DevOps Project 2

Using Terraform to setup the environment in Azure.

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "=3.34.0"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = "e9c93904-c316-4f60-a1b3-fc5ae575878f"
  tenant_id       = "624a0ea3-8660-4927-96de-a646f74ad52b"
  client_id       = "0feeb28b-8afe-4960-8e65-4c7f823bc736"
  client_secret   = "Z~r8Q~vq5lLuxXDRjWnMlh9zLn6KcmZ09sH~GcA2"
}

resource "azurerm_resource_group" "devops" {
  name     = "devops-rg"
  location = "West Europe"
}

resource "azurerm_virtual_network" "devops" {
  name                = "devops-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
}

resource "azurerm_subnet" "devops" {
  name                 = "devops-subnet"
  resource_group_name  = azurerm_resource_group.devops.name
  virtual_network_name = azurerm_virtual_network.devops.name
  address_prefixes     = ["10.0.9.0/24"]
}

resource "azurerm_ssh_public_key" "devops" {
  name                = "devops"
  resource_group_name = azurerm_resource_group.devops.name
  location            = azurerm_resource_group.devops.location
  public_key          = file("~/.ssh/id_rsa.pub")
}

resource "azurerm_network_security_group" "devops" {
  name                = "devops-nsg"
  resource_group_name = azurerm_resource_group.devops.name
  location            = azurerm_resource_group.devops.location

  security_rule {
```

```
    name                       = "SSH"
    priority                   = 100
    access                     = "Allow"
    direction                  = "Inbound"
    protocol                   = "Tcp"
    source_address_prefix      = "*"
    source_port_range          = "*"
    destination_address_prefix = "*"
    destination_port_range     = "22"
  }
  security_rule {
    name                       = "HTTP"
    priority                   = 101
    access                     = "Allow"
    direction                  = "Inbound"
    protocol                   = "Tcp"
    source_address_prefix      = "*"
    source_port_range          = "*"
    destination_address_prefix = "*"
    destination_port_range     = "80"
  }
  security_rule {
    name                       = "Jenkins"
    priority                   = 102
    access                     = "Allow"
    direction                  = "Inbound"
    protocol                   = "Tcp"
    source_address_prefix      = "*"
    source_port_range          = "*"
    destination_address_prefix = "*"
    destination_port_range     = "8080"
  }

}

// Jenkins
resource "azurerm_public_ip" "jenkins" {
  name                = "jenkins-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}

resource "azurerm_network_interface" "jenkins" {
  name                = "jenkins-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "jenkins-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.jenkins.id
  }
```

```
}

resource "azurerm_network_interface_security_group_association" "jenkins" {
  network_interface_id      = azurerm_network_interface.jenkins.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "jenkins" {
  name                             = "Jenkins"
  location                         = azurerm_resource_group.devops.location
  resource_group_name              = azurerm_resource_group.devops.name
  vm_size                          = "Standard_D2s_v3"
  network_interface_ids            = [azurerm_network_interface.jenkins.id]
  delete_data_disks_on_termination = true
  delete_os_disk_on_termination    = true

  storage_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts-gen2"
    version   = "latest"
  }

  storage_os_disk {
    name              = "jenkins-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "jenkins"
    admin_username = "azureuser"
    admin_password = "Password@123"
  }

  os_profile_linux_config {
    disable_password_authentication = true
    ssh_keys {
      path     = "/home/azureuser/.ssh/authorized_keys"
      key_data = azurerm_ssh_public_key.devops.public_key
    }
  }

}

// Kubernetes
resource "azurerm_public_ip" "kubernetes" {
  name                = "kubernetes-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}
```

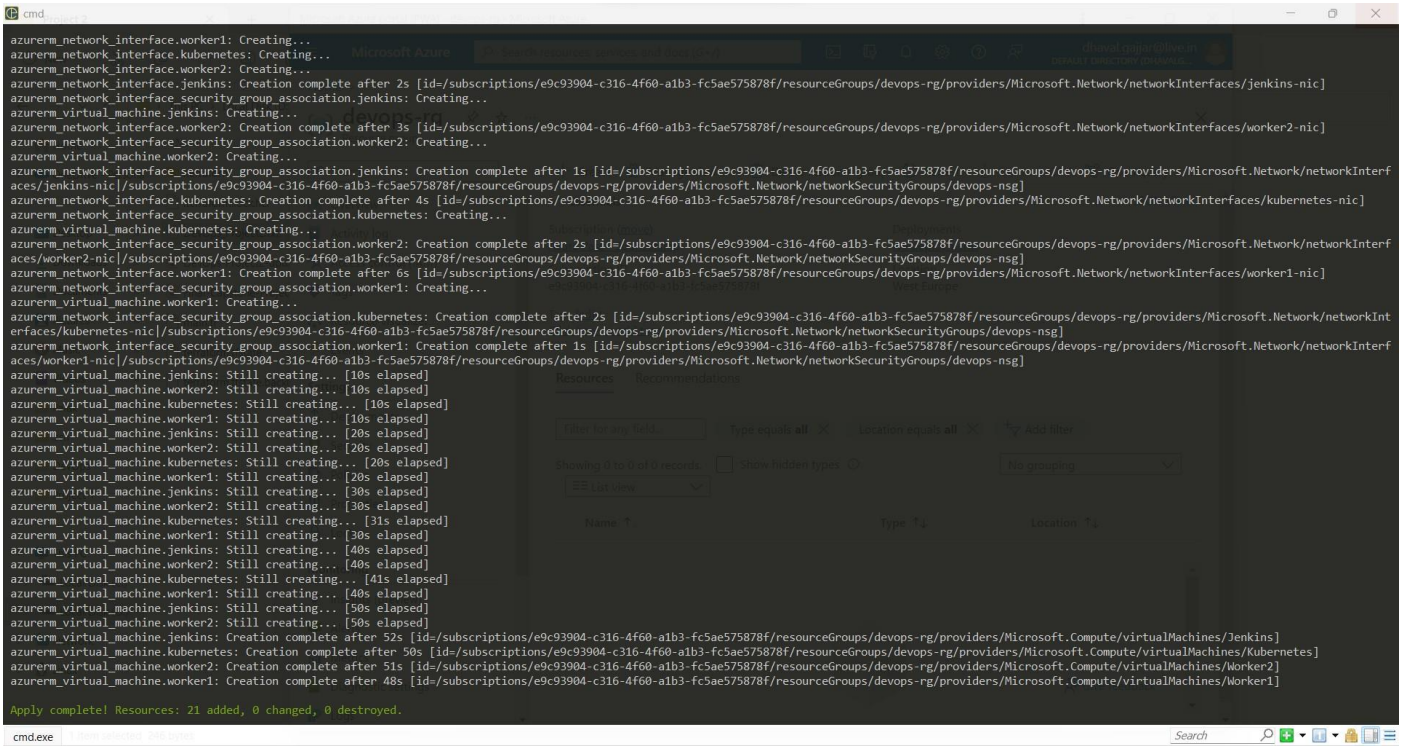
```
resource "azurerm_network_interface" "kubernetes" {
  name                = "kubernetes-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "kubernetes-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.kubernetes.id
  }
}

resource "azurerm_network_interface_security_group_association" "kubernetes" {
  network_interface_id      = azurerm_network_interface.kubernetes.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "kubernetes" {
  name                             = "Kubernetes"
  location                         = azurerm_resource_group.devops.location
  resource_group_name              = azurerm_resource_group.devops.name
  vm_size                          = "Standard_D2s_v3"
  network_interface_ids            = [azurerm_network_interface.kubernetes.id]
  delete_data_disks_on_termination = true
  delete_os_disk_on_termination    = true

  storage_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts-gen2"
    version   = "latest"
  }

  storage_os_disk {
    name              = "kubernetes-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "kubernetes"
    admin_username = "azureuser"
    admin_password = "Password@123"
  }

  os_profile_linux_config {
    disable_password_authentication = true
    ssh_keys {
      path     = "/home/azureuser/.ssh/authorized_keys"
      key_data = azurerm_ssh_public_key.devops.public_key
    }
  }
```

```
}

// Slave 1
resource "azurerm_public_ip" "worker1" {
  name                = "worker1-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}

resource "azurerm_network_interface" "worker1" {
  name                = "worker1-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "worker1-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.worker1.id
  }
}

resource "azurerm_network_interface_security_group_association" "worker1" {
  network_interface_id      = azurerm_network_interface.worker1.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "worker1" {
  name                             = "Worker1"
  location                         = azurerm_resource_group.devops.location
  resource_group_name              = azurerm_resource_group.devops.name
  vm_size                          = "Standard_D2s_v3"
  network_interface_ids            = [azurerm_network_interface.worker1.id]
  delete_data_disks_on_termination = true
  delete_os_disk_on_termination    = true

  storage_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts-gen2"
    version   = "latest"
  }

  storage_os_disk {
    name              = "worker1-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "worker1"
```

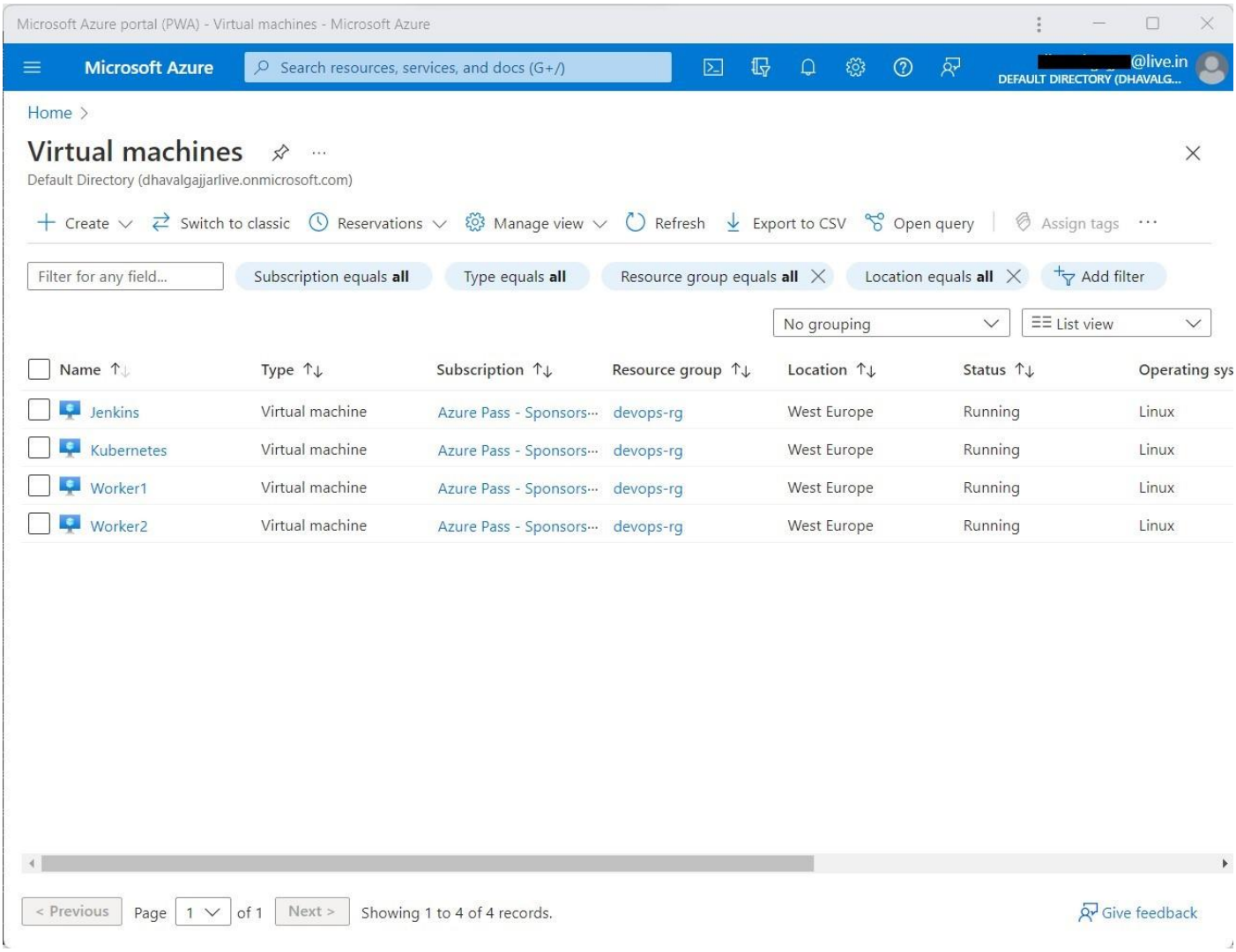
```
    admin_username = "azureuser"
    admin_password = "Password@123"
  }

  os_profile_linux_config {
    disable_password_authentication = true
    ssh_keys {
      path     = "/home/azureuser/.ssh/authorized_keys"
      key_data = azurerm_ssh_public_key.devops.public_key
    }
  }

}

// Worker 2
resource "azurerm_public_ip" "worker2" {
  name                = "worker2-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}

resource "azurerm_network_interface" "worker2" {
  name                = "worker2-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "worker2-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.worker2.id
  }
}

resource "azurerm_network_interface_security_group_association" "worker2" {
  network_interface_id      = azurerm_network_interface.worker2.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "worker2" {
  name                             = "Worker2"
  location                         = azurerm_resource_group.devops.location
  resource_group_name              = azurerm_resource_group.devops.name
  vm_size                          = "Standard_D2s_v3"
  network_interface_ids            = [azurerm_network_interface.worker2.id]
  delete_data_disks_on_termination = true
  delete_os_disk_on_termination    = true

  storage_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts-gen2"
    version   = "latest"
```

```
  }

  storage_os_disk {
    name              = "worker2-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "worker2"
    admin_username = "azureuser"
    admin_password = "Password@123"
  }

  os_profile_linux_config {
    disable_password_authentication = true
    ssh_keys {
      path     = "/home/azureuser/.ssh/authorized_keys"
      key_data = azurerm_ssh_public_key.devops.public_key
    }
  }

}
```

$ terraform init
$ terraform plan
$ terraform apply

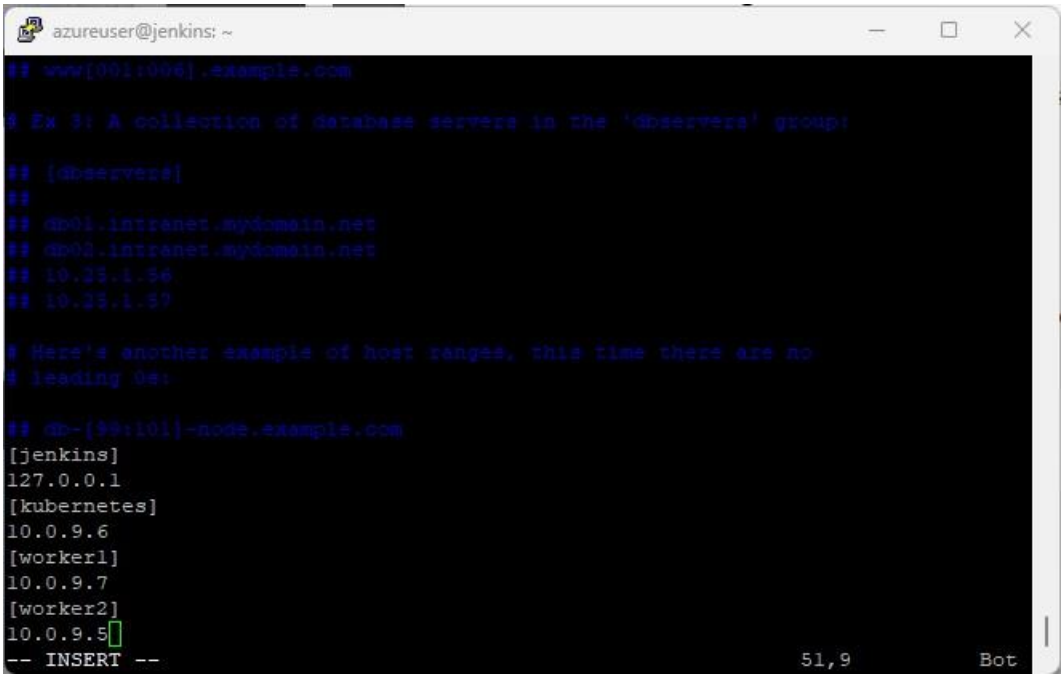


All VMs are created.



Install ansible.

$ sudo apt-get update
$ sudo apt install software-properties-common aptitude -y
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo aptitude install ansible -y



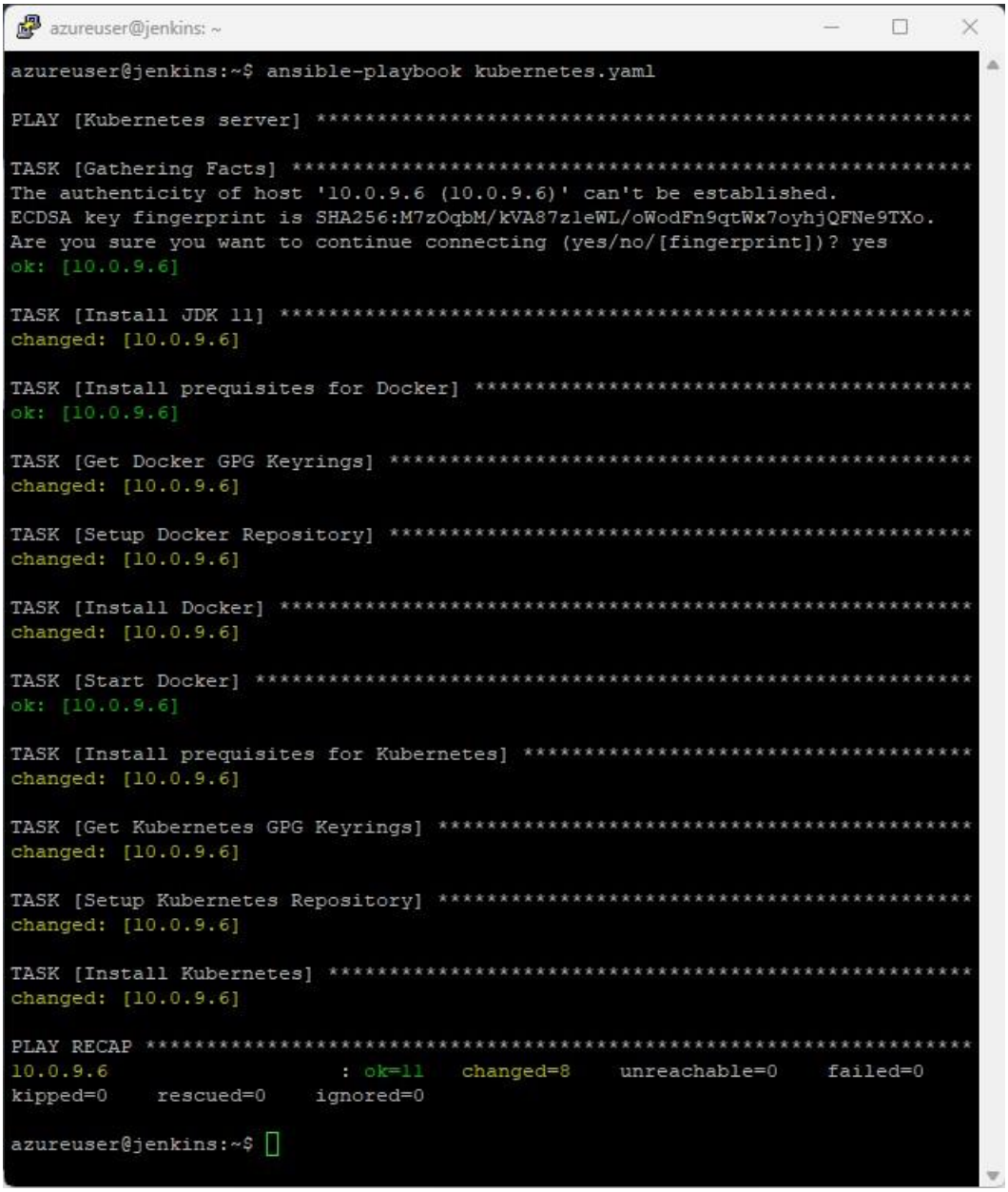
Using ansible-playbook setup jenkins machine softwares.

```
---
- name: Jenkins server
  hosts: jenkins
  remote_user: root
  tasks:
    - name: Install JDK 11
      apt:
        name: openjdk-11-jdk
        update_cache: true
    - name: Get Jenkins Keyrings
      get_url:
        url: https://pkg.jenkins.io/debian-stable/jenkins.io.key
        dest: /usr/share/keyrings/jenkins-keyring.asc
    - name: Configure Jenkins Repository
      apt_repository:
        repo: deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/
        state: present
    - name: Install Jenkins
      apt:
        name: jenkins
        update_cache: true
    - name: Start Jenkins
      systemd:
        name: jenkins
        enabled: true
        masked: no
        daemon_reload: yes
```

Using ansible-playbook setup kubernetes machine softwares.

```
---
- name: Kubernetes server
  hosts: kubernetes
  remote_user: root
  tasks:
    - name: Install JDK 11
      apt:
        name: openjdk-11-jdk
        update_cache: true
    - name: Install prequisites for Docker
      apt:
        pkg:
          - ca-certificates
          - curl
          - gnupg
          - lsb-release
    - name: Get Docker GPG Keyrings
      shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o
/usr/share/keyrings/docker.gpg
    - name: Setup Docker Repository
```

```
      shell: echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
    - name: Install Docker
      apt:
        pkg:
          - docker-ce
          - docker-ce-cli
          - containerd.io
          - docker-compose-plugin
        update_cache: true
    - name: Start Docker
      systemd:
        name: docker
        enabled: true
        masked: no
        daemon_reload: yes
    - name: Install prequisites for Kubernetes
      apt:
        pkg:
          - ca-certificates
          - curl
          - apt-transport-https
    - name: Get Kubernetes GPG Keyrings
      shell: curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg --
dearmor -o /usr/share/keyrings/kubernetes.gpg
    - name: Setup Kubernetes Repository
      shell: echo "deb [signed-by=/usr/share/keyrings/kubernetes.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | tee
/etc/apt/sources.list.d/kubernetes.list > /dev/null
    - name: Install Kubernetes
      apt:
        pkg:
          - kubelet
          - kubeadm
          - kubectl
        update_cache: true
```



Using ansible-playbook setup worker1 machine softwares.

```
---
- name: Worker1 server
  hosts: worker1
  remote_user: root
  tasks:
    - name: Install prequisites for Docker
      apt:
        pkg:
          - ca-certificates
          - curl
          - gnupg
          - lsb-release
    - name: Get Docker GPG Keyrings
      shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o
/usr/share/keyrings/docker.gpg
    - name: Setup Docker Repository
      shell: echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
    - name: Install Docker
      apt:
        pkg:
          - docker-ce
          - docker-ce-cli
          - containerd.io
          - docker-compose-plugin
        update_cache: true
    - name: Start Docker
      systemd:
        name: docker
        enabled: true
        masked: no
        daemon_reload: yes
    - name: Install prequisites for Kubernetes
      apt:
        pkg:
          - ca-certificates
          - curl
          - apt-transport-https
    - name: Get Kubernetes GPG Keyrings
      shell: curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg --
dearmor -o /usr/share/keyrings/kubernetes.gpg
    - name: Setup Kubernetes Repository
      shell: echo "deb [signed-by=/usr/share/keyrings/kubernetes.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | tee
/etc/apt/sources.list.d/kubernetes.list > /dev/null
    - name: Install Kubernetes
      apt:
        pkg:
          - kubelet
          - kubeadm
          - kubectl
        update_cache: true
```

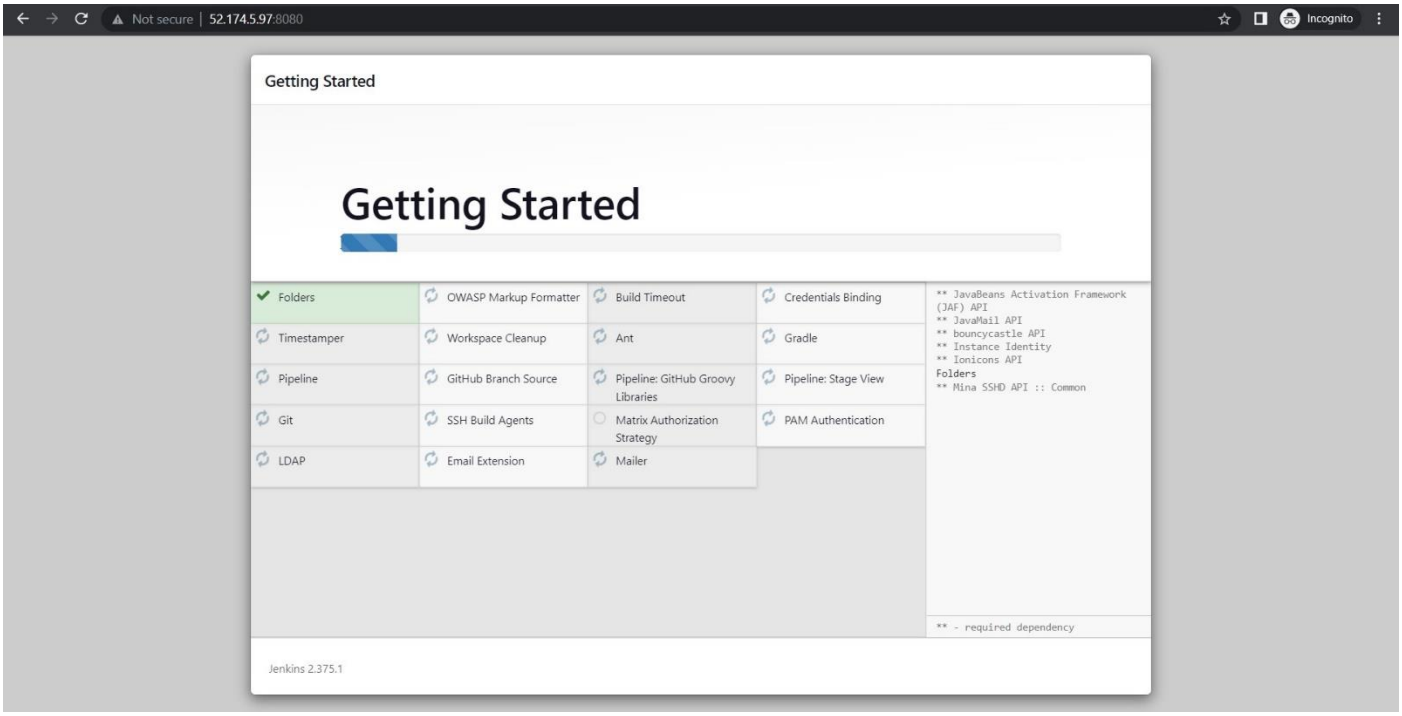
Using ansible-playbook setup worker2 machine softwares.

```
---
- name: Worker2 server
  hosts: worker2
  remote_user: root
  tasks:
    - name: Install prequisites for Docker
      apt:
        pkg:
          - ca-certificates
          - curl
          - gnupg
          - lsb-release
    - name: Get Docker GPG Keyrings
```

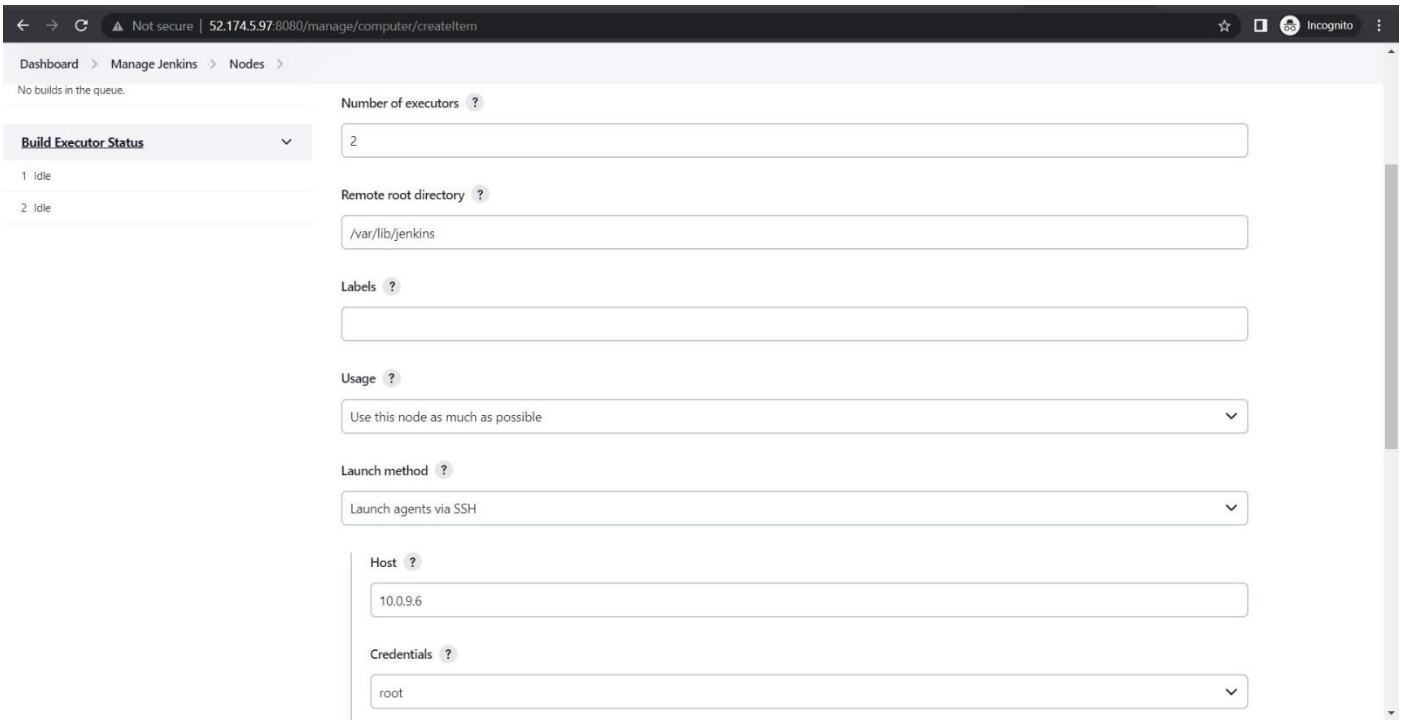
```
      shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o
/usr/share/keyrings/docker.gpg
    - name: Setup Docker Repository
      shell: echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
    - name: Install Docker
      apt:
        pkg:
          - docker-ce
          - docker-ce-cli
          - containerd.io
          - docker-compose-plugin
        update_cache: true
    - name: Start Docker
      systemd:
        name: docker
        enabled: true
        masked: no
        daemon_reload: yes
    - name: Install prequisites for Kubernetes
      apt:
        pkg:
          - ca-certificates
          - curl
          - apt-transport-https
    - name: Get Kubernetes GPG Keyrings
      shell: curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg --
dearmor -o /usr/share/keyrings/kubernetes.gpg
    - name: Setup Kubernetes Repository
      shell: echo "deb [signed-by=/usr/share/keyrings/kubernetes.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | tee
/etc/apt/sources.list.d/kubernetes.list > /dev/null
    - name: Install Kubernetes
      apt:
        pkg:
          - kubelet
          - kubeadm
          - kubectl
        update_cache: true
```


azureuser@jenkins: ~
azureuser@jenkins:~$ ansible-playbook worker2.yaml
PLAY [Worker2 server] ***
*
TASK [Gathering Facts] **
*
The authenticity of host '10.0.9.5 (10.0.9.5)' can't be established.
ECDSA key fingerprint is SHA256:eOOIL/iYqK+4cURmkUS7VfGpYep2iw6VaxHZNd1zZ9Q.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
ok: [10.0.9.5]
TASK [Install prequisites for Docker] ***
*
ok: [10.0.9.5]
TASK [Get Docker GPG Keyrings] **
*
changed: [10.0.9.5]
TASK [Setup Docker Repository] **
*
changed: [10.0.9.5]
TASK [Install Docker] ***
*
changed: [10.0.9.5]
TASK [Start Docker] ***
*
ok: [10.0.9.5]
TASK [Install prequisites for Kubernetes] ***************************************
*
changed: [10.0.9.5]
TASK [Get Kubernetes GPG Keyrings] **
*
changed: [10.0.9.5]
TASK [Setup Kubernetes Repository] **
*
changed: [10.0.9.5]
TASK [Install Kubernetes] ***
*
changed: [10.0.9.5]
PLAY RECAP **
*
10.0.9.5 : ok=10 changed=7 unreachable=0 failed=0
skipped=0 rescued=0 ignored=0
azureuser@jenkins:~$

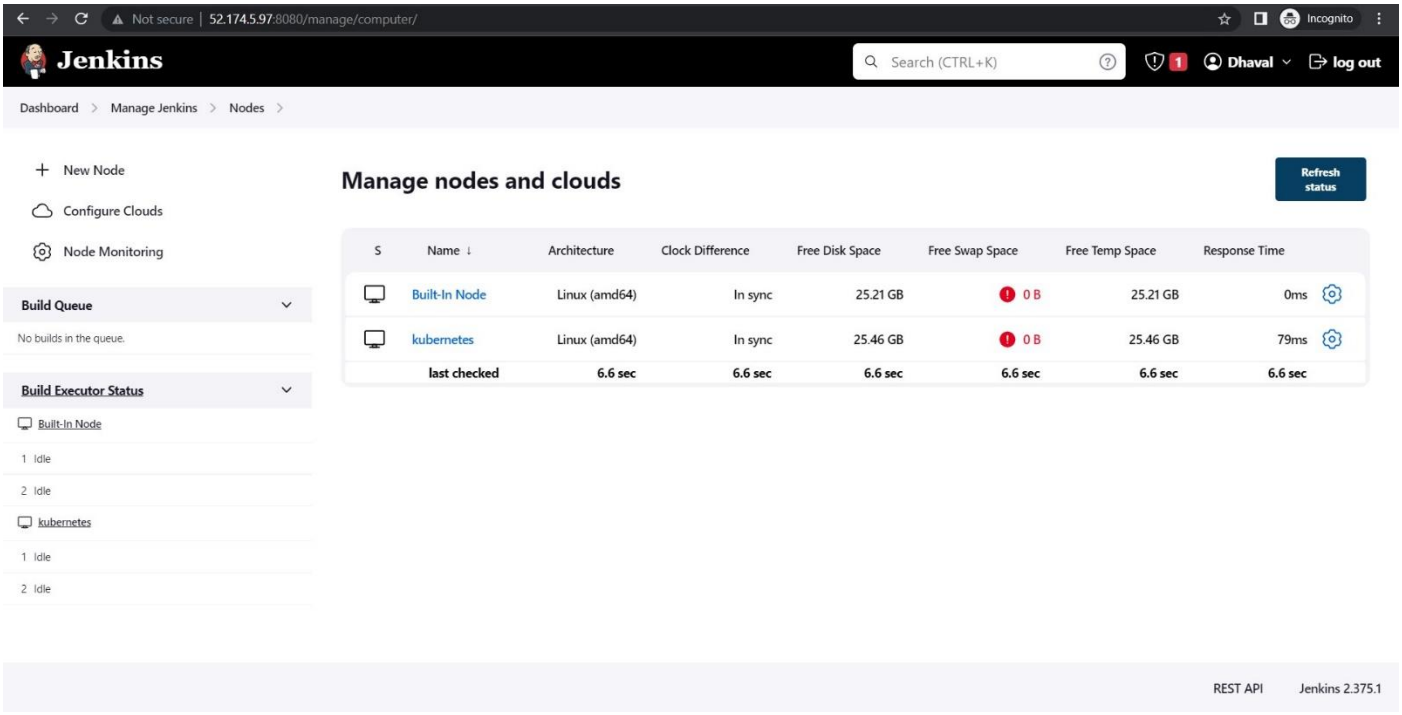
Setup the Jenkins.



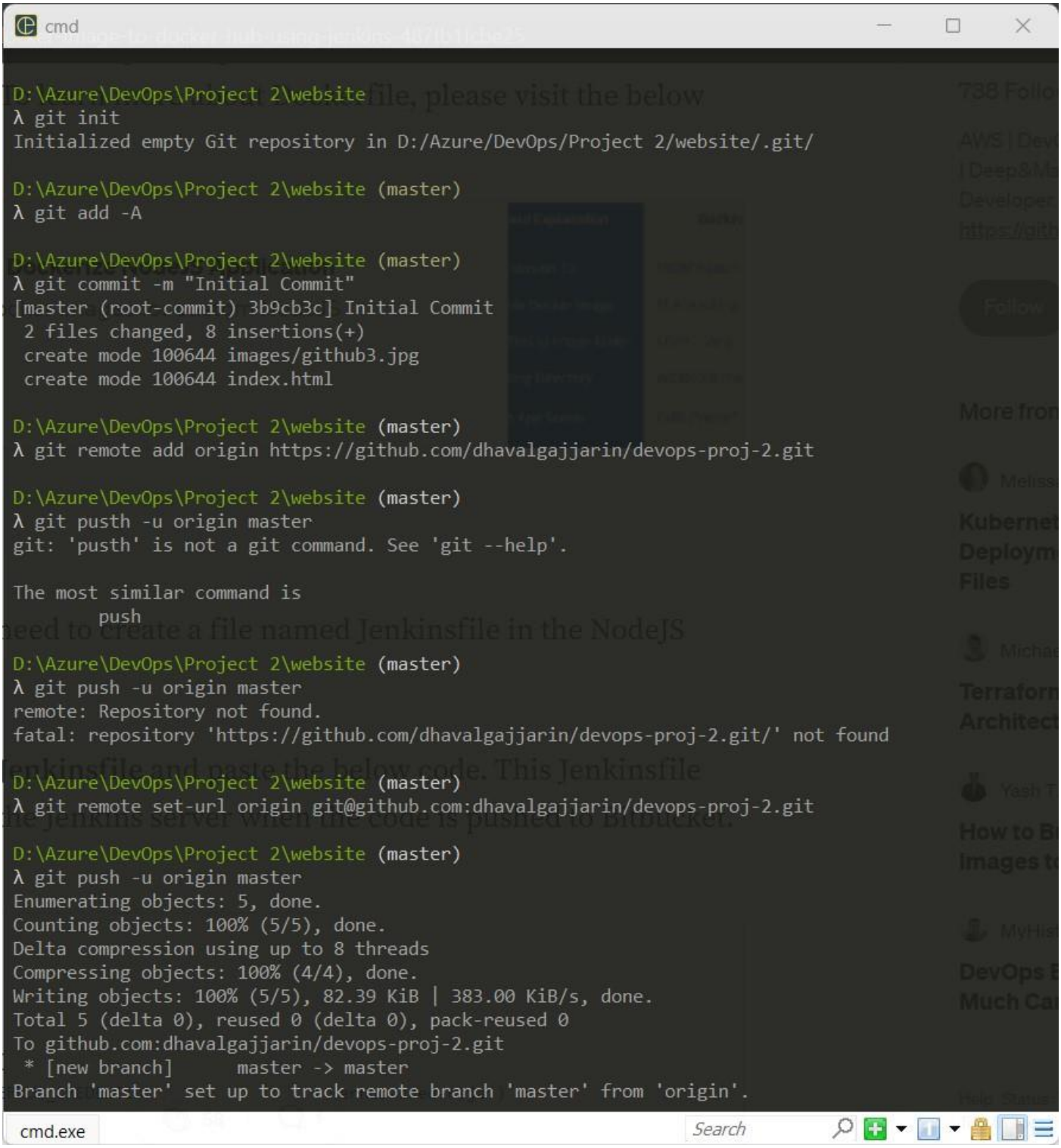
Setup new Jenkins node for kubernetes environment.



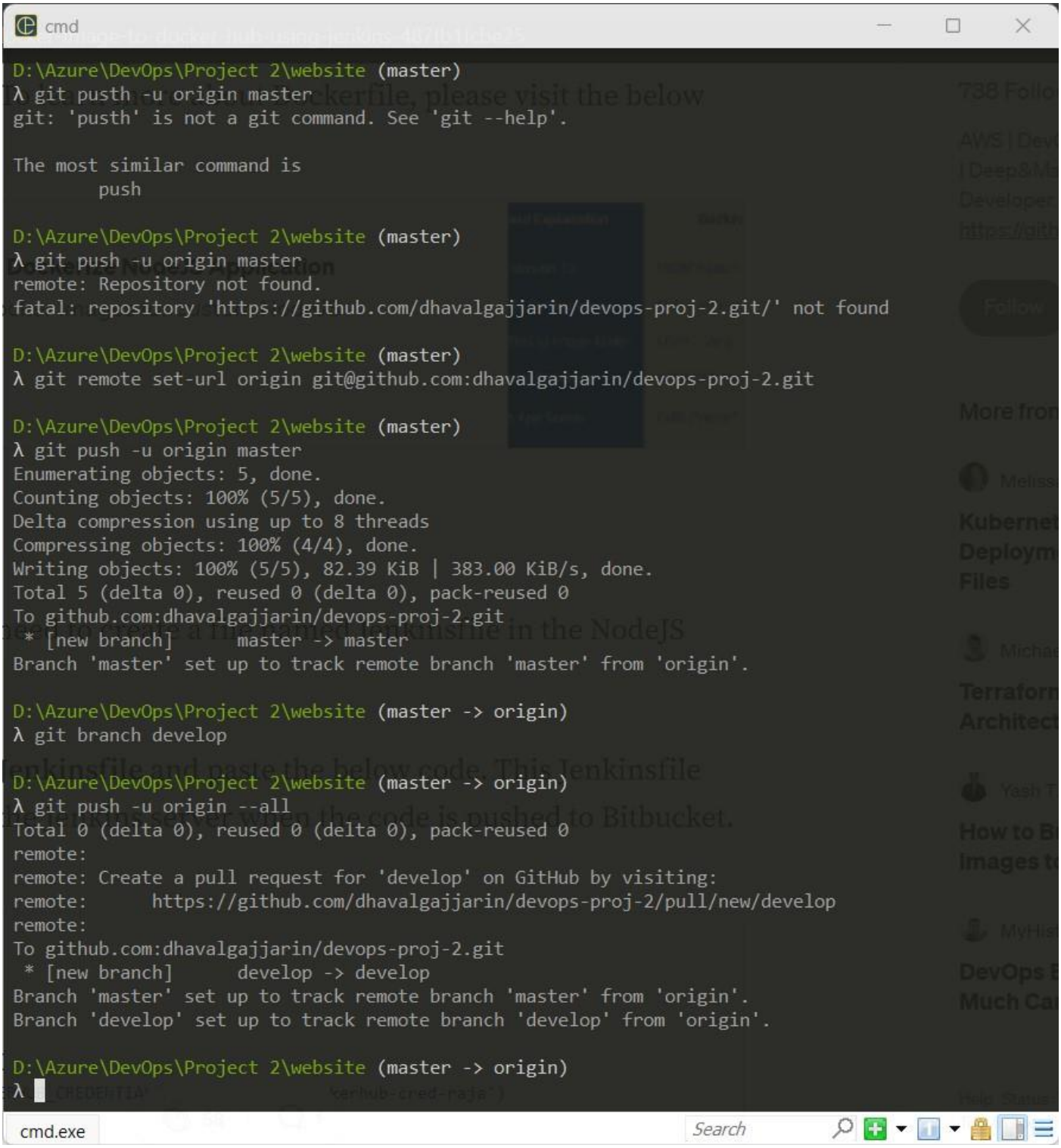
Make sure nodes are up and online.



Setup the initial commit to GitHub repository in the master branch.

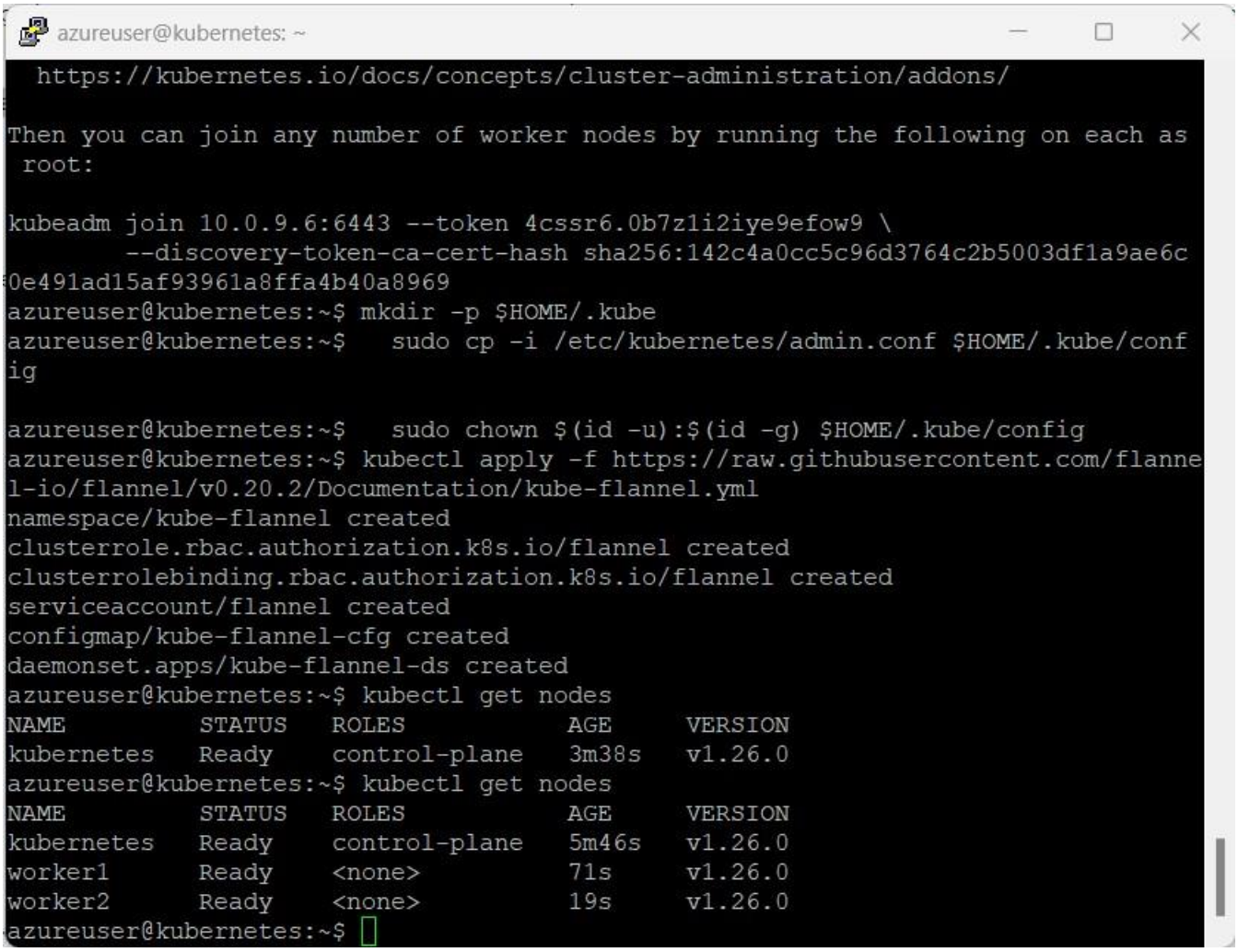

cmd
D:\Azure\DevOps\Project 2\website
λ git init
Initialized empty Git repository in D:/Azure/DevOps/Project 2/website/.git/
D:\Azure\DevOps\Project 2\website (master)
λ git add -A
D:\Azure\DevOps\Project 2\website (master)
λ git commit -m "Initial Commit"
[master (root-commit) 3b9cb3c] Initial Commit
2 files changed, 8 insertions(+)
create mode 100644 images/github3.jpg
create mode 100644 index.html
D:\Azure\DevOps\Project 2\website (master)
λ git remote add origin https://github.com/dhavalgajjarin/devops-proj-2.git
D:\Azure\DevOps\Project 2\website (master)
λ git pusth -u origin master
git: 'pusth' is not a git command. See 'git --help'.
The most similar command is
push
D:\Azure\DevOps\Project 2\website (master)
λ git push -u origin master
remote: Repository not found.
fatal: repository 'https://github.com/dhavalgajjarin/devops-proj-2.git/' not found
D:\Azure\DevOps\Project 2\website (master)
λ git remote set-url origin git@github.com:dhavalgajjarin/devops-proj-2.git
D:\Azure\DevOps\Project 2\website (master)
λ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 82.39 KiB | 383.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dhavalgajjarin/devops-proj-2.git
* [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
cmd.exe
Search

Create GitHub workflow with creating develop branch.



Setup the Kubernetes cluster.

Join the node and setup the Kubernetes networking.

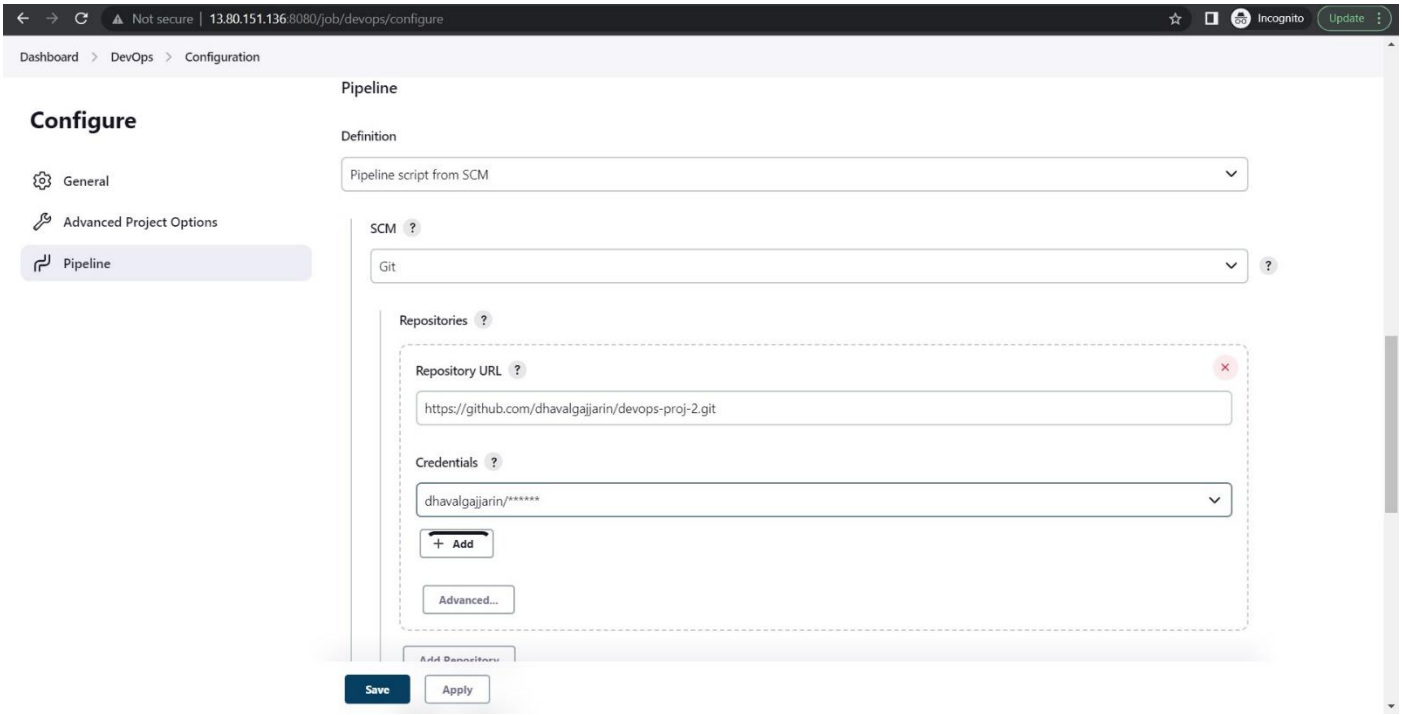


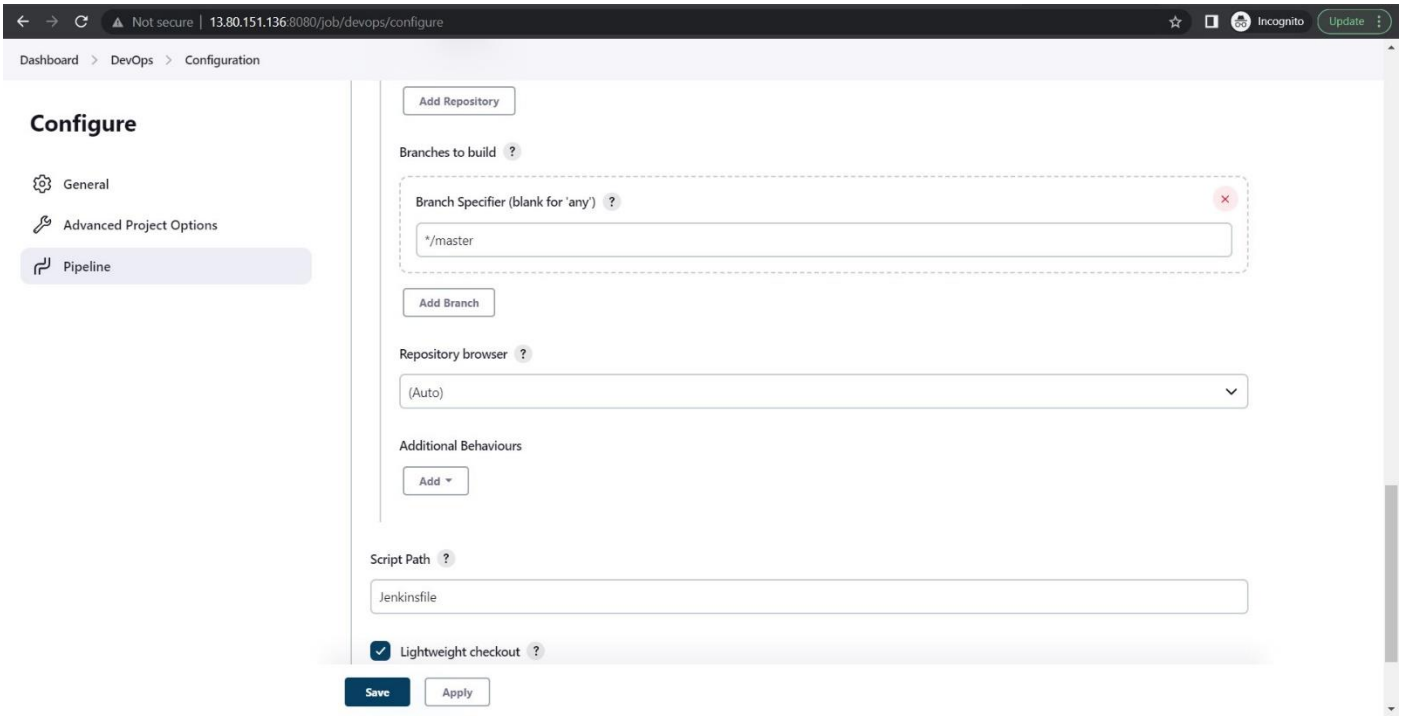
Create nodeport service on port 30008.

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: devops
  labels:
    app: devops
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30008
      name: http
  selector:
    app: devops
```

```
azureuser@kubernetes: ~
13 updates can be applied immediately.
13 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Dec 15 17:31:15 2022 from 49.206.36.205
azureuser@kubernetes:~$ kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
kubernetes   Ready    control-plane   2d18h   v1.26.0
worker1      Ready    <none>          2d18h   v1.26.0
worker2      Ready    <none>          2d18h   v1.26.0
azureuser@kubernetes:~$ envsubst
^C
azureuser@kubernetes:~$ vi nodeport.yaml
azureuser@kubernetes:~$ kubectl create -f nodeport.yaml
service/devops created
azureuser@kubernetes:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
devops       NodePort    10.97.148.98   <none>        80:30008/TCP   16s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        2d18h
azureuser@kubernetes:~$
```

Create Jenkins pipeline script job with Jenkinsfile.





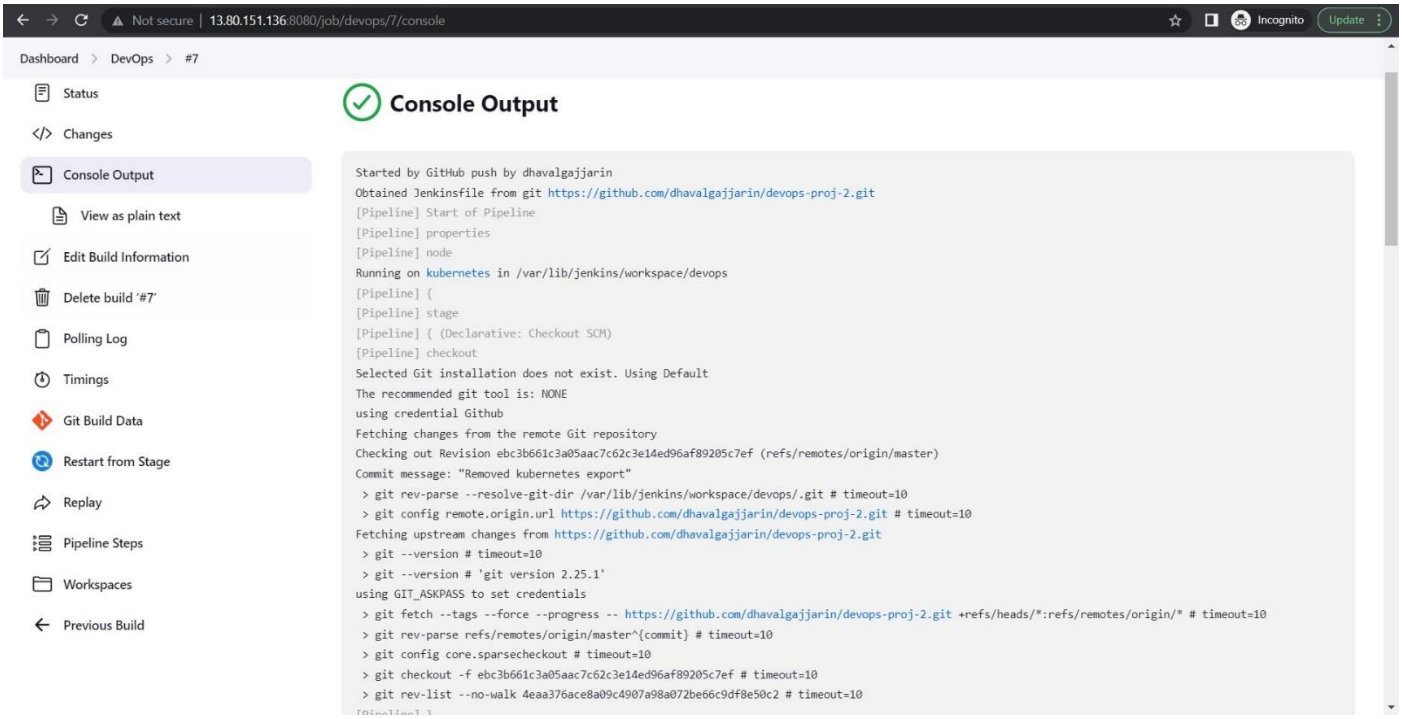

```
properties([pipelineTriggers([githubPush()])])

pipeline {
    agent {
        label 'kubernetes'
    }

    environment {
        DOCKERHUB_CREDENTIALS=credentials('Docker')
        REPOSITORY_TAG="dhavalgajjarin/devops:v${BUILD_NUMBER}"
    }

    stages {
        stage('Docker Build') {
            steps {
                sh 'docker build -t ${REPOSITORY_TAG} .'
            }
        }
        stage('Docker Login') {
            steps {
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
            }
        }
        stage('Docker Push') {
            steps {
                sh 'docker push ${REPOSITORY_TAG}'
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                sh 'envsubst < ${WORKSPACE}/deploy.yaml | kubectl apply -f -'
            }
        }
```

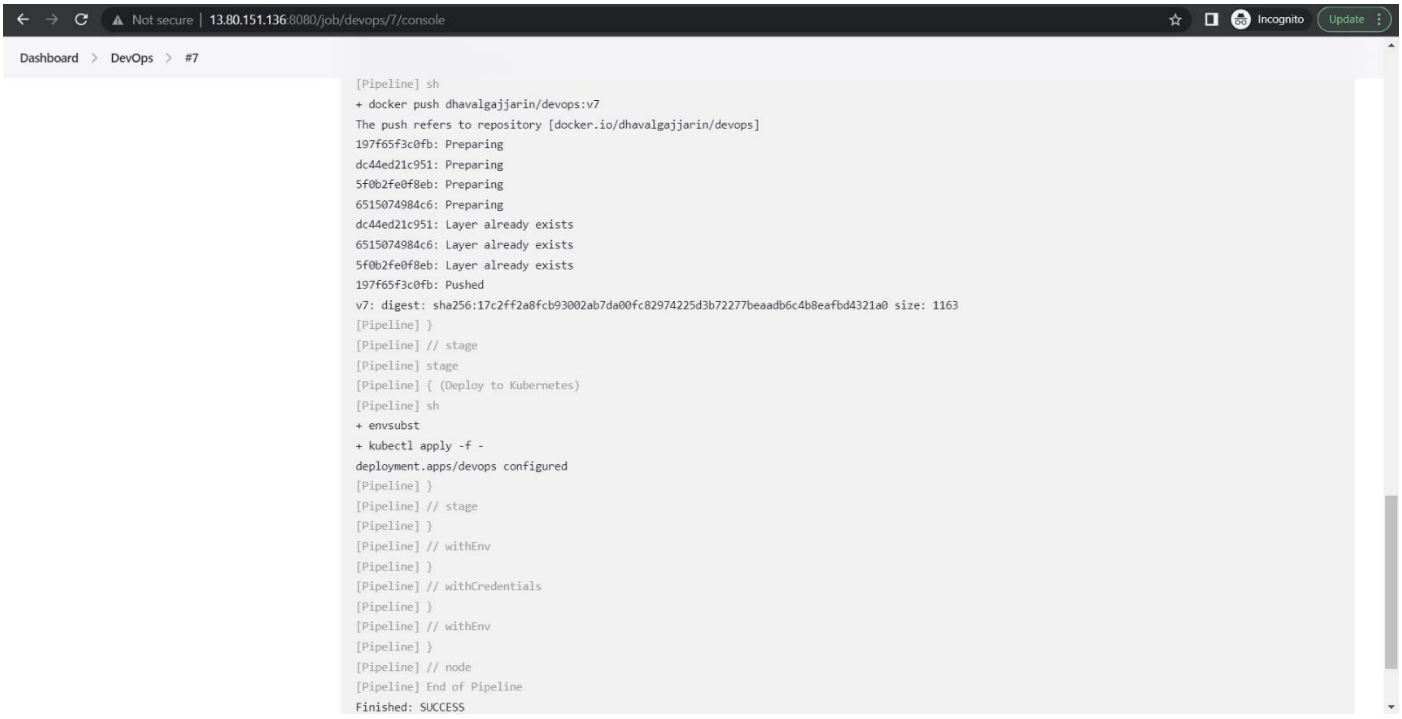
```
    }
}
```

Use deploy.yaml file to deploy latest release to Kubernetes

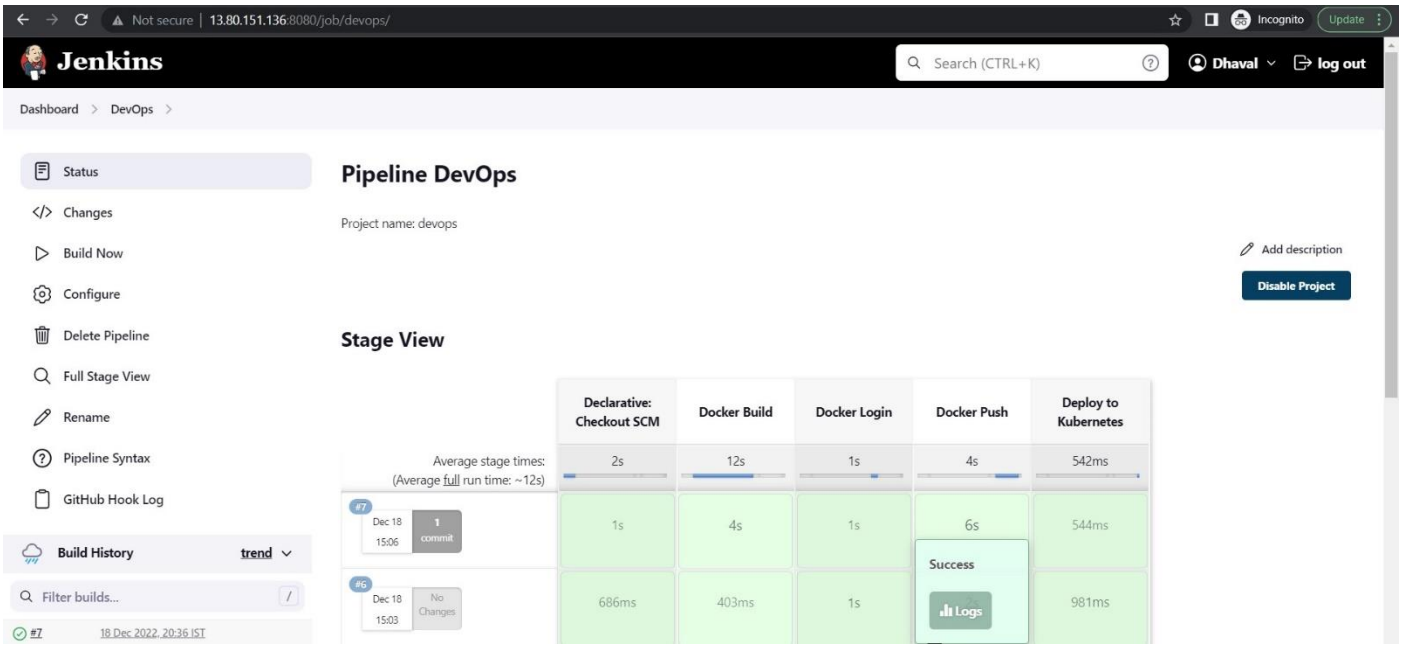
```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: devops
  labels:
    app: devops
spec:
  replicas: 2
  selector:
    matchLabels:
      app: devops
  template:
    metadata:
      labels:
        app: devops
    spec:
      containers:
        - name: devops
          image: $REPOSITORY_TAG
          ports:
            - containerPort: 80
```

On git commit Jenkins will start a new build and deploy.





You can see build steps in pipeline view.



Check the nodeport after deployment its working.