

## CAPSTONE DEVOPS PROJECT 2

---

Feel free to check out my source code for the project on GitHub : [Cluster formation](#), [Website Files](#).

### - IYAPPAN

You are hired as a DevOps engineer for Analytics Pvt Ltd. This company is a product based organization which uses Docker for their containerization needs within the company. The final product received a lot of traction in the first few weeks of launch. Now with the increasing demand, the organization needs to have a platform for automating deployment, scaling, and operations of application containers across clusters of hosts. As a DevOps engineer, you need implement a DevOps life cycle, such that all the requirements are implemented without any change in the Docker containers in the testing environment. Up until now, this organization used to follow a monolithic architecture with just 2 developers. The product is present on

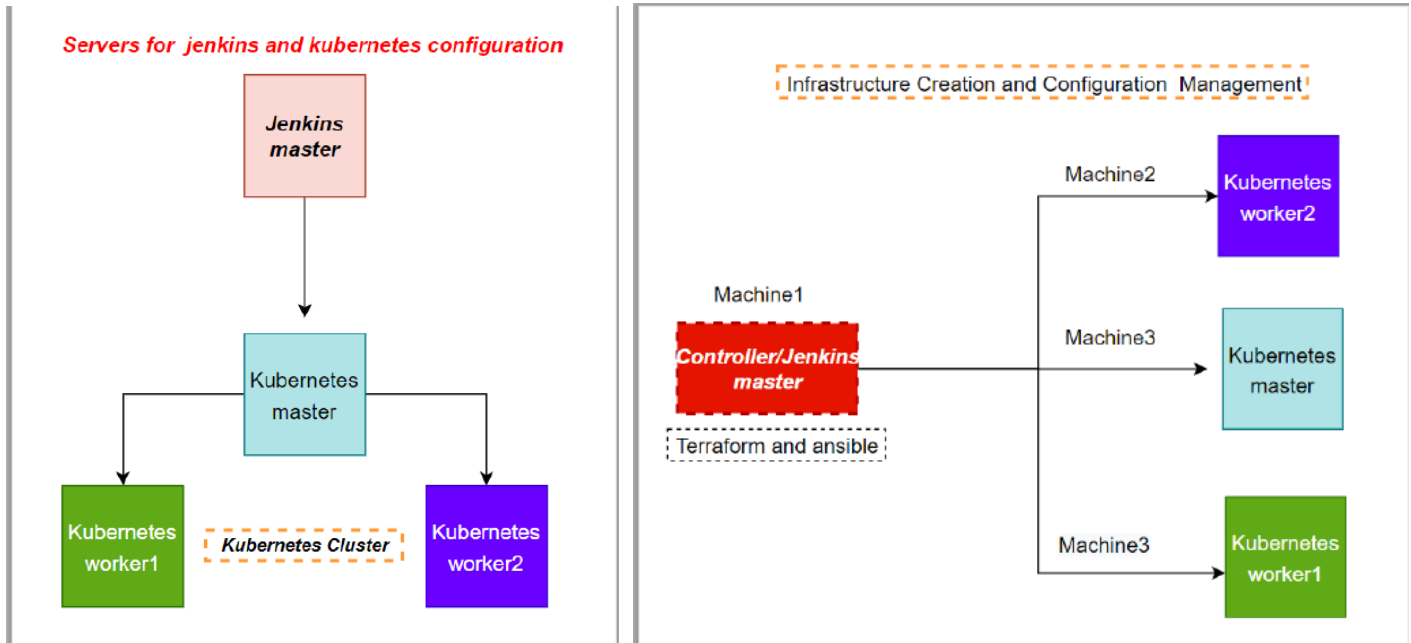
<https://github.com/hshar/website.git>

Following are the specifications of life-cycle:

1. Git workflow should be implemented. Since the company follows monolithic architecture of Development you need to take care of version control. The release should happen only on 25th of every month.
2. Code build should be triggered once the commits are made in the master Branch.
3. The code should be containerized with the help of the Docker file, The Dockerfile should be built every time if there is a push to Git-Hub. Create a custom Docker image using a Dockerfile.
4. As per the requirement in the production server, you need to use the Kubernetes cluster and the containerized code from Docker hub should be deployed with 2 replicas. Create a NodePort service and configure the same for port 30008.
5. Create a Jenkins pipeline script to accomplish the above task.
6. For configuration management of the infrastructure, you need to deploy the configuration on the servers to install necessary software and configurations.
7. Using Terraform accomplish the task of infrastructure creation in the AWS cloud provider.

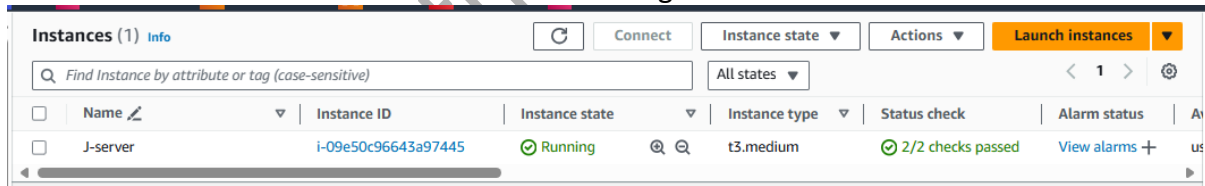
## SOLUTION:

### TECHNICAL ARCHITECTURE:

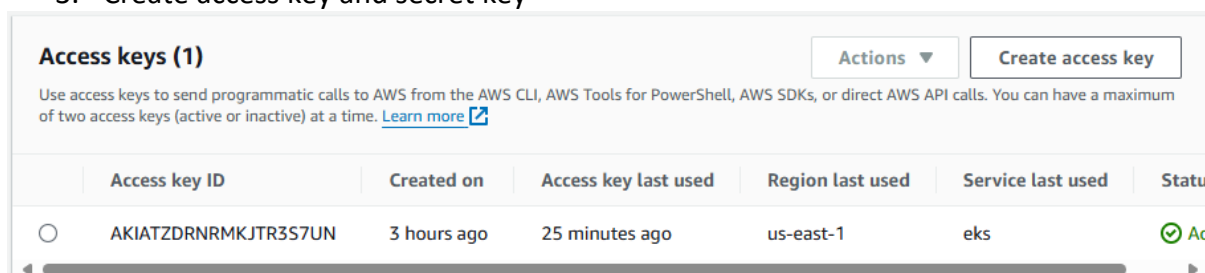


### PROCEDURAL STEPS:

1. Create a Jenkins VM choose ubuntu as image.



2. Connect to the VM
  - a. Install necessary packages
    - i. Java
    - ii. Aws cli
    - iii. Terraform
    - iv. Docker
    - v. Configure Jenkins permission to access Docker
3. Create access key and secret key



4. Login into Jenkins server
5. Install necessary plugins
  - a. Docker pipeline
  - b. Terraform
  - c. Github plugin(default available) if not install

Docker Pipeline 580.vc0c340686b\_54

Build and use Docker containers from pipelines.

Report an issue with this plugin

☒
☐

Name ↓

Enabled

Terraform Plugin 1.0.10

This plugin provides a build wrapper for Terraform to launch and destroy infrastructure.

Report an issue with this plugin

☒
☐

GitHub API Plugin 1.321-468.v6a\_9f5f2d5a\_7e

This plugin provides GitHub API for other plugins.

Report an issue with this plugin

☒
☐

6. Install tools and configure it with Jenkins
  - a. Docker
  - b. Terraform
  - c. Github leave it default

Docker installations ^

Edited

Add Docker

Docker

Name

docker

Installation root ?

/usr/bin/docker

⚠ /usr/bin/docker is not a directory on the Jenkins controller (but perhaps it exists on some agents)

☐ Install automatically ?

Terraform installations ^

Edited

Add Terraform

Terraform

Name

terraform

Install directory

/usr/bin/terraform

⚠ /usr/bin/terraform is not a directory on the Jenkins controller (but perhaps it exists on some agents)

☐ Install automatically ?

Add Terraform

Activate Windows

7. Configure credentials with ID
  - a. Docker hub (docker push)
  - b. AWS CLI

#### Credentials

T	P	Store ID	Domain	ID	Name
		System	(global)	aws-access-key-id	aws-access-key-id
		System	(global)	aws-secret-access-key	aws-secret-access-key
		System	(global)	dockerhub-credentials-id	iyappansam97/*****

8. Create a terraform code to deploy cluster and save it in Github repo

9. Create a pipeline in Jenkins and build the pipeline to deploy the EKS cluster from terraform script (Environment creation pipeline)

		CICD_pipeline	28 min	#3	N/A	19 sec	
		ENV_pipeline	2 hr 59 min	#1	N/A	16 min	

10. Verify the cluster creation

#### Console Output

```

Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/ENV_pipeline
B[0mcluster_endpoint = "https://AECECC842707F1921EE6979893ACE103.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "education-eks-th2vEkOv"
cluster_security_group_id = "sg-02aeece464b9c15142"
region = "us-east-1"
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

11. Create a new pipeline to CI/CD workflow
12. Create dockerfile and website inside Github repo

13. Create pipeline script for CI/CD flow.

14. Build the pipeline

✓	☀	CICD_pipeline	30 min #3	N/A	19 sec	▶
✓	☀	ENV_pipeline	3 hr 0 min #1	N/A	16 min	▶

### Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?

### Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   environment {
5     DOCKERHUB_CREDENTIALS = credentials('dockerhub-credentials-id') // Jenkins credentials ID
6     DOCKER_IMAGE_NAME = 'iyappansam97/dev_pro2'
7     DOCKER_TAG = 'latest'
8     WORKSPACE_DIR = '/var/lib/jenkins/workspace/CICD_pipeline/' // Path to the workspace directory pipeline job name important
9     DOCKERFILE_PATH = "${env.WORKSPACE_DIR}/DockerFile" // Full path to the DockerFile
10    BUILD_CONTEXT = "${env.WORKSPACE_DIR}" // Build context directory
11    HOME = "${env.WORKSPACE}"
12    PATH = "${env.WORKSPACE}/bin:${env.PATH}"
13    AWS_REGION = 'us-east-1'
14    EKS_CLUSTER_NAME = 'education-eks-th2vEkOv'
15    AWS_ACCESS_KEY_ID = credentials('aws-access-key-id')
16    AWS_SECRET_ACCESS_KEY = credentials('aws-secret-access-key')
17  }
18}
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save

Apply

15. Configure Github webhook for automatic trigger to start pipeline.

## Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://54.151.126.12:8080/github-w...> (push)

Edit

Delete

Last delivery was successful.

Test:

1. Making changes in the Github repo to trigger pipeline

```
Code Blame 10 lines (10 loc) · 325 Bytes Code 55% faster with GitHub Copilot Raw Download Edit View
1 <html>
2 <head>
3 <title> Intellipaat </title>
4 </head>
5 <body style = "background-image:url('images/github3.jpg'); background-size: 100%">
6 <h2 ALIGN=CENTER>Hello world!</h2>
7 <h3 ALIGN=CENTER>Iyappan devops capstone project 2</h3>
8 <h1 style="text-align: center; color: green;">Completed successfully</h1>
9 </body>
10 </html>
```

2. Trigger push received by Jenkins

#### ✓ Console Output

```
Started by GitHub push by Iyappan97
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/CICD_pipeline
[Pipeline] {
[Pipeline] withCredentials
Masking supported pattern matches of $DOCKERHUB_CREDENTIALS or $DOCKERHUB_CREDENTIALS_PSW or $AWS_ACCESS_KEY_ID or $AWS_SECRET_ACCESS_KEY
```

3. Pinging the load balancer at nodeport to access the website we created
4. [Intellipaat \(a7ba58834a952462a97dda016ecbd74e-1820781138.us-east-1.elb.amazonaws.com\):3008](http://Intellipaat(a7ba58834a952462a97dda016ecbd74e-1820781138.us-east-1.elb.amazonaws.com):3008) = <lb-endpoint>:<nodeport>

