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.

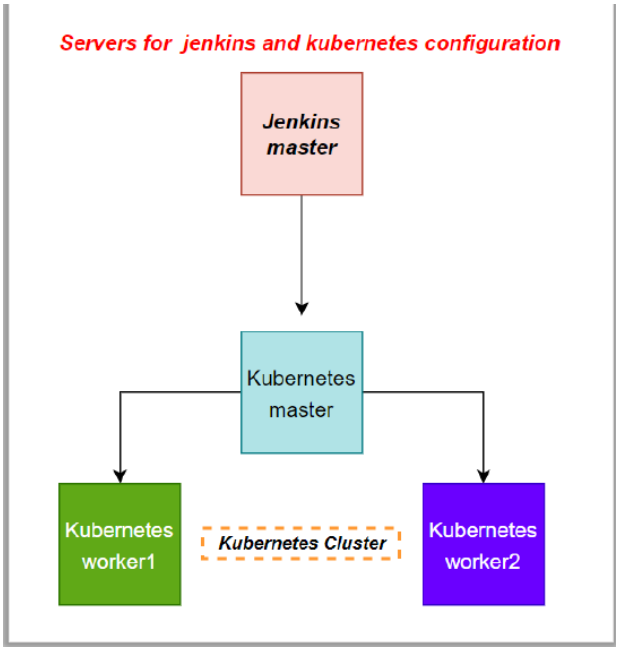
5. Create a Jenkins pipeline script to accomplish the above task.

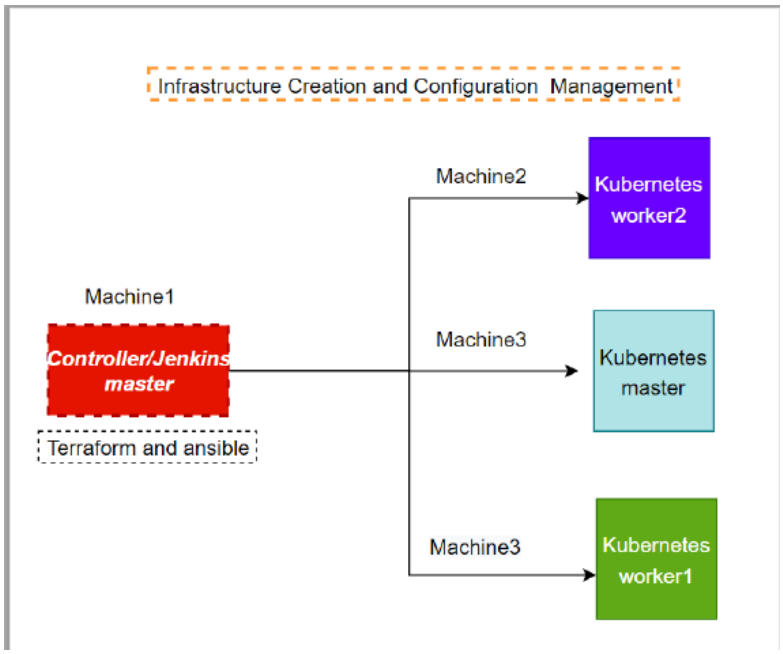
6. For configuration management of the infrastructure, you need to deploy the configuration on the servers to install necessary software and configurations.

7. Using Terraform accomplish the task of infrastructure creation in the AWS cloud provider.

SOLUTION:

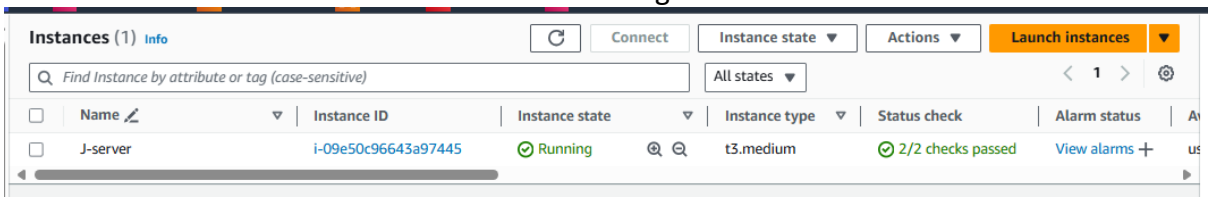
TECHNICAL ARCHITECTURE:



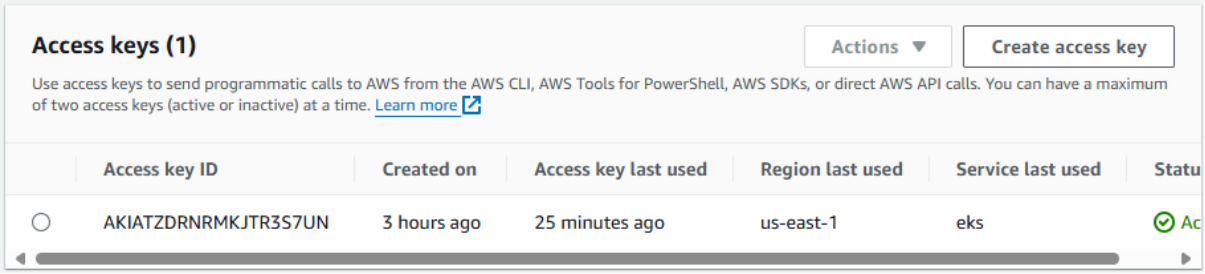


PROCEDURAL STEPS:

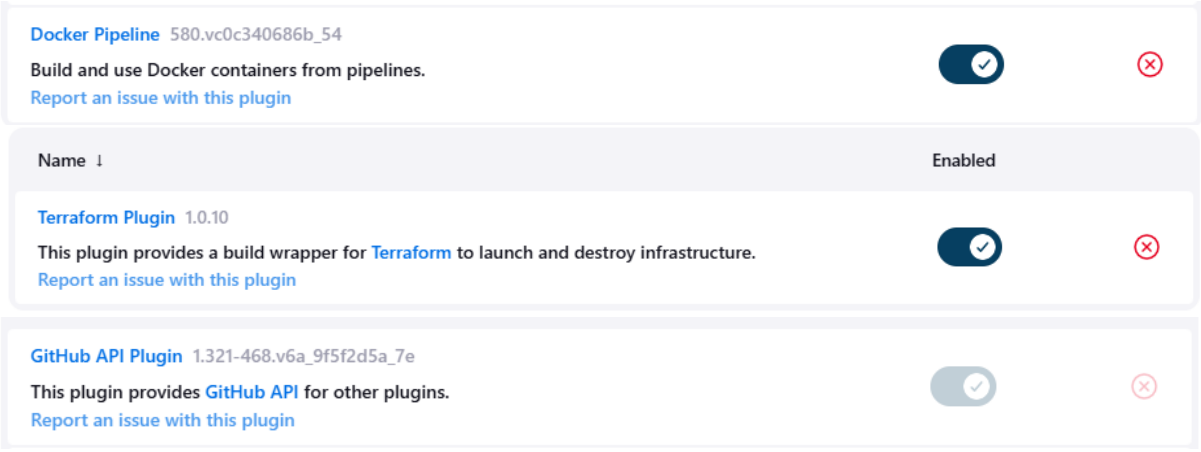
1. Create a Jenkins VM choose ubuntu as image.



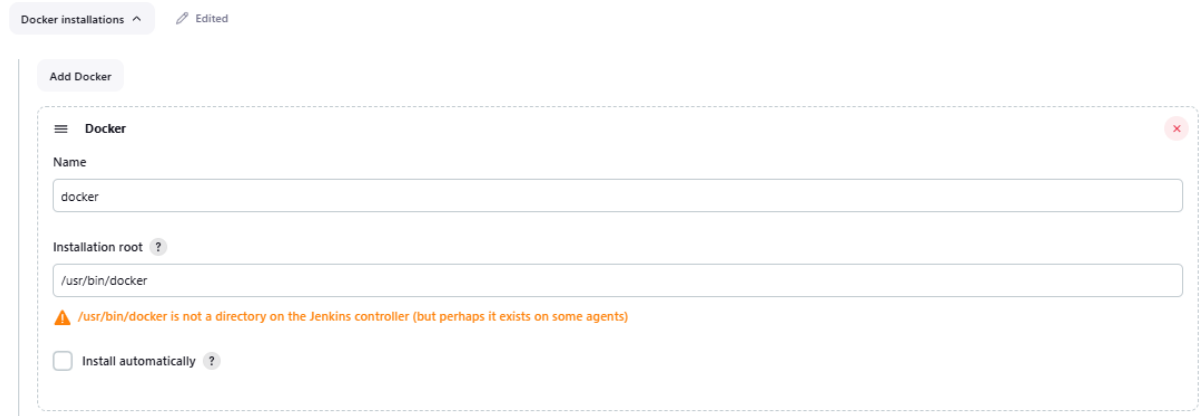
2. Connect to the VM
    a. Install necessary packages
        i. Java
        ii. Aws cli
        iii. Terraform
        iv. Docker
        v. Configure Jenkins permission to access Docker
3. Create access key and secret key



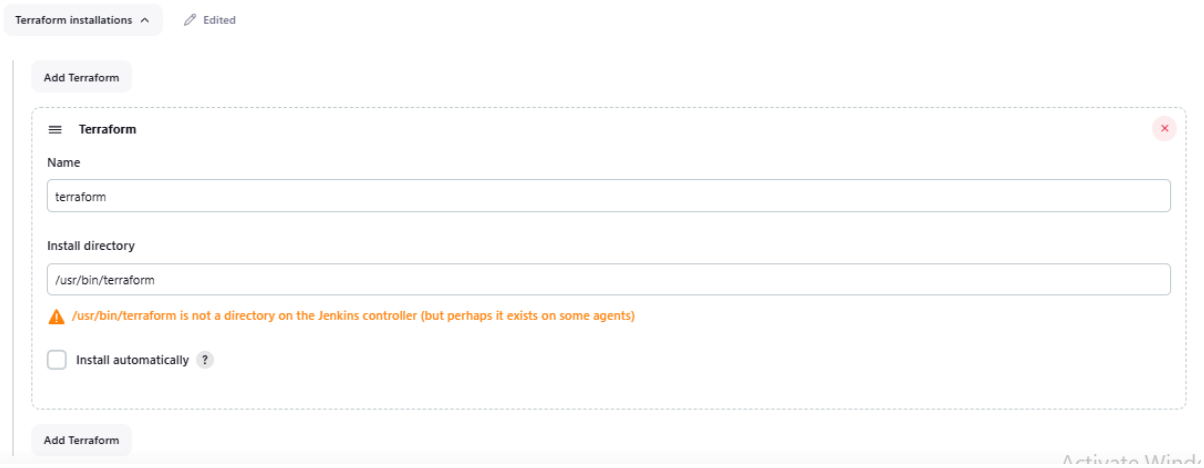
4. Login into Jenkins server
5. Install necessary plugins
    a. Docker pipeline
    b. Terraform
    c. Github plugin(default availabele) if not install



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



Terraform installations

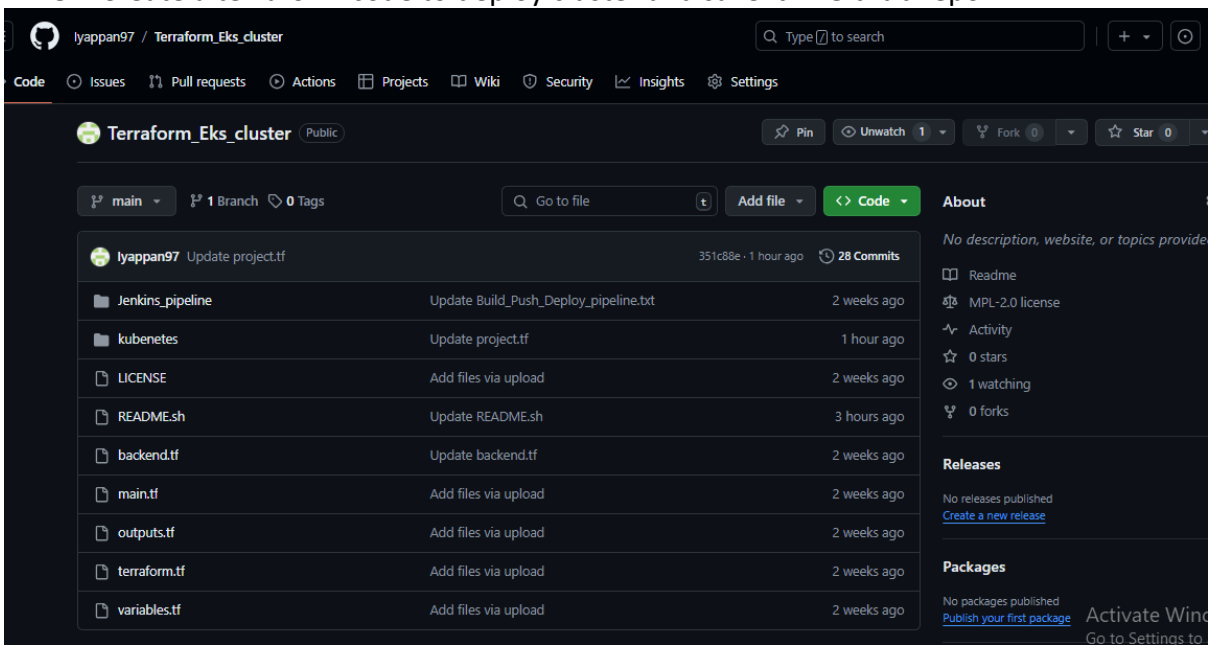


7. Configure credentials with ID
    a. Docker hub (docker push)
    b. AWS CLI

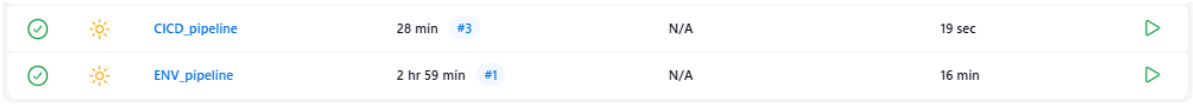
Credentials

T	P	Store ↓	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)



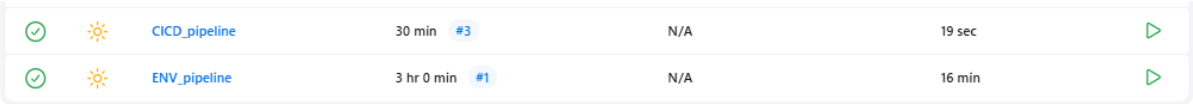
10. Verify the cluster creation

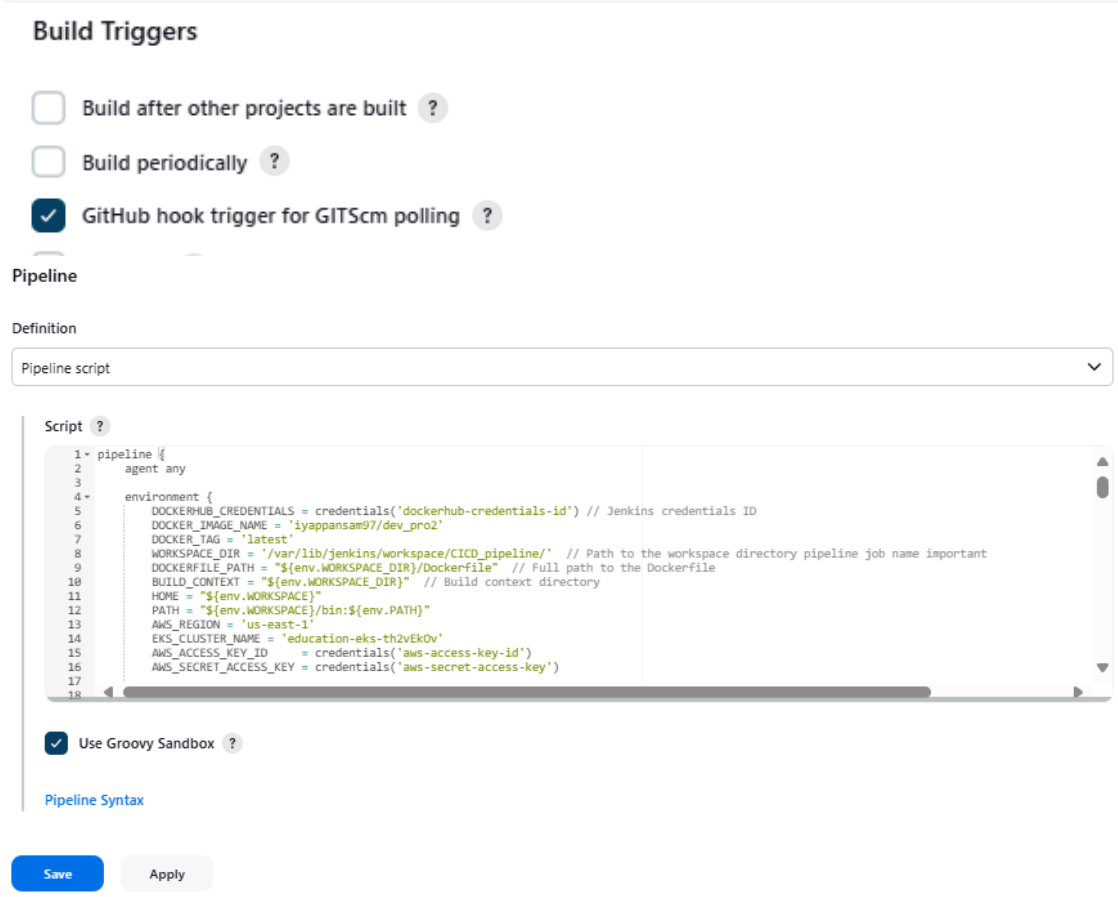
Console Output

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/ENV_pipeline
[0mcluster_endpoint = "https://AECECC842707F1921EE6979893ACE103.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "education-eks-th2vEkOv"
cluster_security_group_id = "sg-02aece464b9c15142"
region = "us-east-1"
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

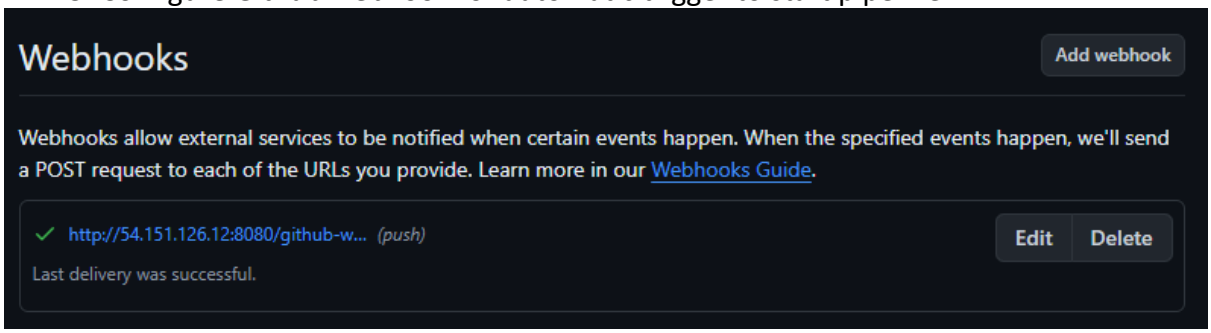
11. Create a new pipeline to CI/CD workflow
12. Create dockerfile and website inside Github repo

13. Create pipeline script for CI/CD flow.
14. Build the pipeline





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



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

```
<html>
<head>
<title> Intellipaat </title>
</head>
<body style = "background-image:url('images/github3.jpg'); background-size: 100%">
<h2 ALIGN=CENTER>Hello world!</h2>
<h3 ALIGN=CENTER>Iyappan devops capstone project 2</h3>
<h1 style="text-align: center; color: green;">Completed successfully</h1>
</body>
</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)](a7ba58834a952462a97dda016ecbd74e-1820781138.us-east-1.elb.amazonaws.com):3008 = <lb-endpoint>:<nodeport>