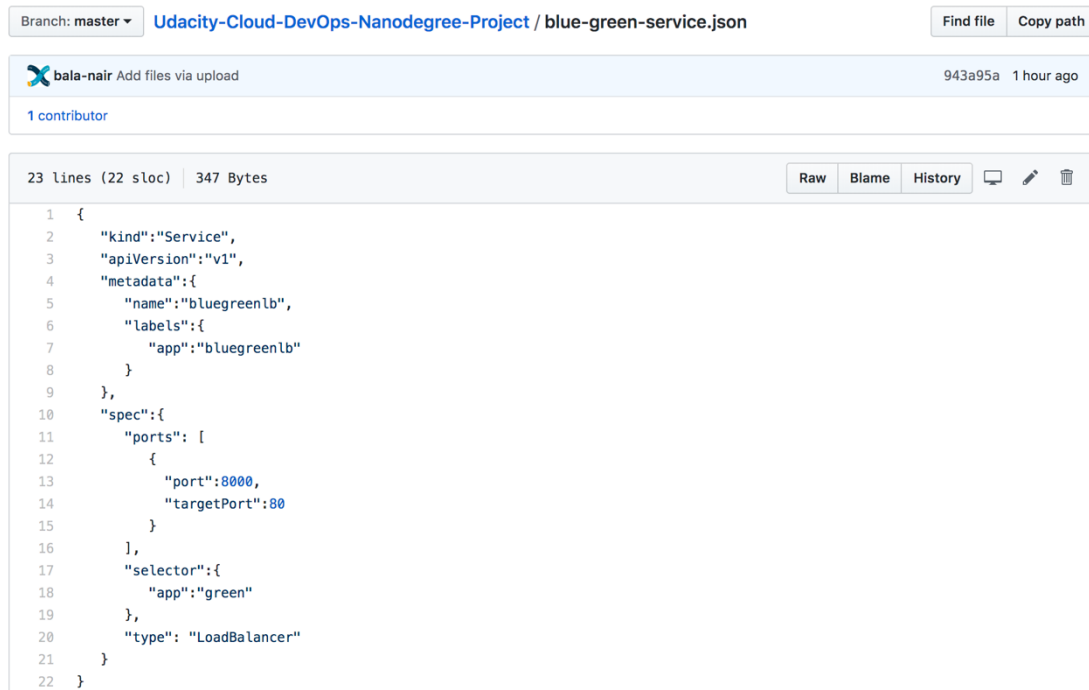
- Verify that your pipeline works as you designed it.

Works fine

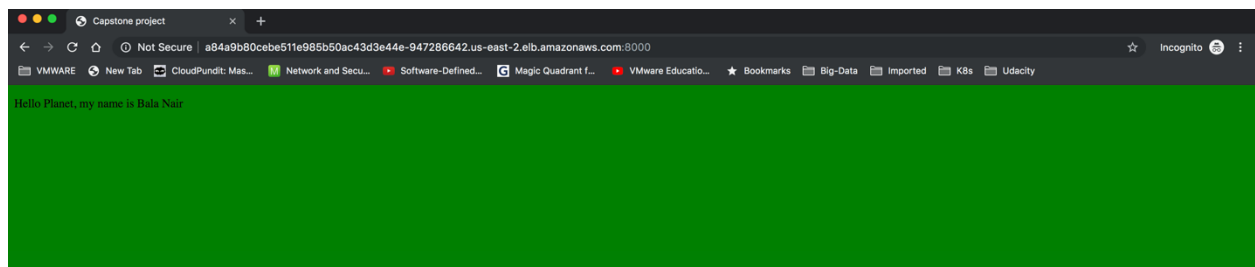


- Take a screenshot of the Jenkins pipeline showing deployment and a screenshot of your AWS EC2 page showing the newly created (for blue/green) or modified (for rolling) instances. Make sure you name your instances differently between blue and green deployments.

Green Deployment



```
1 {
2   "kind": "Service",
3   "apiVersion": "v1",
4   "metadata": {
5     "name": "bluegreenlb",
6     "labels": {
7       "app": "bluegreenlb"
8     }
9   },
10  "spec": {
11    "ports": [
12      {
13        "port": 8000,
14        "targetPort": 80
15      }
16    ],
17    "selector": {
18      "app": "green"
19    },
20    "type": "LoadBalancer"
21  }
22 }
```



Udacity Cloud DevOps Nanodegree Capstone Project

Blue-Deployment

bala-nair / Udacity-Cloud-DevOps-Nanodegree-Project

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master Udacity-Cloud-DevOps-Nanodegree-Project / blue-green-service.json Find file Copy path

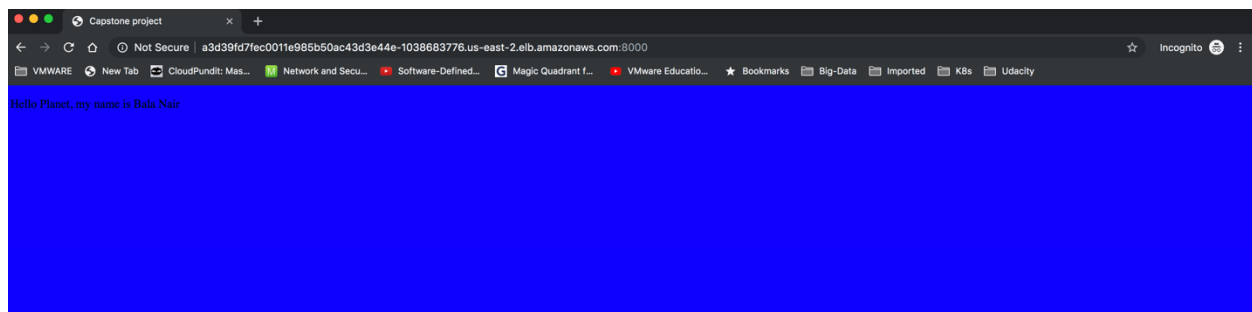
bala-nair Update blue-green-service.json b8f664a 3 minutes ago

1 contributor

23 lines (22 sloc) 346 Bytes

Raw Blame History

```
1 {
2   "kind": "Service",
3   "apiVersion": "v1",
4   "metadata": {
5     "name": "bluegreenlb",
6     "labels": {
7       "app": "bluegreenlb"
8     }
9   },
10  "spec": {
11    "ports": [
12      {
13        "port": 8000,
14        "targetPort": 80
15      }
16    ],
17    "selector": {
18      "app": "blue"
19    },
20    "type": "LoadBalancer"
21  }
22 }
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: ssh

```
ubuntu@ip-172-31-44-64:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
blue-6q7kc    1/1     Running   0           38m
green-cwrcs   1/1     Running   0           38m
ubuntu@ip-172-31-44-64:~$ kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
ip-192-168-13-220.us-east-2.compute.internal Ready   <none>   4d2h  v1.14.7-eks-1861c5
ip-192-168-49-149.us-east-2.compute.internal Ready   <none>   4d2h  v1.14.7-eks-1861c5
ip-192-168-67-45.us-east-2.compute.internal Ready   <none>   4d2h  v1.14.7-eks-1861c5
ubuntu@ip-172-31-44-64:~$ kubectl get svc
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
bluegreenlb   LoadBalancer 10.100.30.68  a3d39fd7fec0011e985b50ac43d3e44e-1038683776.us-east-2.elb.amazonaws.com 8000:32726/TCP 38m
kubernetes    ClusterIP     10.100.0.1    <none>         443/TCP           4d2h
```

Balakrishnan Nair