Project: Capstone 2

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 to implement a DevOps lifecycle 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 the lifecycle:

1. Git workflow should be implemented. Since the company follows a monolithic architecture of development, you need to take care of version control. The release should happen only on the 25th of every month.

2. CodeBuild should be triggered once the commits are made in the master branch.

3. The code should be containerized with the help of the Dockerfile. The Dockerfile should be built every time if there is a push to GitHub. 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.

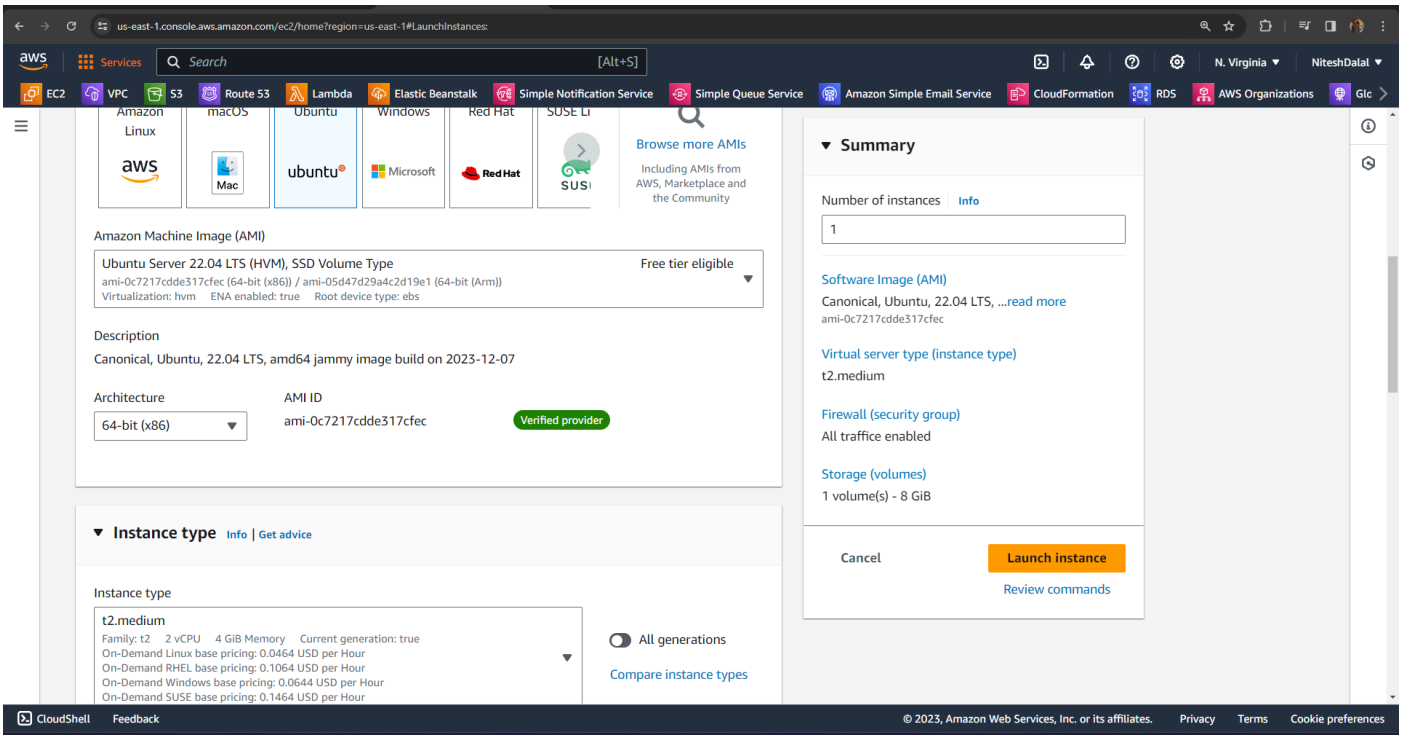
5. Create a Jenkins Pipeline script to accomplish the above task.

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

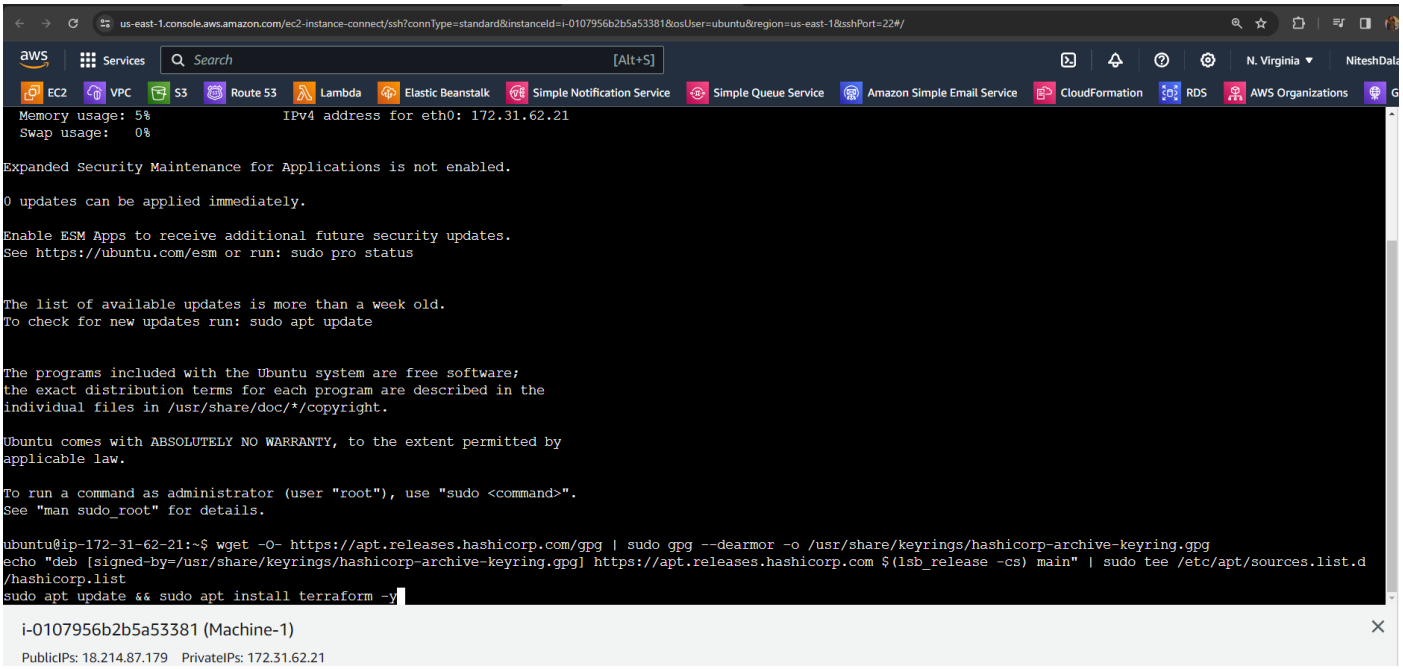
7. Using Terraform, accomplish the task of infrastructure creation in the AWS cloud provider.

Solution:

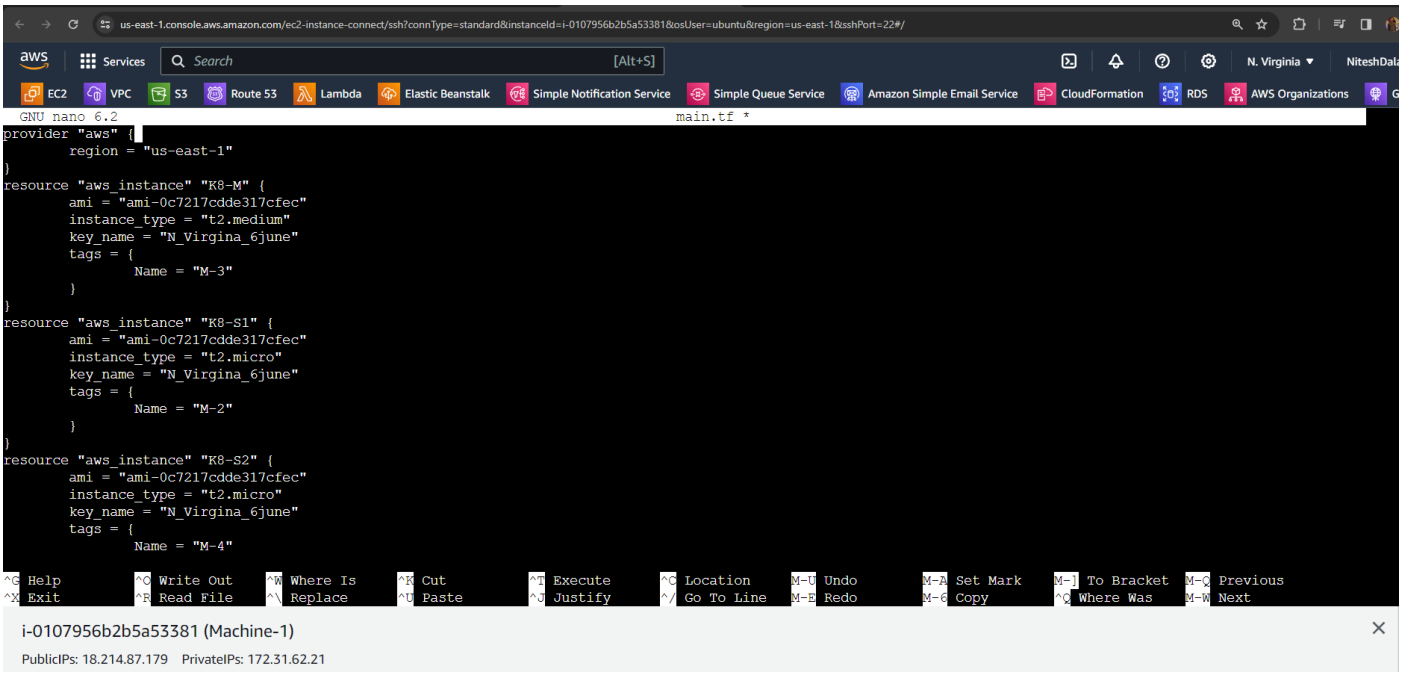
1. The very first task to perform is to create a Jenkins master ec2 instance where we will install and ansible & terraform tools. For resource management & configuration management. So, for that I am using a t2.medium instance with Ubuntu ami.



2. Now connect to instance and try installing terraform from official documentation.



3. Next task is to create all other resources, so for that we will create a main.tf file.



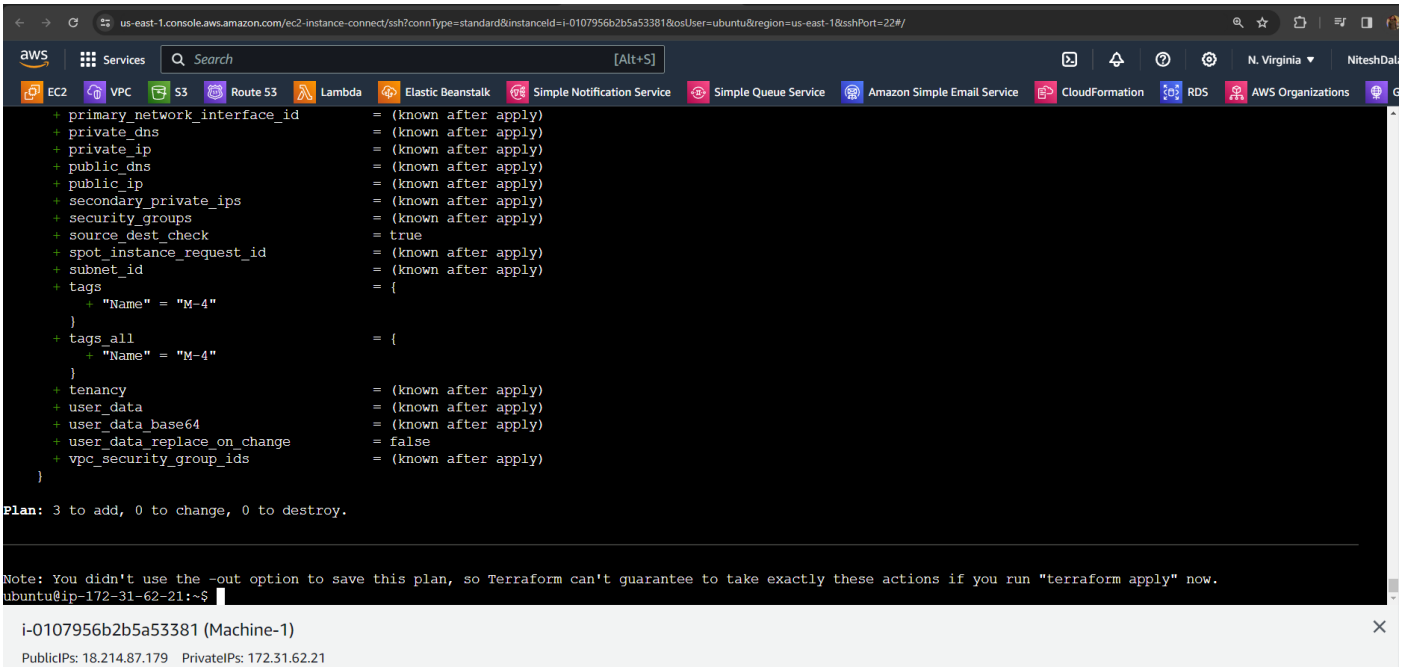
4. Now let's first initialize the terraform by terraform init command.

```
ubuntu@ip-172-31-62-21:~$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ip-172-31-62-21:~$
```

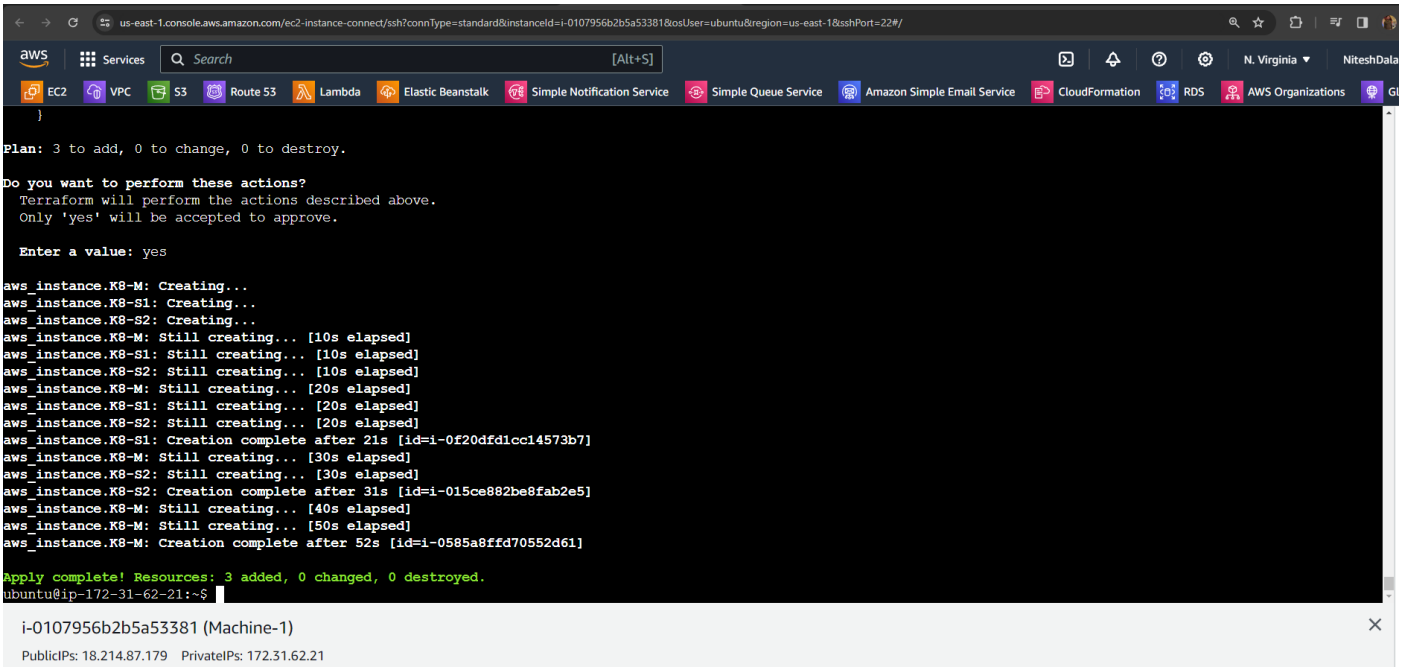
i-0107956b2b5a53381 (Machine-1)

PublicIPs: 18.214.87.179 PrivateIPs: 172.31.62.21

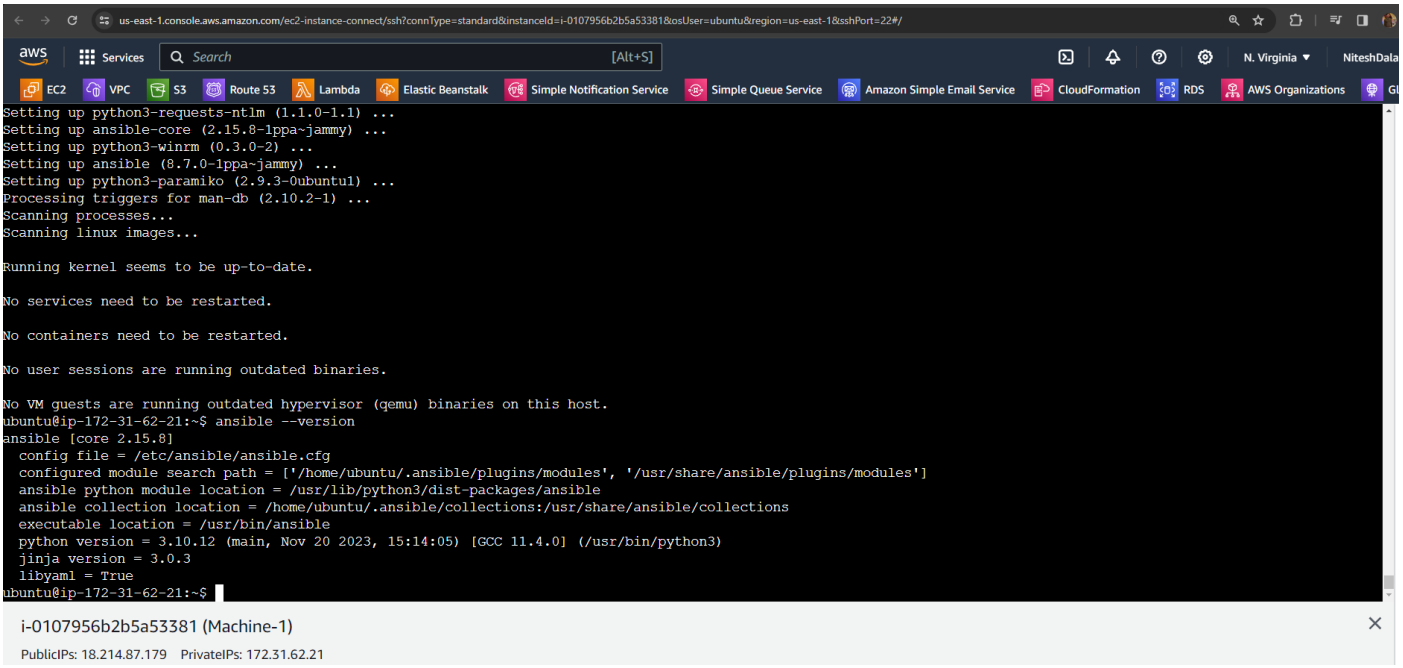
5. Now run the terraform plan command.



6. Now let's run the terraform apply command.

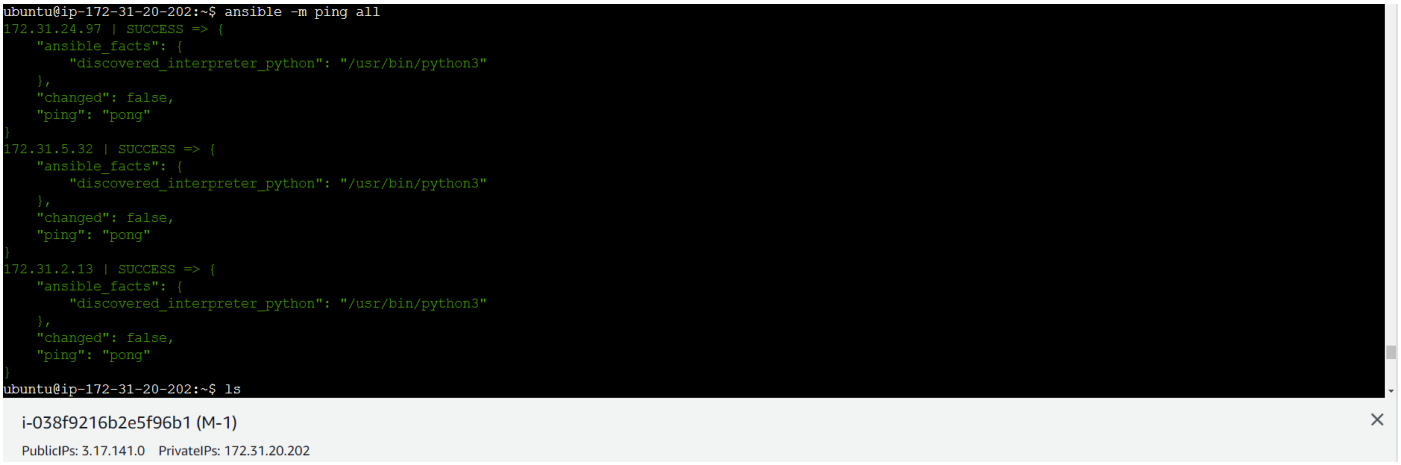


7. Now next task is to install ansible on Jenkins master to create the desired configuration of software's. Will follow the official documentation only.



8. After the ansible installation we will create password-less SSH connection b/w master machine i.e. M-1 and all others.

9. Now checking the connection.



10. Now creating three shell scripts to install the configuration.

```
ubuntu@ip-172-31-23-144:~$ cat w1.sh
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
ubuntu@ip-172-31-23-144:~$ cat w24.sh
sudo apt-get update
sudo apt-get install docker.io -y
sudo apt update
sudo apt upgrade -y
sudo apt install -y curl apt-transport-https ca-certificates software-properties-common
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo swapoff -a
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
ubuntu@ip-172-31-23-144:~$ cat w3.sh
sudo apt-get update
sudo apt install openjdk-11-jdk -y
sudo apt-get install docker.io -y
sudo apt update
sudo apt upgrade -y
sudo apt install -y curl apt-transport-https ca-certificates software-properties-common
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo swapoff -a
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
```

11. Creating a play.yaml file to execute this scripts.

```
---
- name: installing jenkins and java
  hosts: localhost
  become: true
  tasks:
  - name: executing script w1.sh
    script: w1.sh
- name: installing java, docker and k8s
  hosts: master
  become: true
  tasks:
  - name: executing w3.sh
    script: w3.sh
- name: installing k8s and docker in k8s slaves
  hosts: slave
  become: true
  tasks:
  - name: executing script w24.sh
    script: w24.sh
```

12. Successful execution of ansible playbook file complete.

```
ubuntu@ip-172-31-20-202:~$ ansible-playbook play.yaml

PLAY [installing jenkins and java] ********************************************************************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************************************************************************************
ok: [localhost]

TASK [executing script w1.sh] *************************************************************************************************************************************************************
changed: [localhost]

PLAY [installing java, docker and k8s] ****************************************************************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************************************************************************************
ok: [172.31.24.97]

TASK [executing w3.sh] ********************************************************************************************************************************************************************
changed: [172.31.24.97]

PLAY [installing k8s and docker in k8s slaves] ********************************************************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************************************************************************************
ok: [172.31.5.32]
ok: [172.31.2.13]

TASK [executing script w24.sh] ************************************************************************************************************************************************************
changed: [172.31.5.32]
```

i-038f9216b2e5f96b1 (M-1)

PublicIPs: 3.17.141.0 PrivateIPs: 172.31.20.202

13. Now to go to our Kubernetes master machine which is M-3 and initialize the cluster.


```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.24.97:6443 --token 2f6ycy.uvriu22qx2ypvobe \
        --discovery-token-ca-cert-hash sha256:4162d7b980f104895bae15eb4c3dc5526a16cd0a7fe2b58d74fdccda136dfe8a
```

i-060a066b2fe4e793f (M-3)

PublicIPs: 3.133.97.99 PrivateIPs: 172.31.24.97

14. Once done run these 3 commands as normal user in M-3.

```
ubuntu@ip-172-31-24-97:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-24-97:~$   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-24-97:~$   sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-24-97:~$ kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
```

15. In slave machine M-2&4 enter the token from master machine to join the network. But as root user.

```
ubuntu@ip-172-31-2-13:~$ sudo su
root@ip-172-31-2-13:/home/ubuntu# kubeadm join 172.31.24.97:6443 --token 2f6ycy.uvriu22qx2ypvobe \
>         --discovery-token-ca-cert-hash sha256:4162d7b980f104895bae15eb4c3dc5526a16cd0a7fe2b58d74fdccda136dfe8a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@ip-172-31-2-13:/home/ubuntu#
```

i-09417e11874c7473b (M-4)

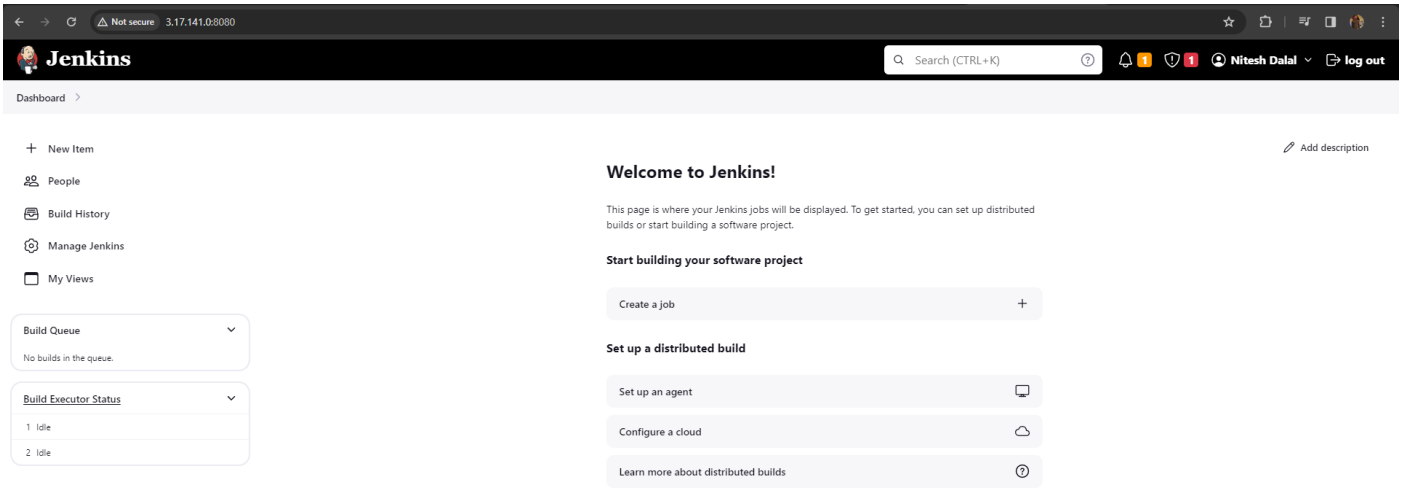
PublicIPs: 3.17.174.149 PrivateIPs: 172.31.2.13

```
ubuntu@ip-172-31-5-32:~$ sudo su
root@ip-172-31-5-32:/home/ubuntu# kubeadm join 172.31.24.97:6443 --token 2f6ycy.uvriu22qx2ypvobe \
>         --discovery-token-ca-cert-hash sha256:4162d7b980f104895bae15eb4c3dc5526a16cd0a7fe2b58d74fdccda136dfe8a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@ip-172-31-5-32:/home/ubuntu#
```

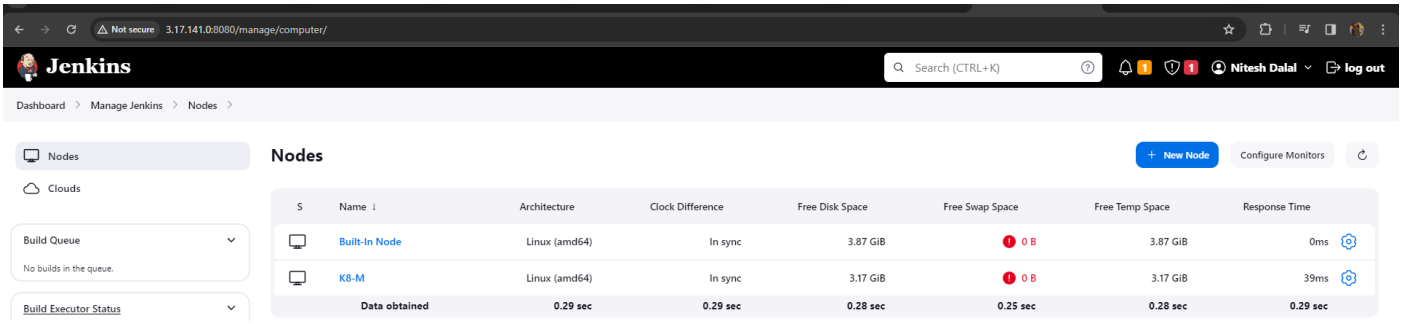
i-09bb5811236e6ef62 (M-2)

PublicIPs: 3.137.183.91 PrivateIPs: 172.31.5.32

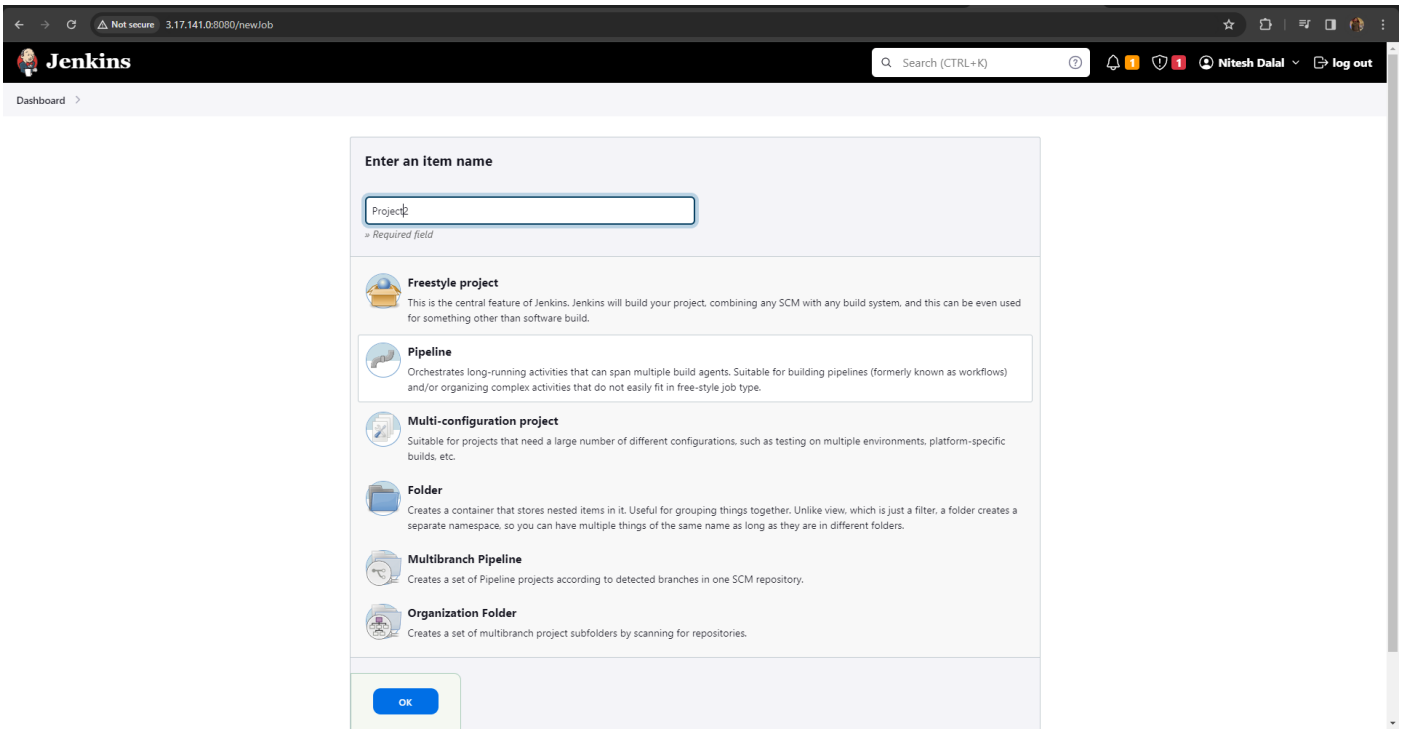
16. Now next task is to setup Jenkins dashboard from port 8080 of master machine i.e. M-1.



17. Add your M-3 machine as a node in Jenkins dashboard.



18. Now next we will create a job in Jenkins as a Pipeline.



19. Now we create a script for the pipeline, and also enable the github trigger.

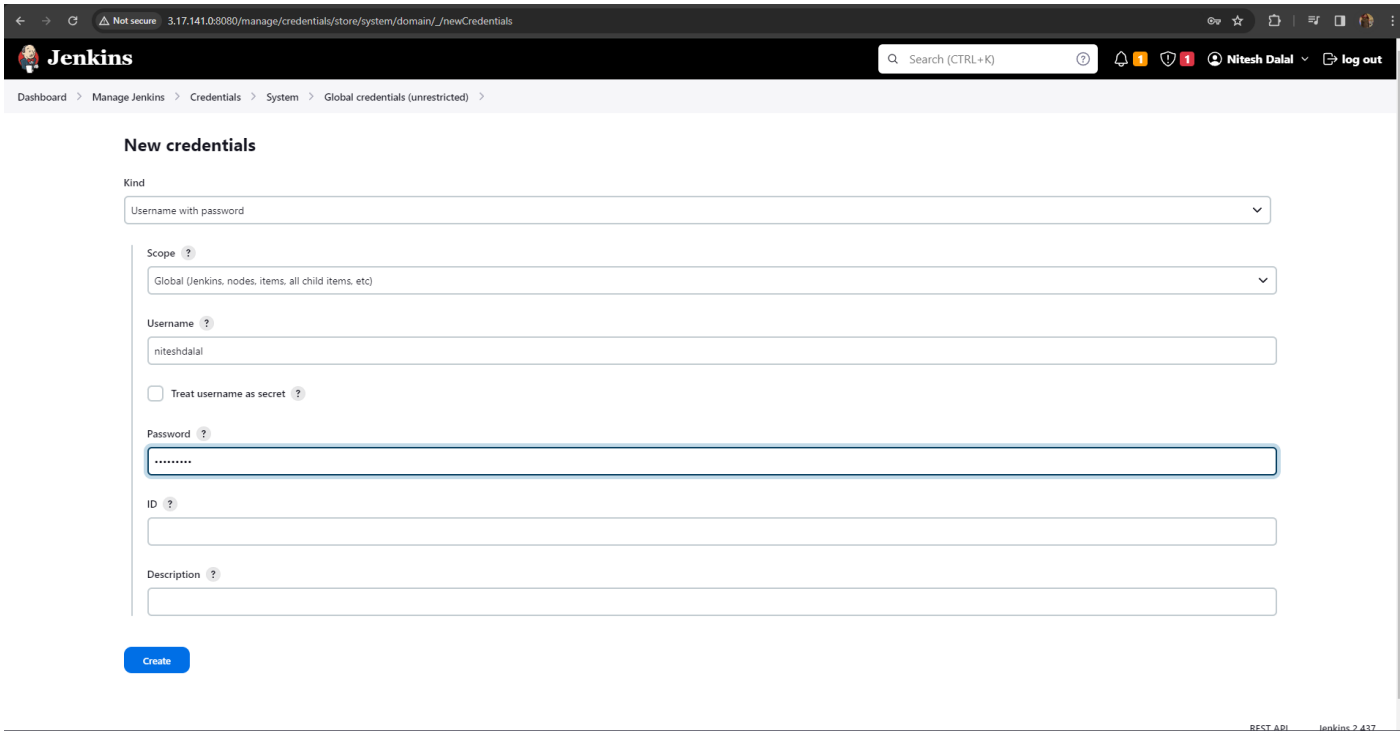
```
pipeline {
    agent none
    environment {
        DOCKERHUB_CREDENTIALS = credentials("7bd1da51-b1a5-4cd3-8667-326e832e3ea3")
    }
    stages {
        stage('Git') {
            agent {
                label 'KM'
            }
            steps {
                script {
                    git 'https://github.com/niteshdalal/website.git'
                }
            }
        }
        stage('Docker') {
```

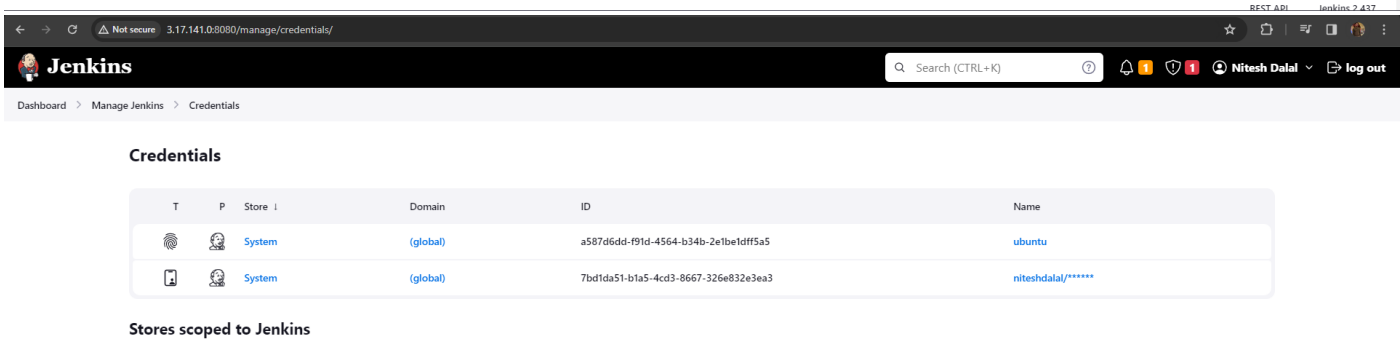
try sample Pipeline...

Script ?

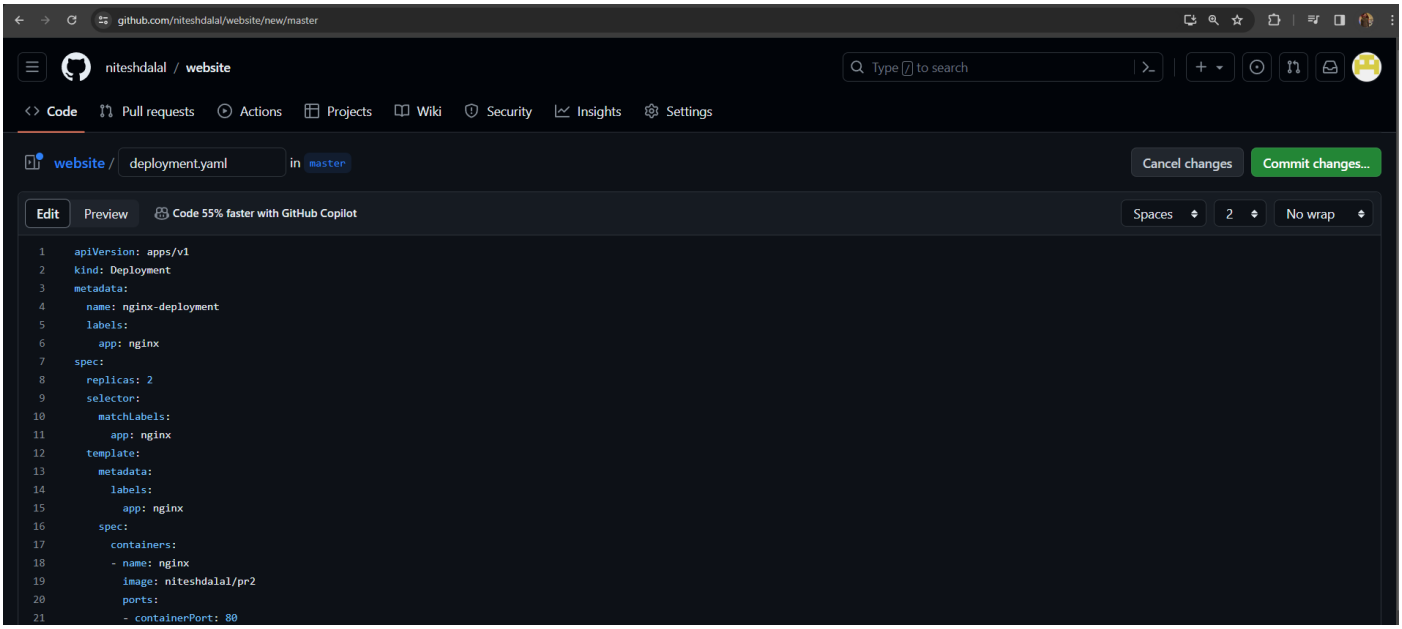
```
            agent {
                label 'KM'
            }
            steps {
                sh 'sudo docker build /home/ubuntu/jenkins/workspace/Project2/ -t niteshdalal/pr2'
                sh 'sudo docker login -u ${DOCKERHUB_CREDENTIALS_USR} -p ${DOCKERHUB_CREDENTIALS_PSW}'
                sh 'sudo docker push niteshdalal/pr2'
            }
        }
        stage('K8s') {
            agent {
                label 'KM'
            }
            steps {

                sh 'kubectl apply -f /home/ubuntu/jenkins/workspace/Project2/deployment.yaml'

                sh 'kubectl apply -f /home/ubuntu/jenkins/workspace/Project2/service.yaml'
            }
        }
    }
}
```

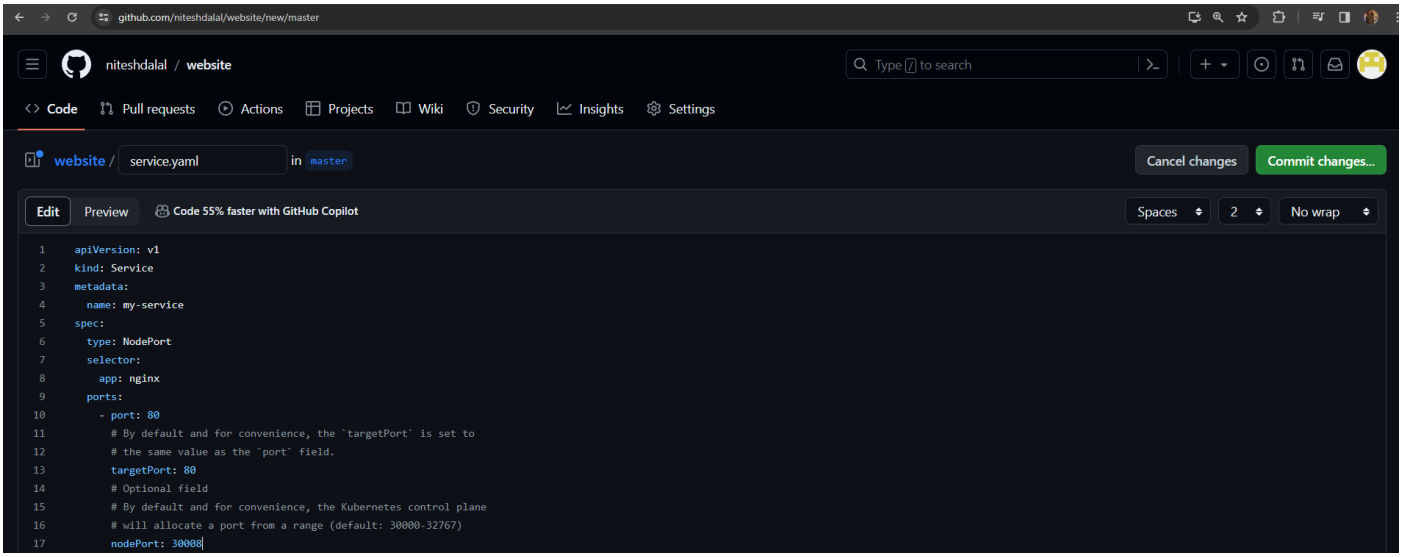
20. Since, we need our Dockerhub username & password for pushing the image to dockerhub. For this we will create a credential variable in Jenkins dashboard. It will be used in groovy script in Jenkins pipeline.



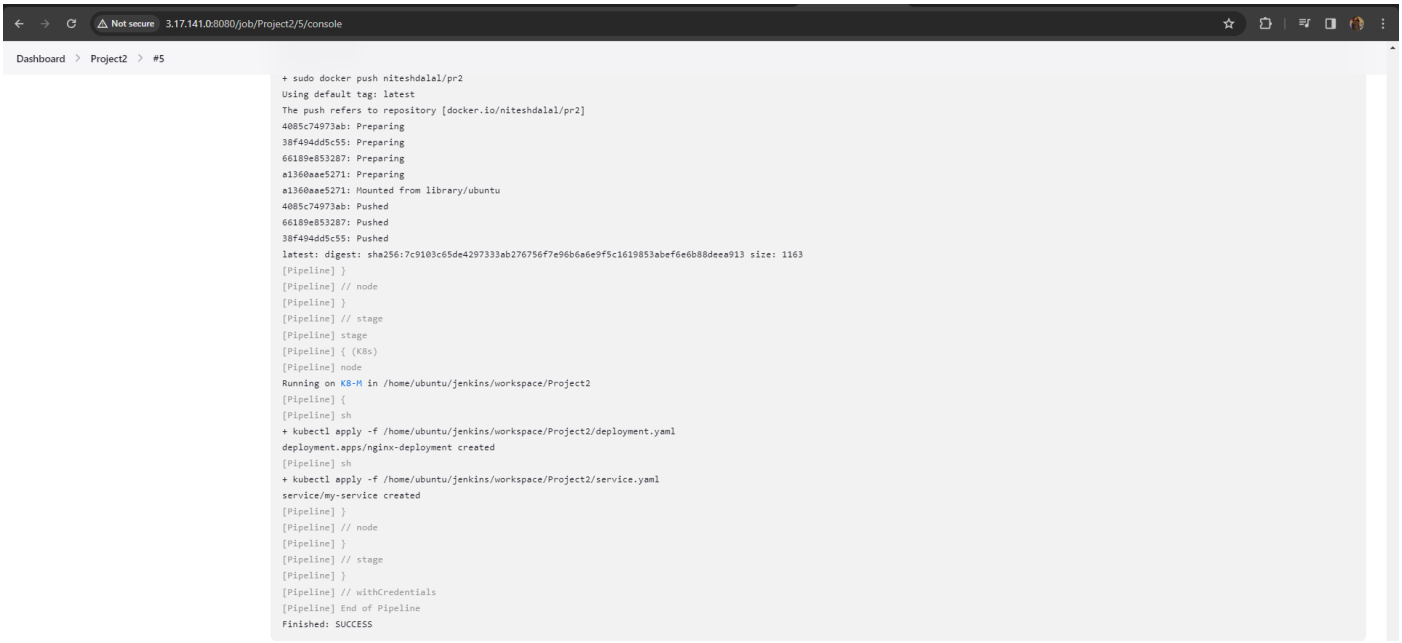


21. Now we will create two manifest file on github. One for deployment and one for service.



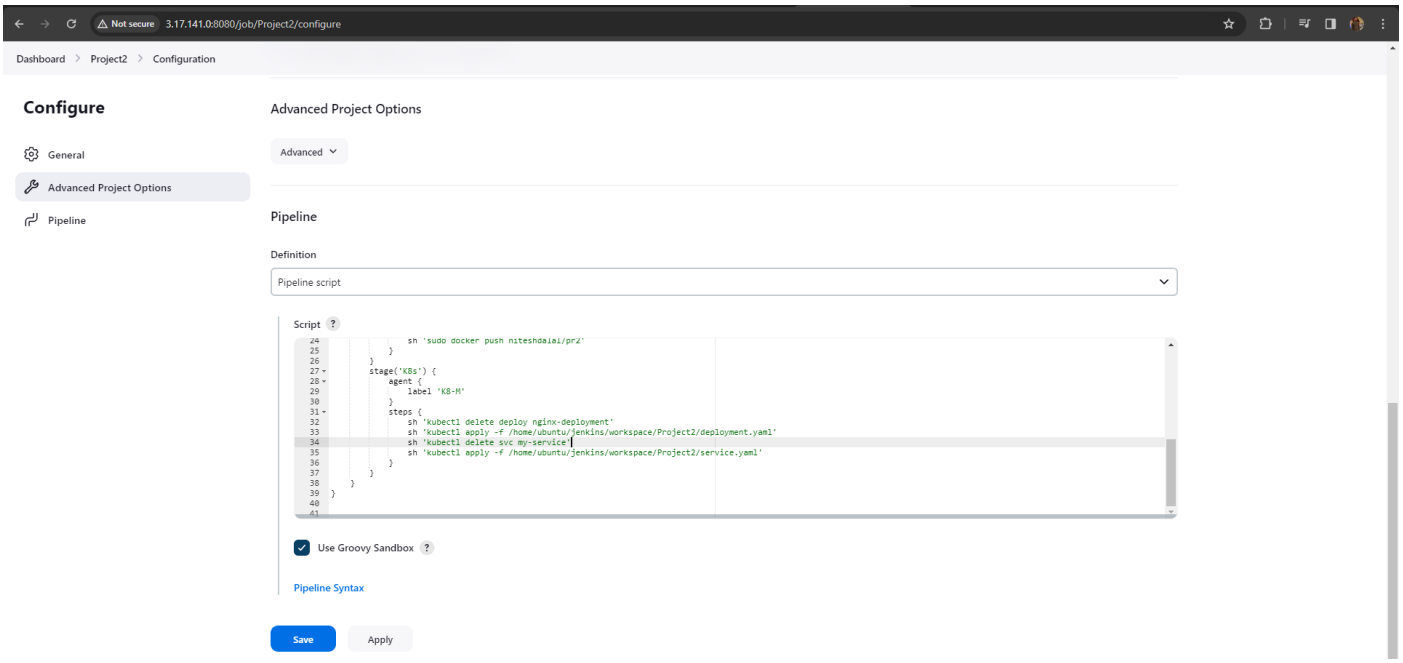


22. Once done go ahead and do you first build.

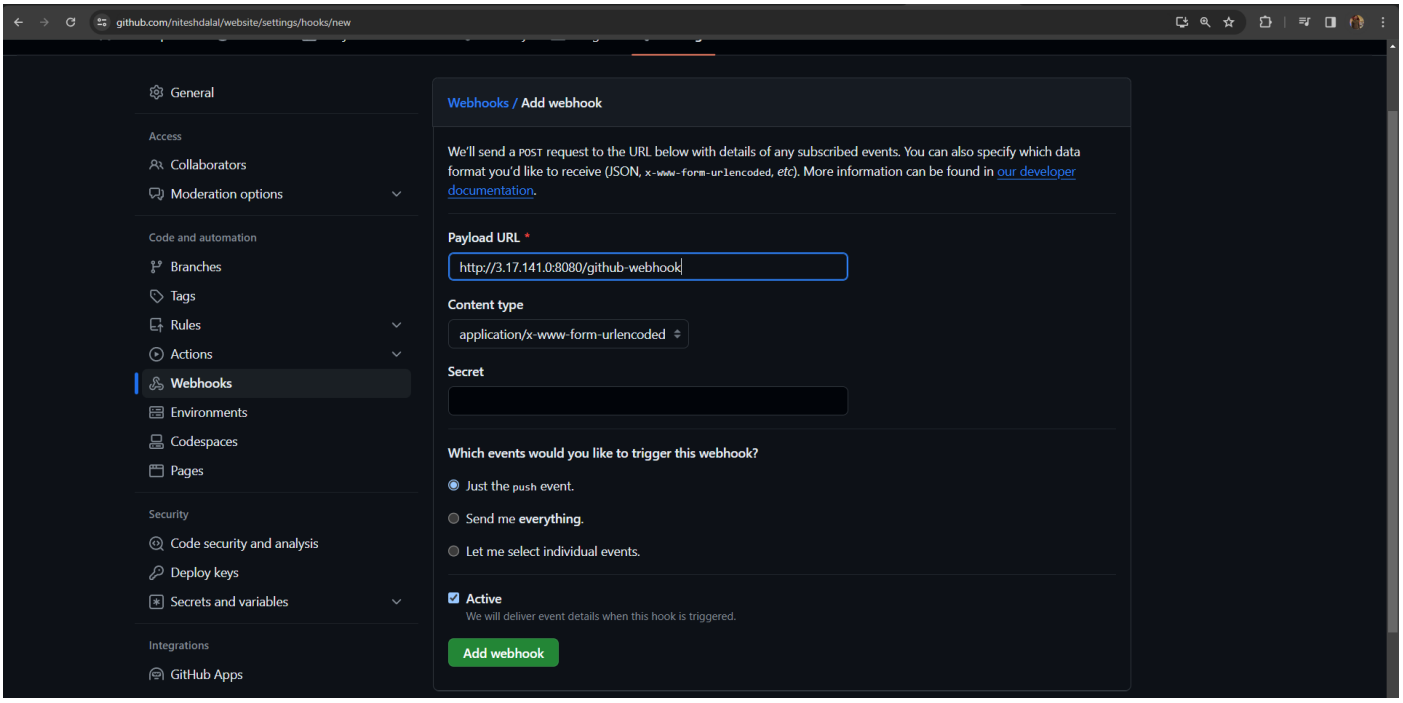


23. If it runs successfully then you should be able to see the website of public IP of worker2&4 followed by the service port 30008.

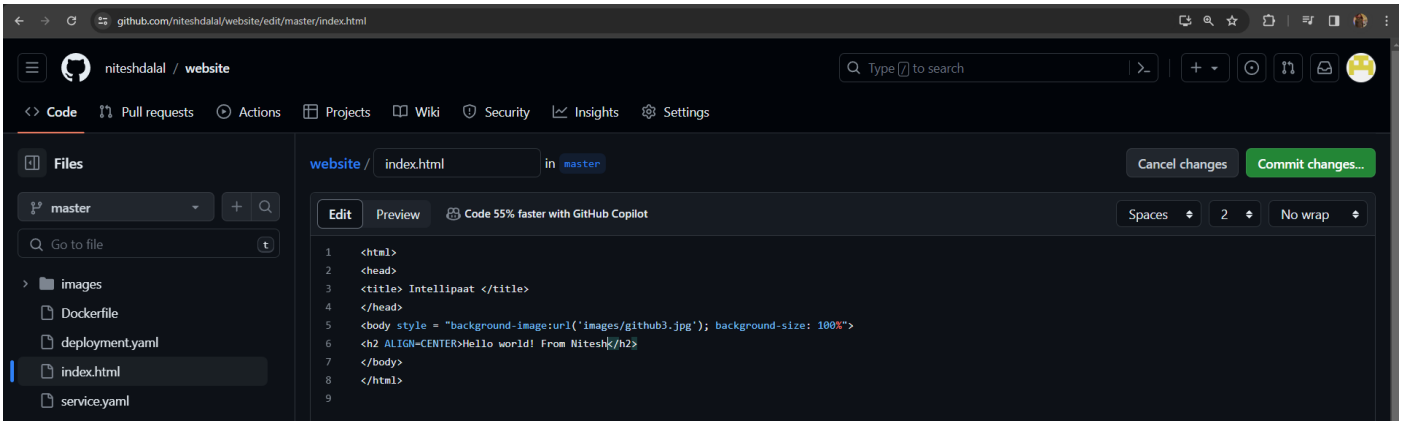
24. Now to avoid error of container already exists. We need to modify the groovy script. I added two lines to force kubernetes to delete old deployment & service every time a new changes occur.



25. Also let's create a webhook for project to ensure code build run automatically.



26. Let's try changing website content to see of code build triggers job automatically.



See the H2 content got changed.