DevOps Project 1

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"
  }

}

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

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

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

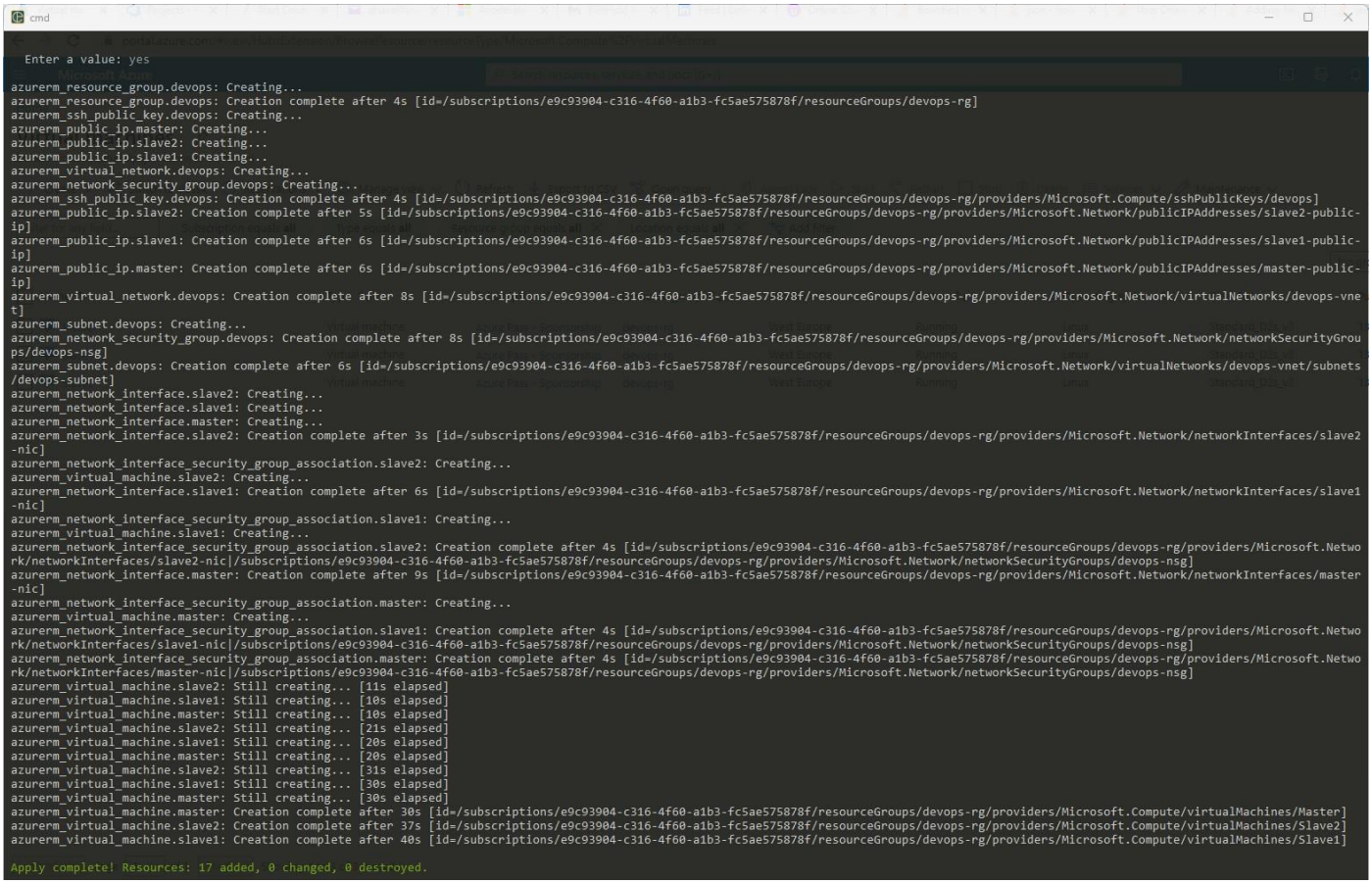
```
  }
}

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

resource "azurerm_virtual_machine" "master" {
  name                             = "Master"
  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.master.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              = "master-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "master"
    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" "slave1" {
  name                = "slave1-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}
```

```
resource "azurerm_network_interface" "slave1" {
  name                = "slave1-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "slave1-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.slave1.id
  }
}

resource "azurerm_network_interface_security_group_association" "slave1" {
  network_interface_id      = azurerm_network_interface.slave1.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "slave1" {
  name                             = "Slave1"
  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.slave1.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              = "slave1-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "slave1"
    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
    }
```

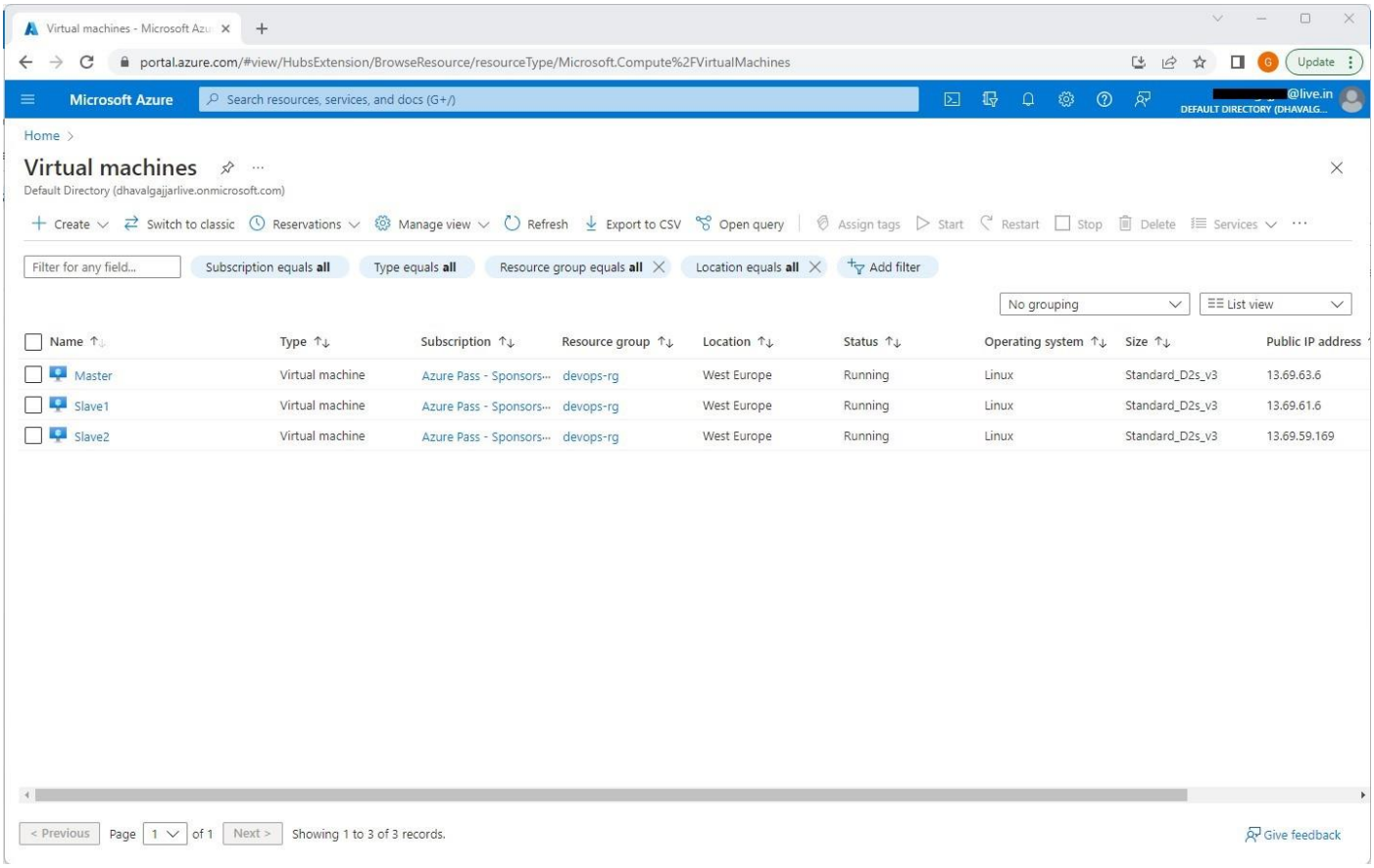
```
  }

}

// Slave 2
resource "azurerm_public_ip" "slave2" {
  name                = "slave2-public-ip"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name
  allocation_method   = "Dynamic"
}

resource "azurerm_network_interface" "slave2" {
  name                = "slave2-nic"
  location            = azurerm_resource_group.devops.location
  resource_group_name = azurerm_resource_group.devops.name

  ip_configuration {
    name                          = "slave2-ip"
    subnet_id                     = azurerm_subnet.devops.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.slave2.id
  }
}

resource "azurerm_network_interface_security_group_association" "slave2" {
  network_interface_id      = azurerm_network_interface.slave2.id
  network_security_group_id = azurerm_network_security_group.devops.id
}

resource "azurerm_virtual_machine" "slave2" {
  name                             = "Slave2"
  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.slave2.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              = "slave2-disk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
```

```
    computer_name  = "slave2"
    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.



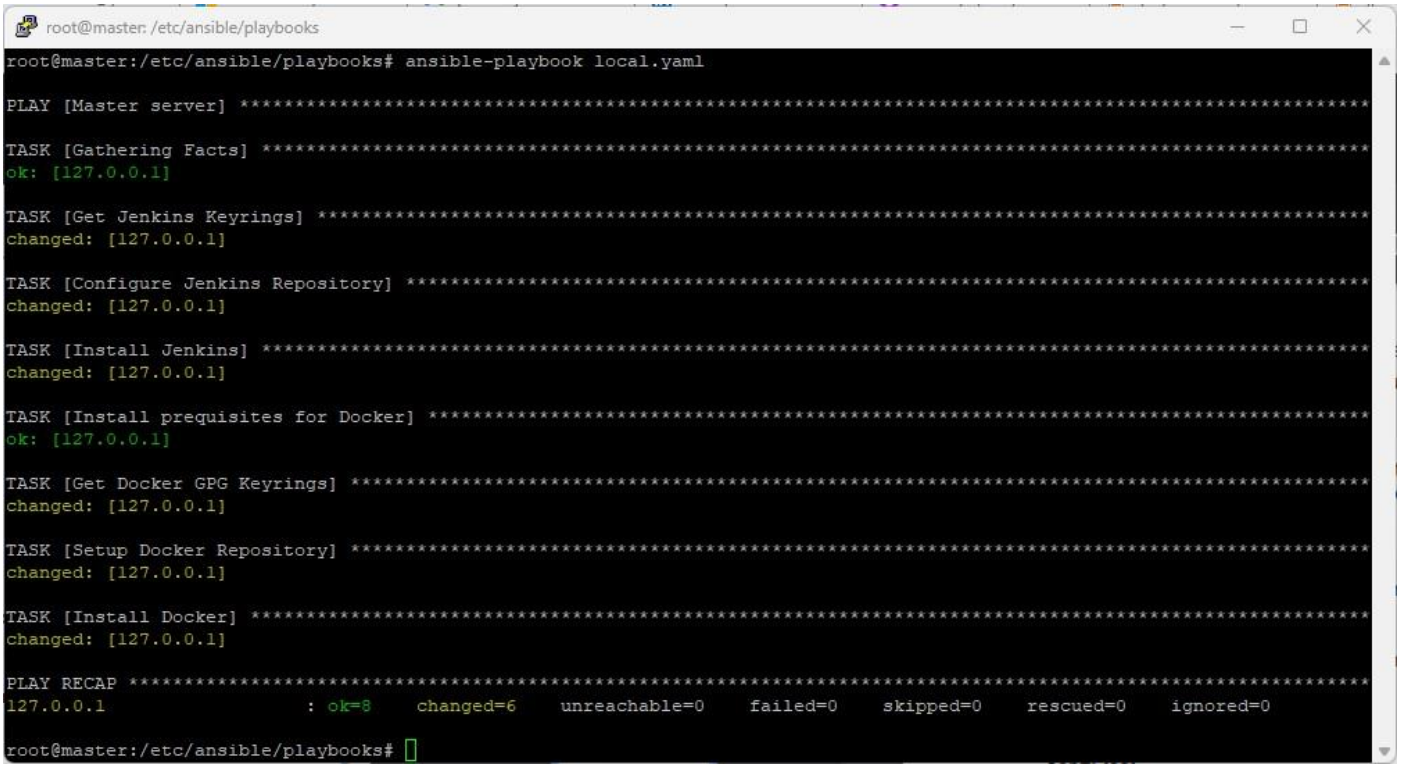
Install ansible.

$ 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 master machine softwares.

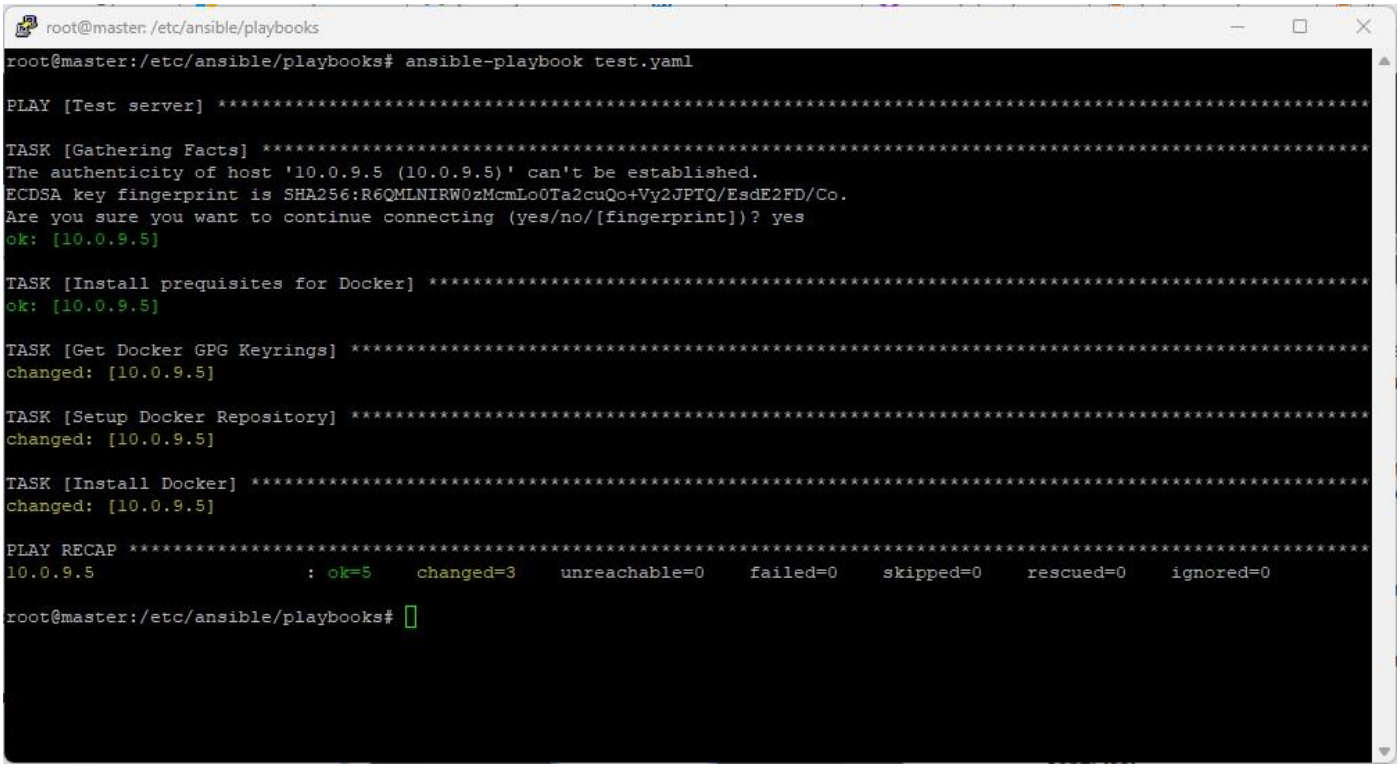
```
---
- name: Master server
  hosts: local
  remote_user: root
  tasks:
    - name: Install JDK 11
      apt:
        name: openjdk-11-jdk
        update_cache: true
    - name: Get Jenkins Keyrings
      get_url:
        url: https://pkg.jenkins.io/debian-stable/jenkins.io.key
        dest: /usr/share/keyrings/jenkins-keyring.asc
    - 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
    - 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
```


root@master: /etc/ansible/playbooks
root@master:/etc/ansible/playbooks# ansible-playbook local.yaml
PLAY [Master server] ***
TASK [Gathering Facts] ***
ok: [127.0.0.1]
TASK [Get Jenkins Keyrings] **
changed: [127.0.0.1]
TASK [Configure Jenkins Repository] **
changed: [127.0.0.1]
TASK [Install Jenkins] ***
changed: [127.0.0.1]
TASK [Install prequisites for Docker] **
ok: [127.0.0.1]
TASK [Get Docker GPG Keyrings] ***
changed: [127.0.0.1]
TASK [Setup Docker Repository] ***
changed: [127.0.0.1]
TASK [Install Docker] **
changed: [127.0.0.1]
PLAY RECAP ***
127.0.0.1 : ok=8 changed=6 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
root@master:/etc/ansible/playbooks#

Using ansible-playbook setup test machine softwares.

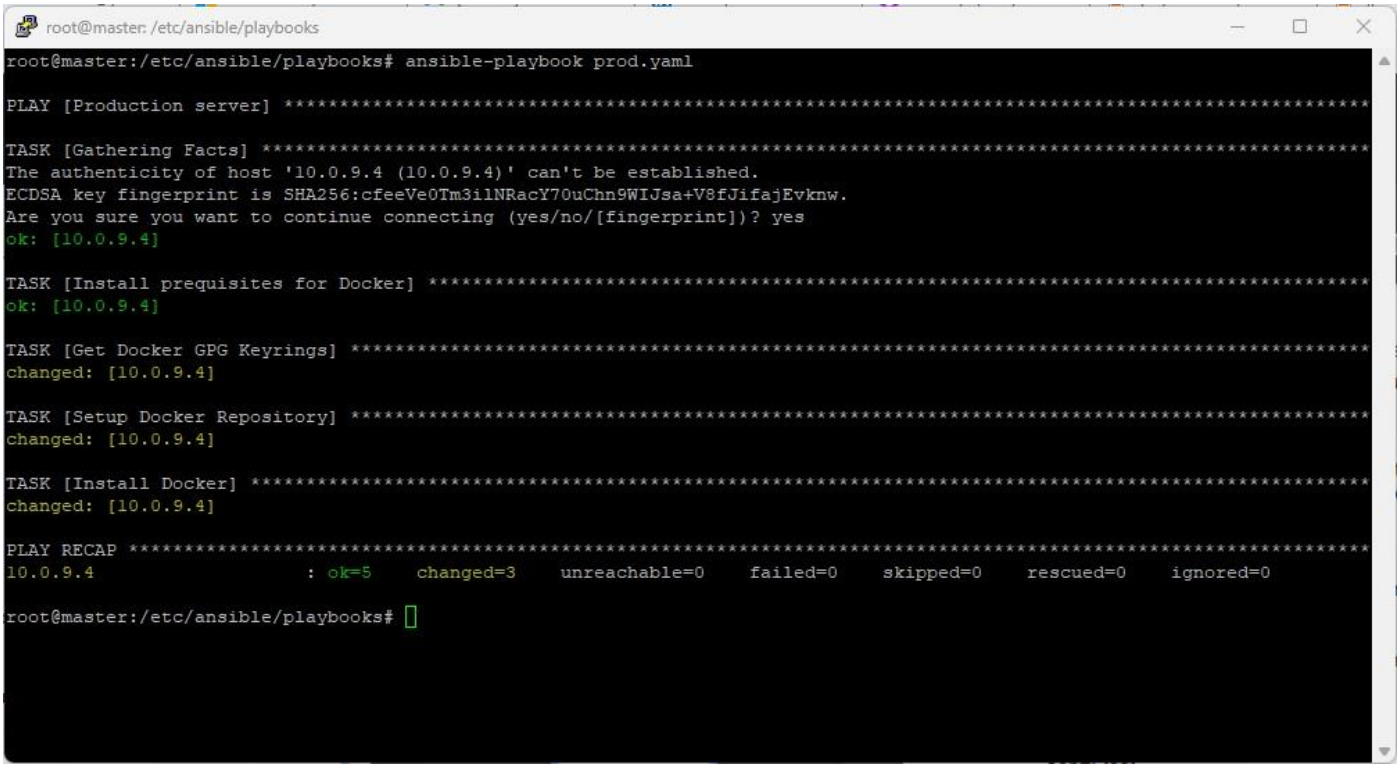
```
---
- name: Test server
  hosts: test
  remote_user: root
  tasks:
    - name: Install prequisites for Docker
      apt:
        pkg:
          - ca-certificates
          - curl
          - gnupg
          - lsb-release
    - name: Get Docker GPG Keyrings
      shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | 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 JDK 11
      apt:
        name: openjdk-11-jdk
```



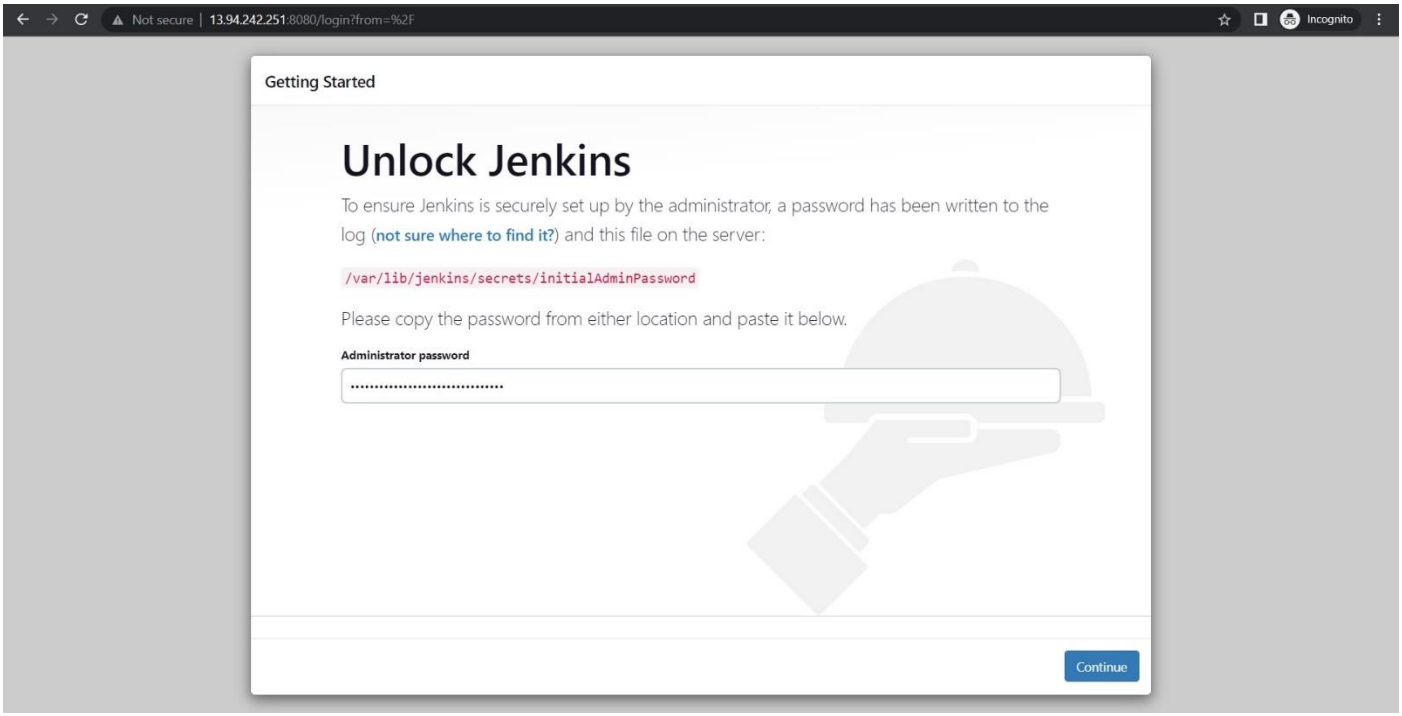
Using ansible-playbook setup production machine softwares.

```
---
- name: Production server
  hosts: prod
  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
```

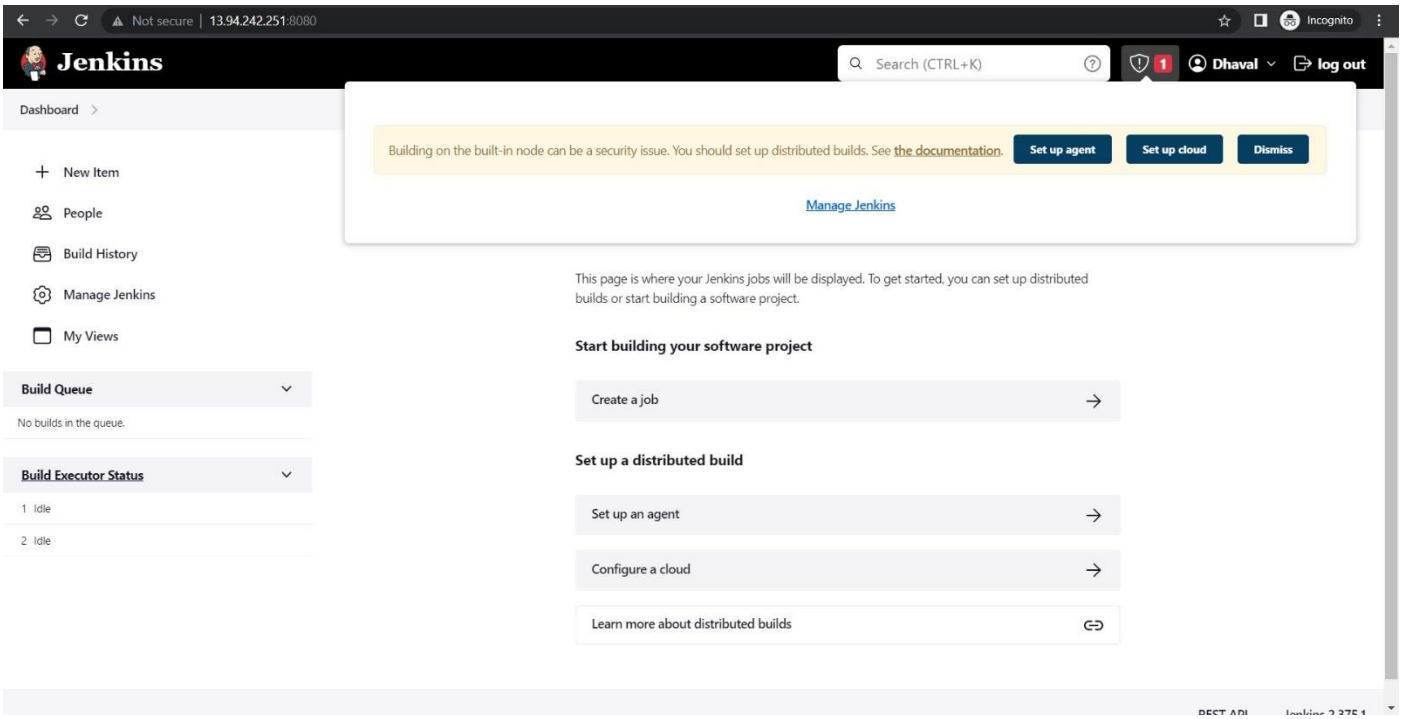
```
        masked: no
        daemon_reload: yes
    - name: Install JDK 11
      apt:
        name: openjdk-11-jdk
```



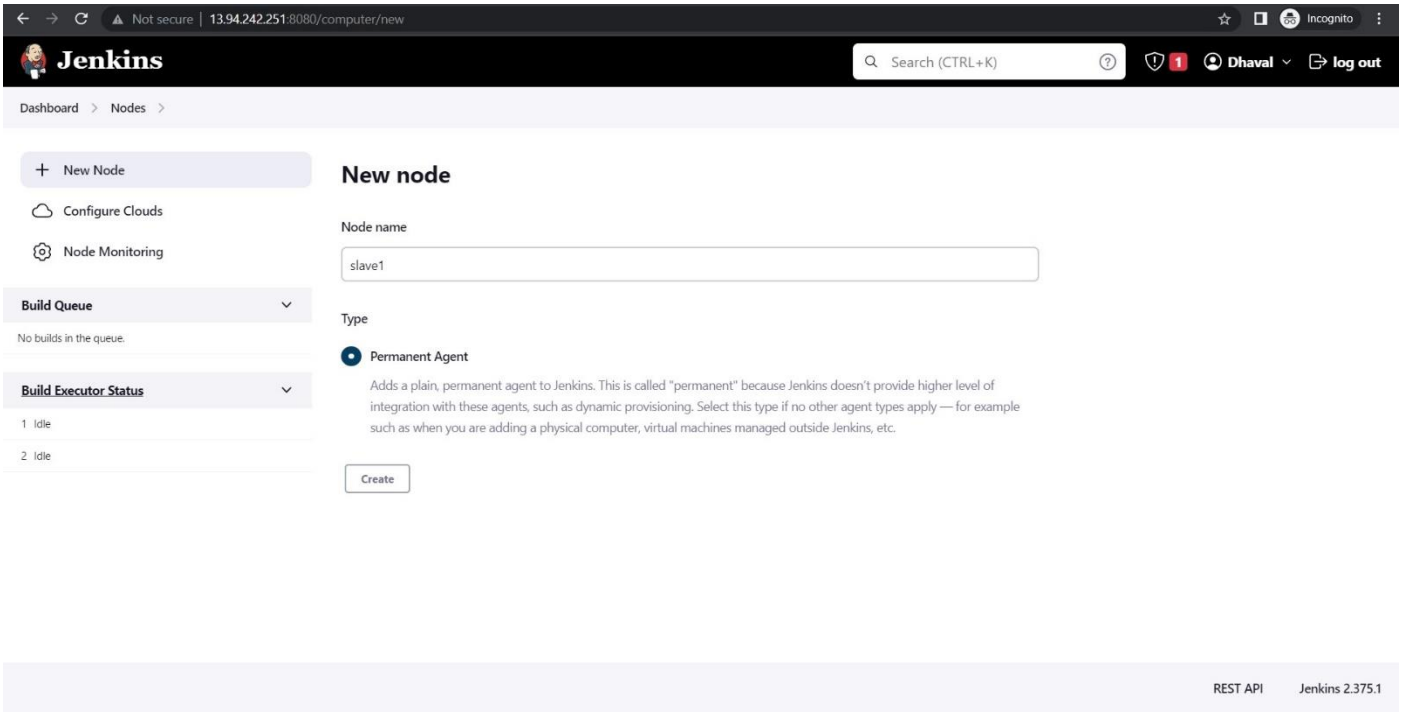
Setup the Jenkins.

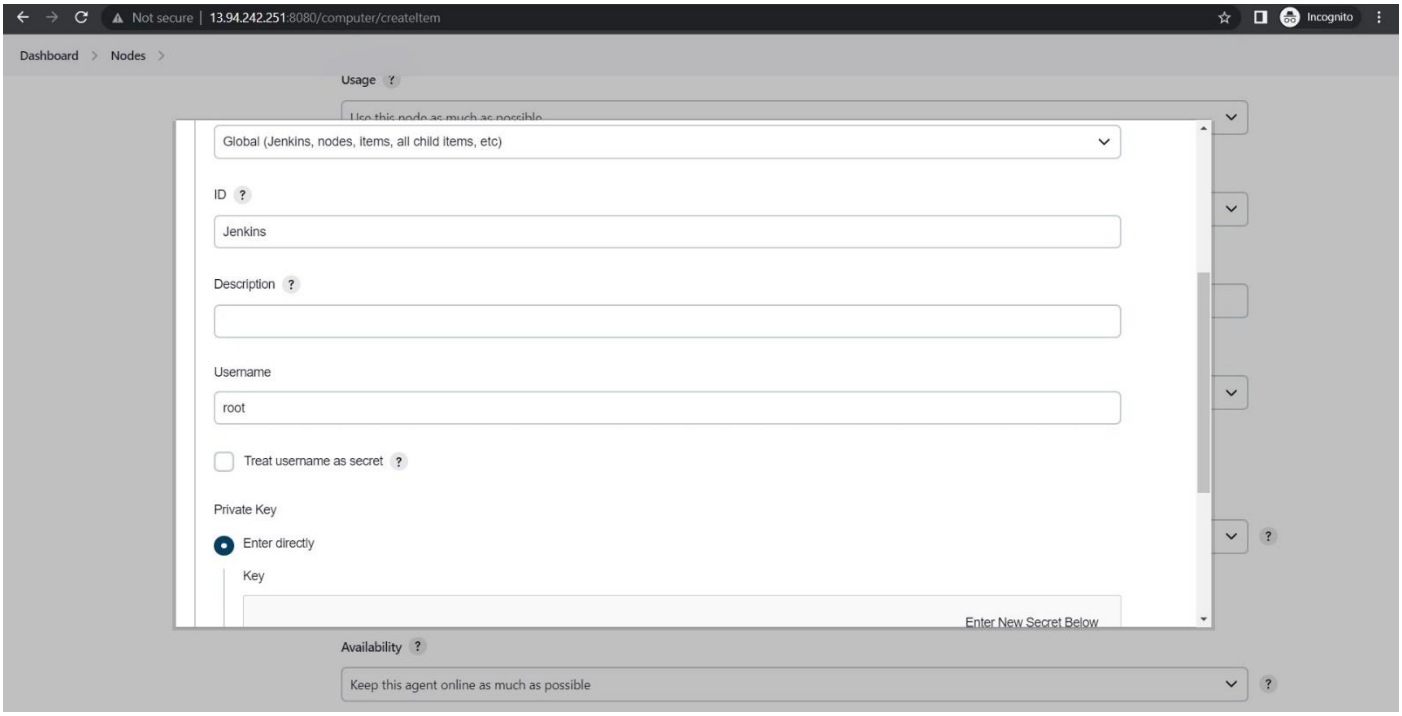


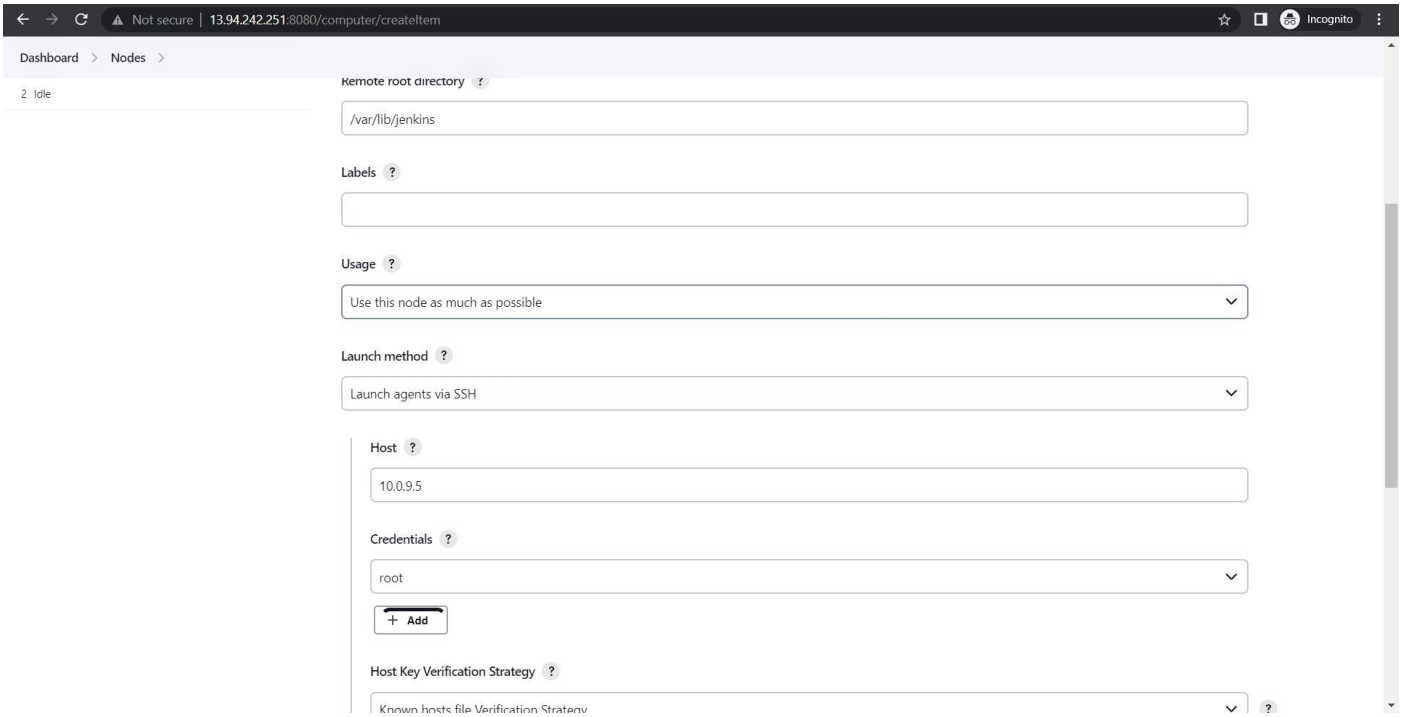
Login to Jenkins.



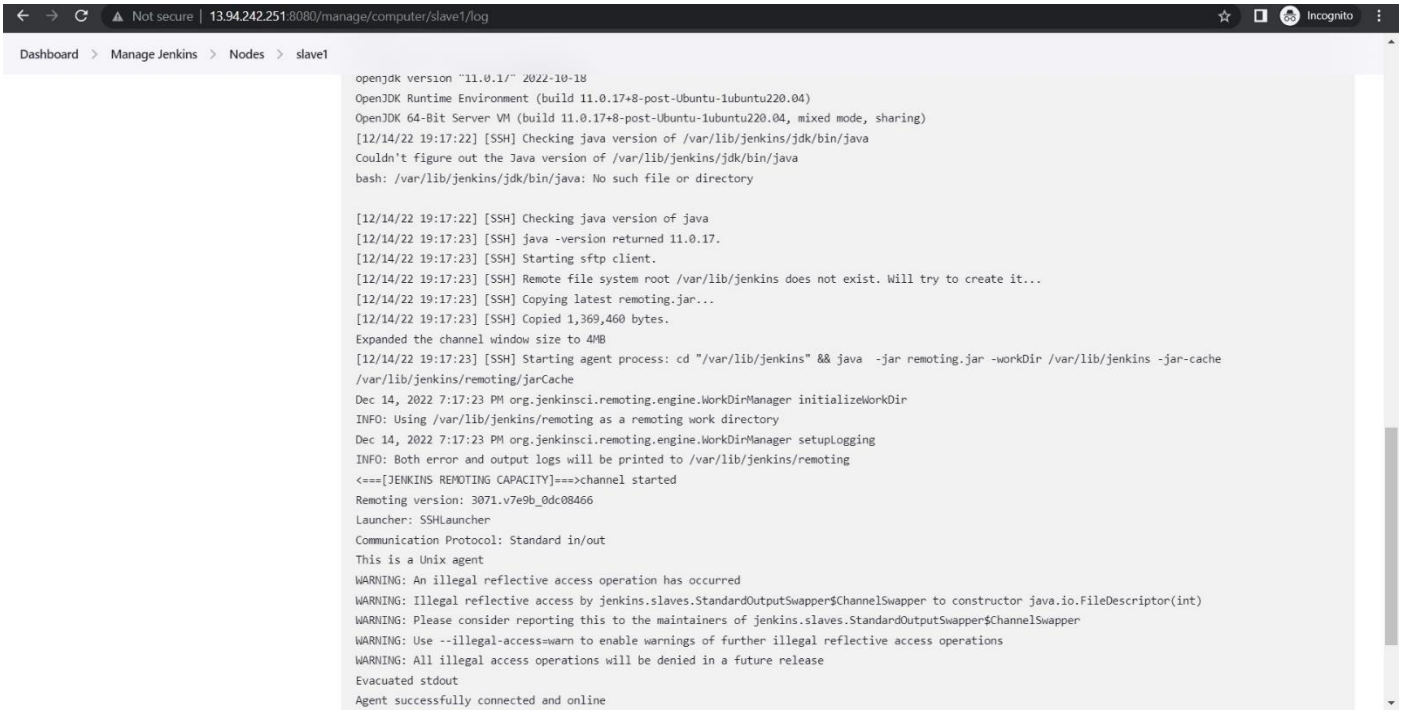
Setup new Jenkins node for test environment.



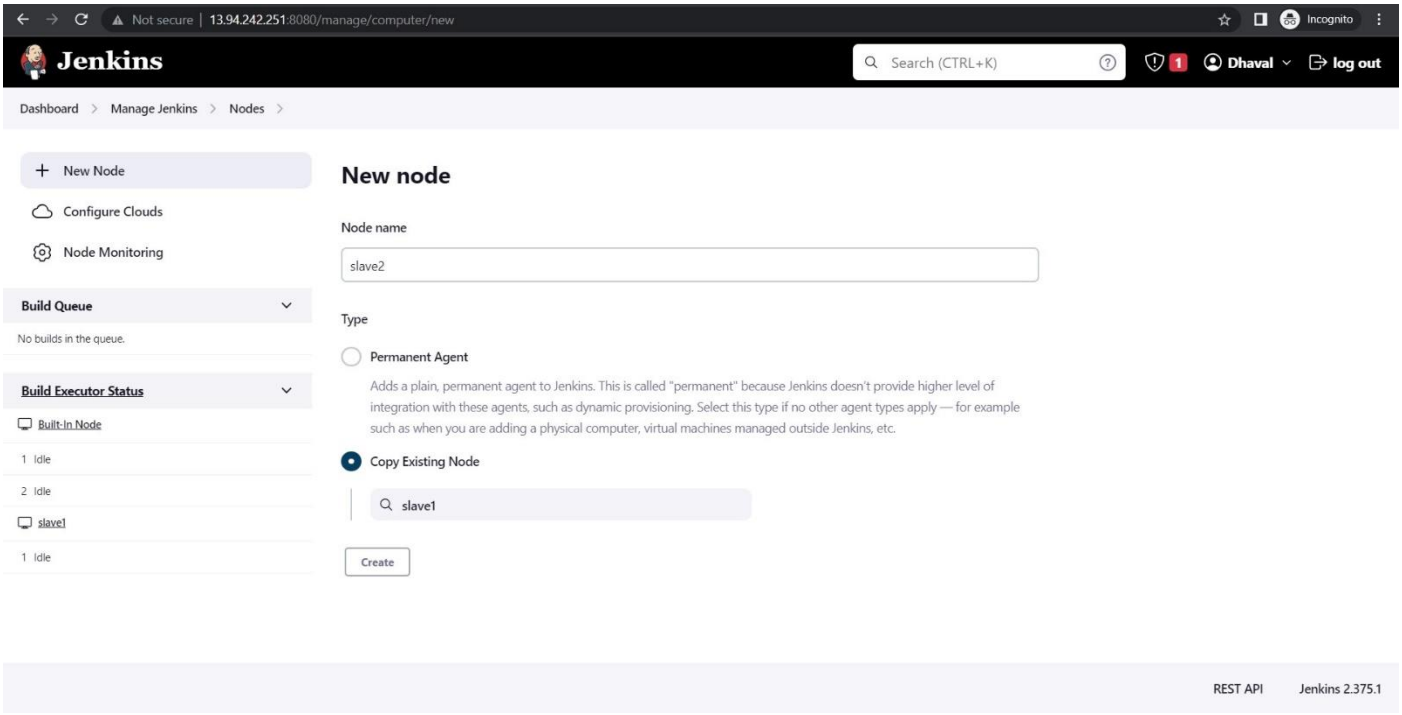


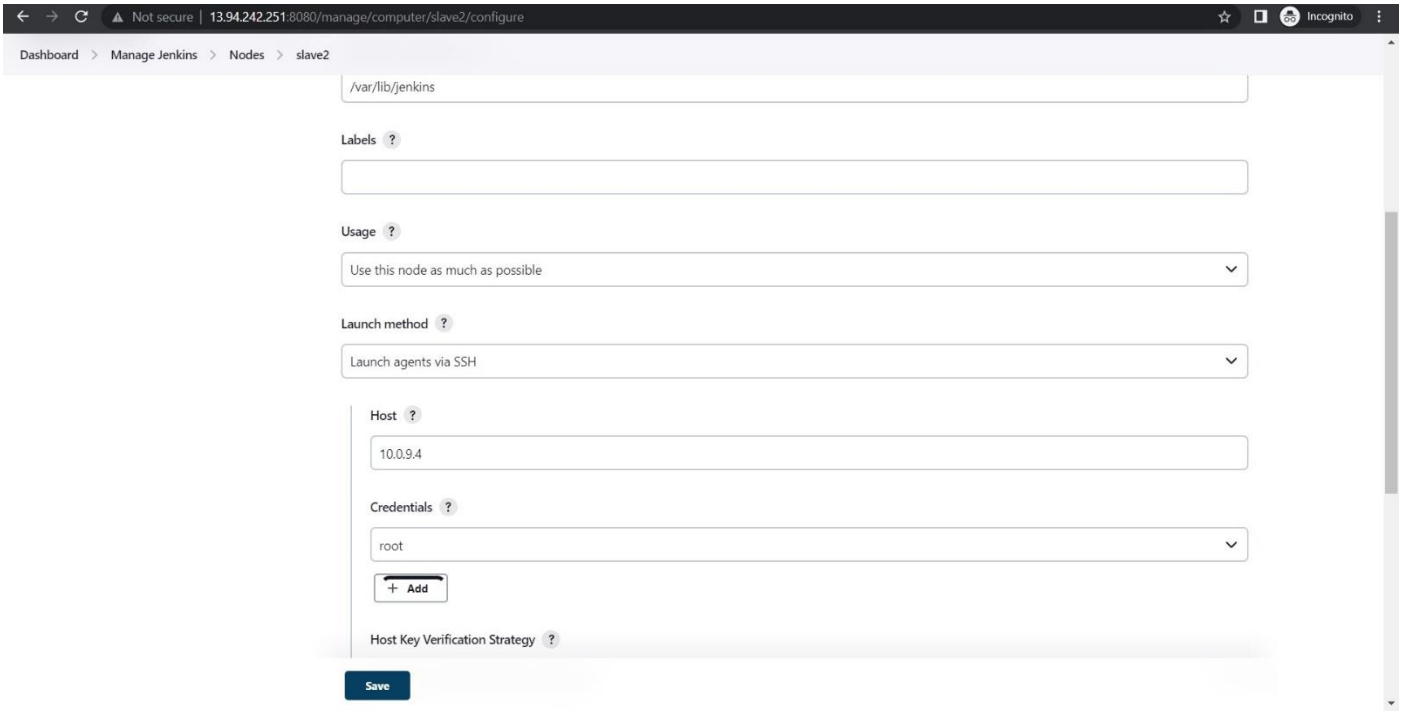


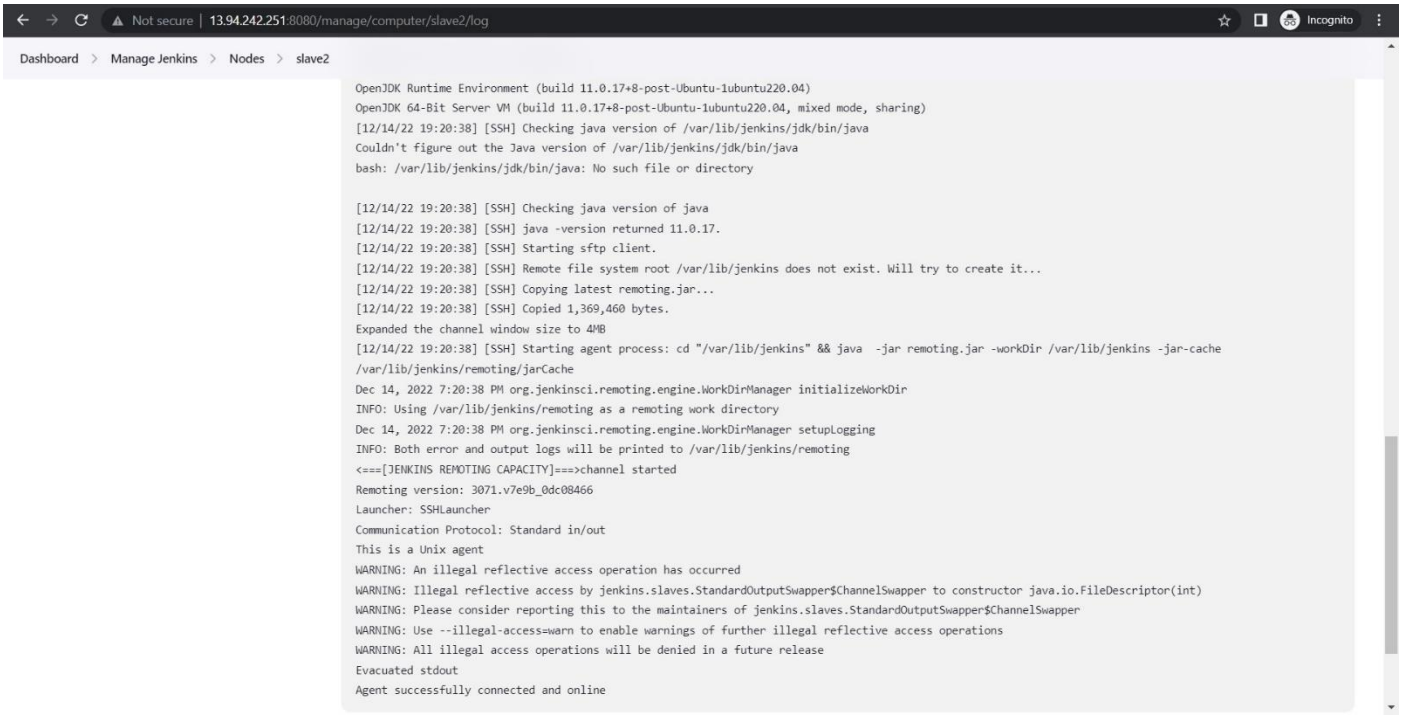
Make sure node is up and online.



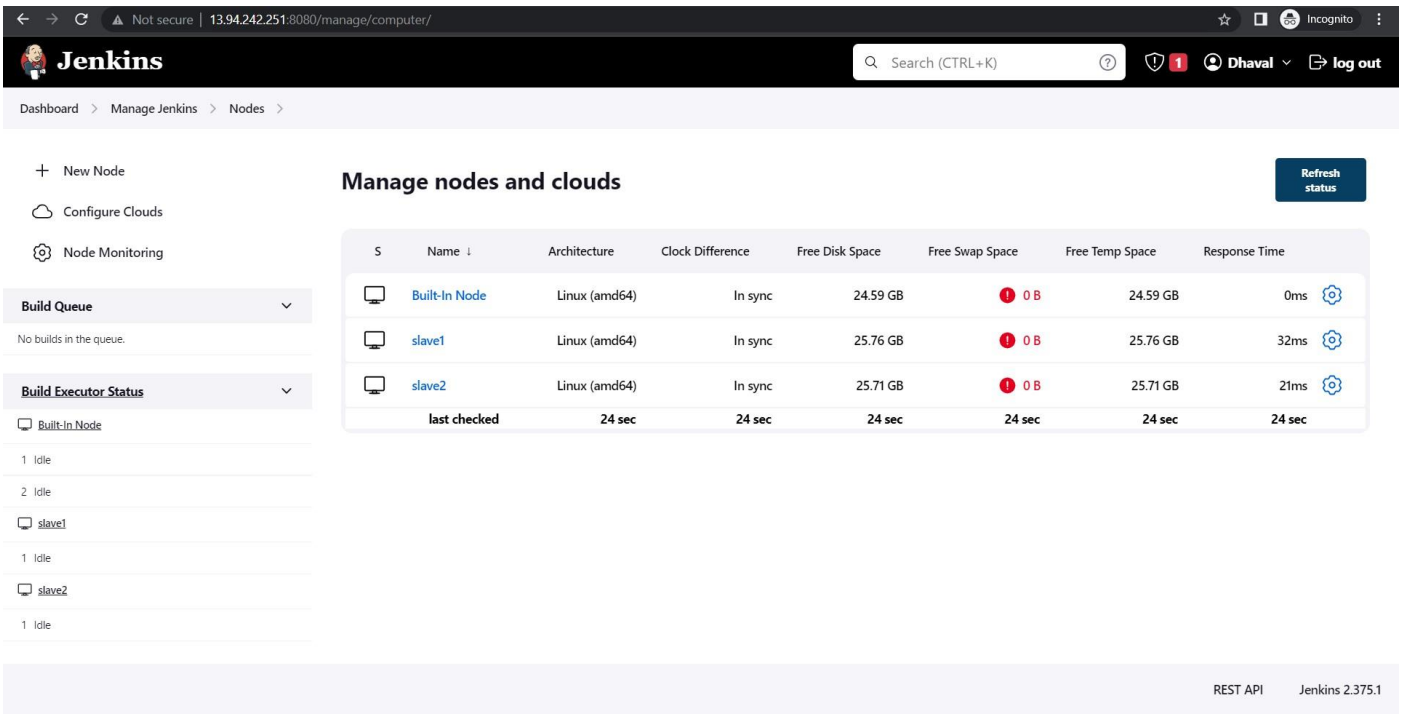
Setup new Jenkins node for production environment.



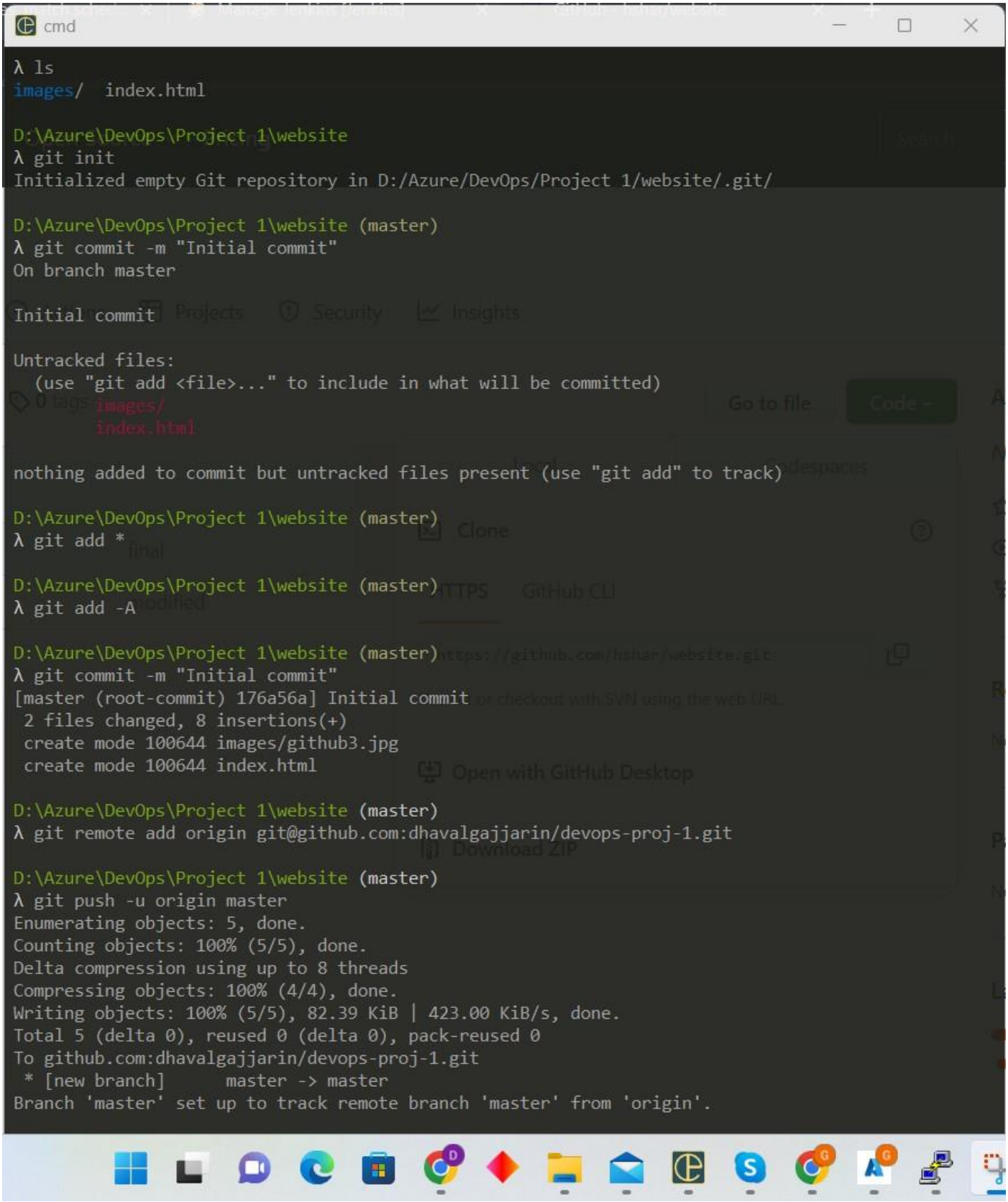




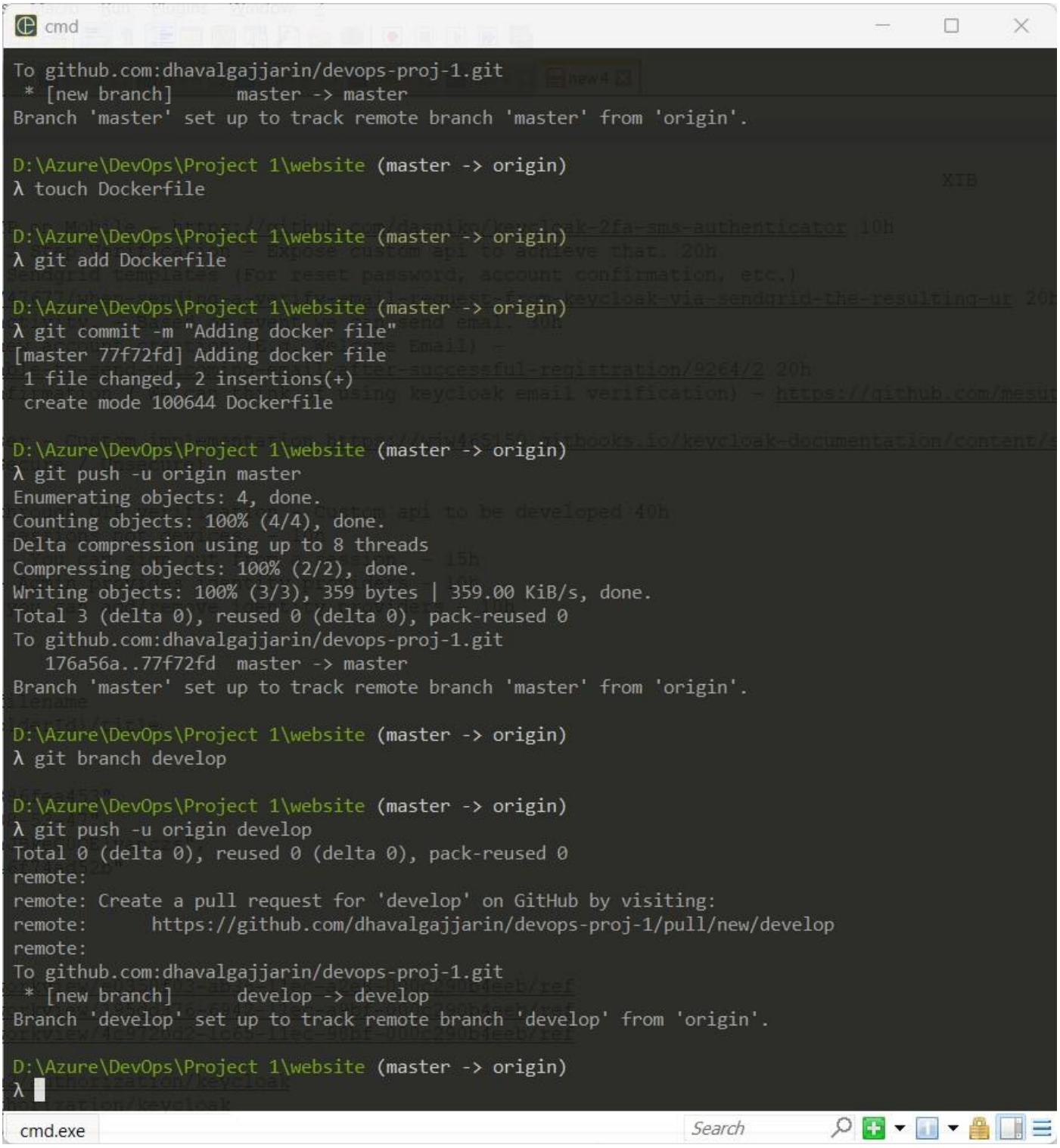
Make sure nodes are up and online.



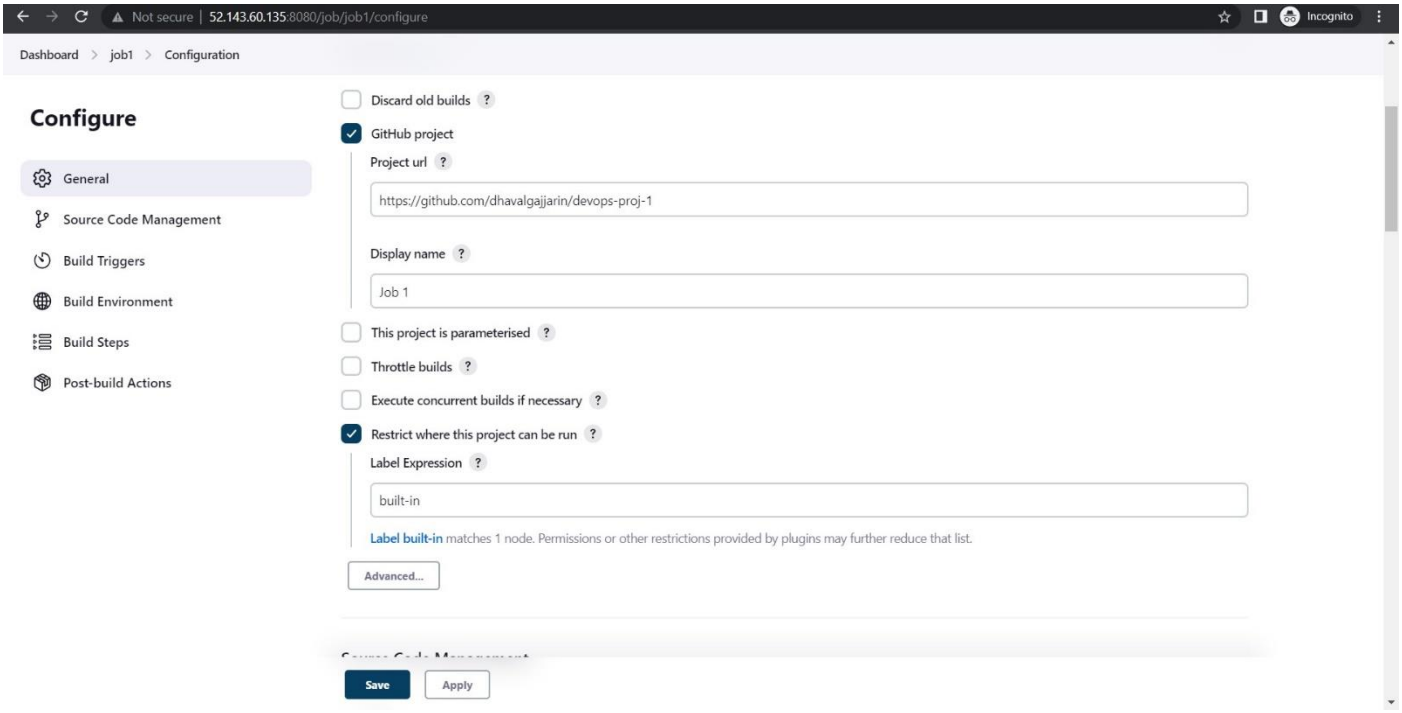
Setup the initial commit to GitHub repository in the master branch.

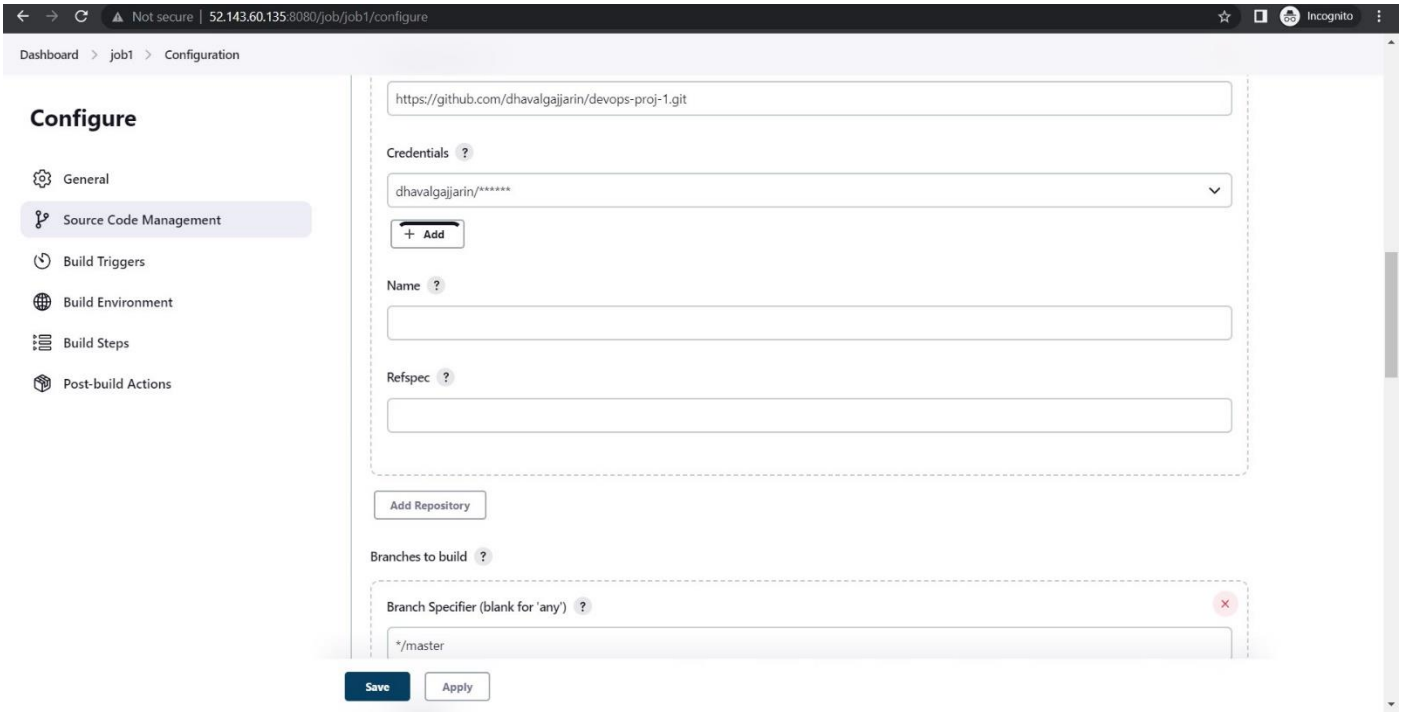


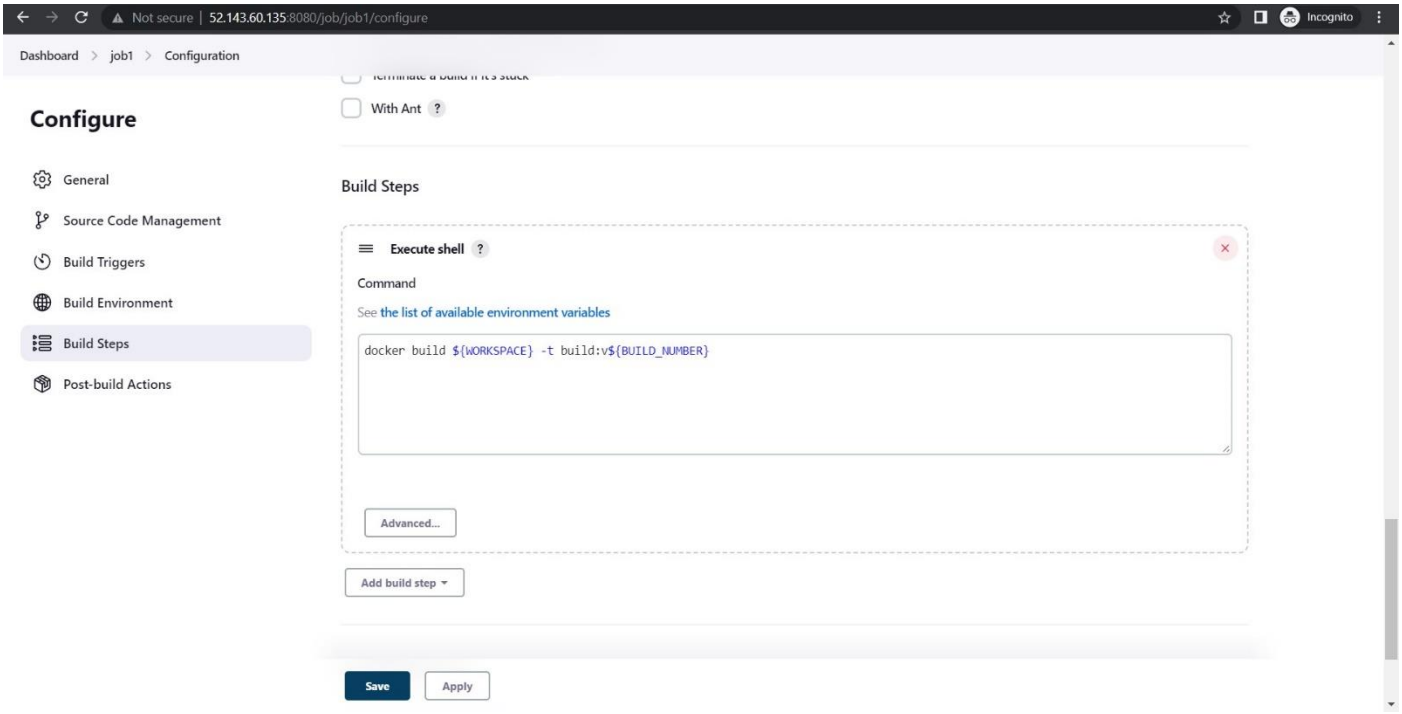
Create GitHub workflow with creating develop branch.

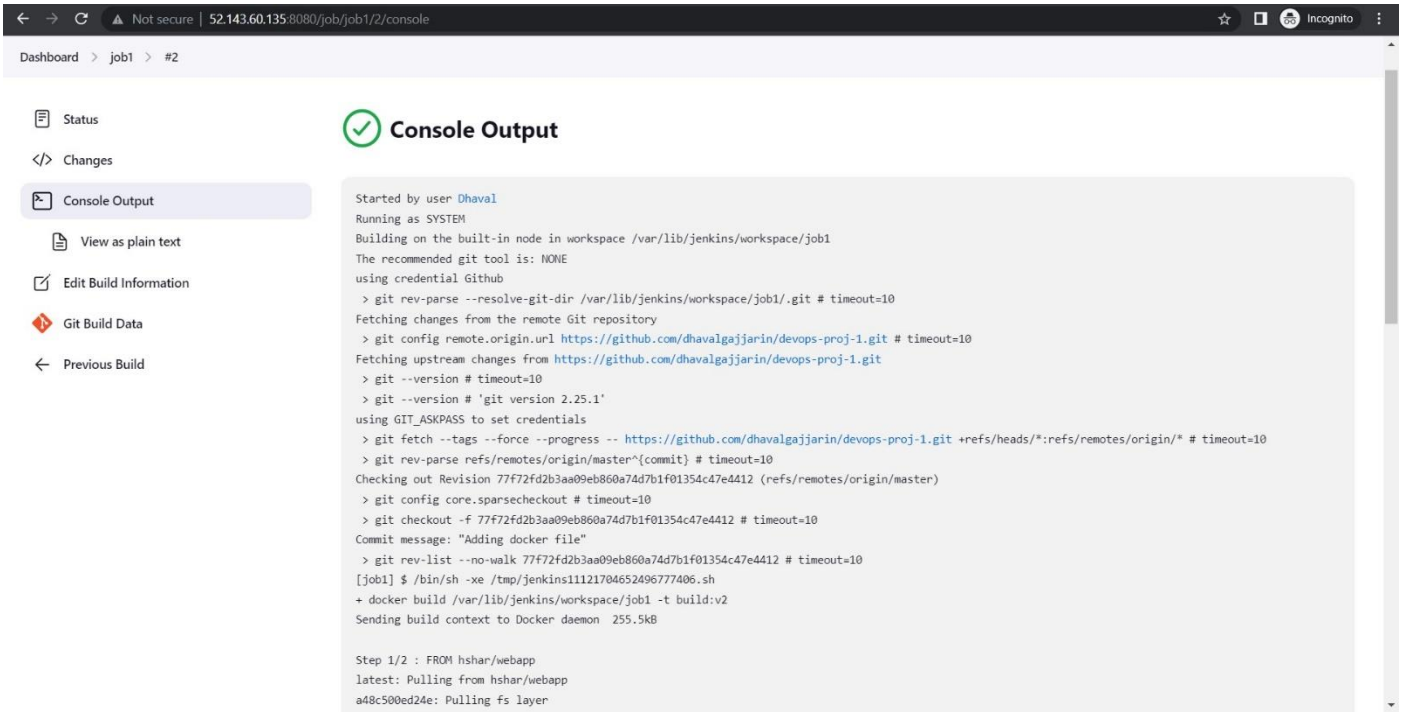


Create Jenkins Job1 which can checkout and build.

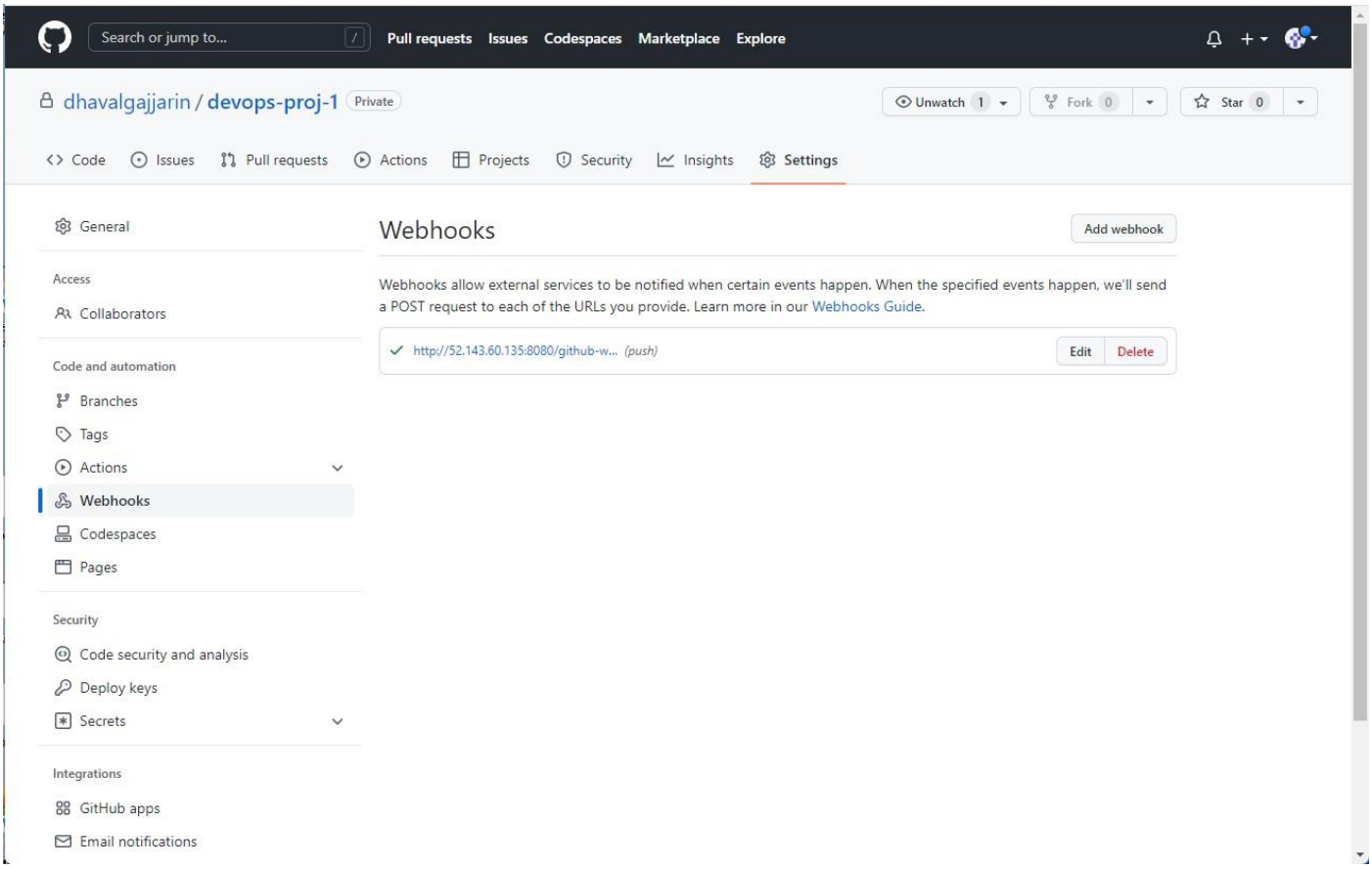




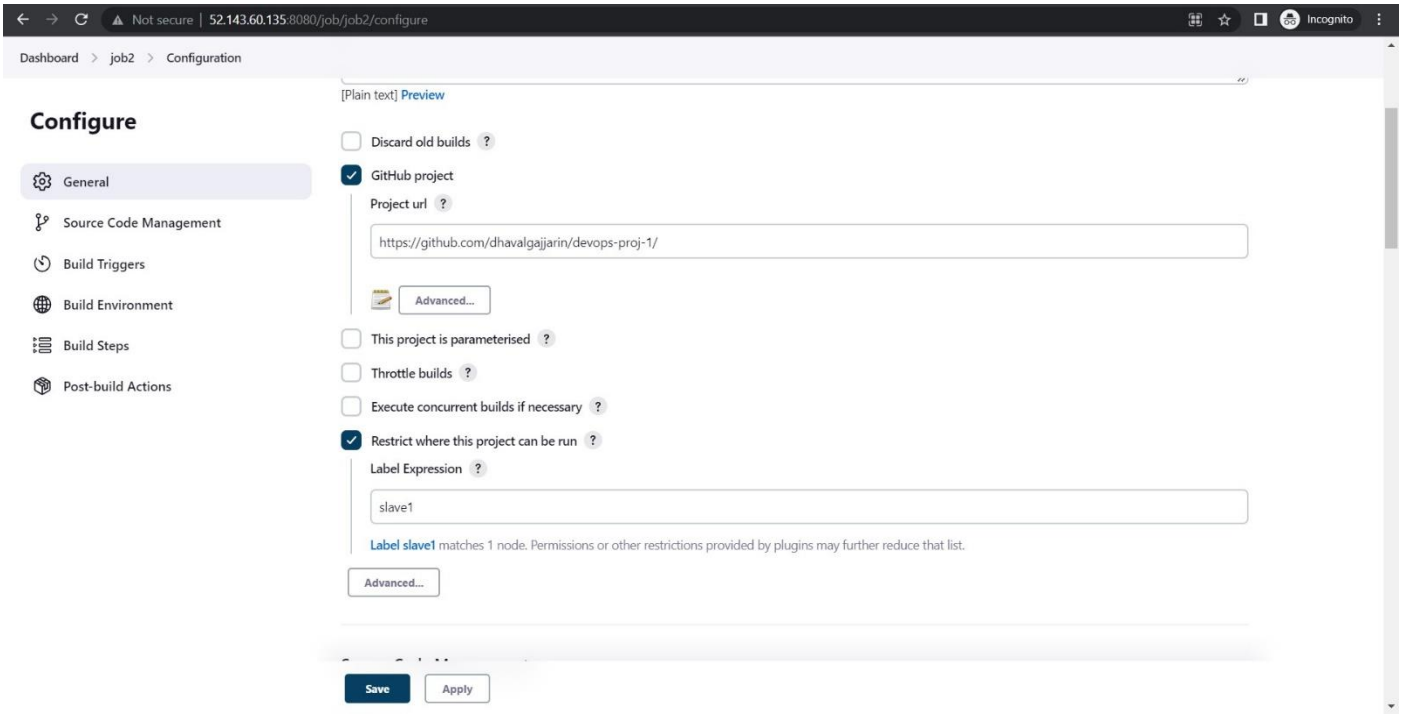

Not secure | 52.143.60.135:8080/job/job1/configure
Incognito
Dashboard > job1 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
docker build ${WORKSPACE} -t build:v${BUILD_NUMBER}
Advanced...
Add build step
Save
Apply

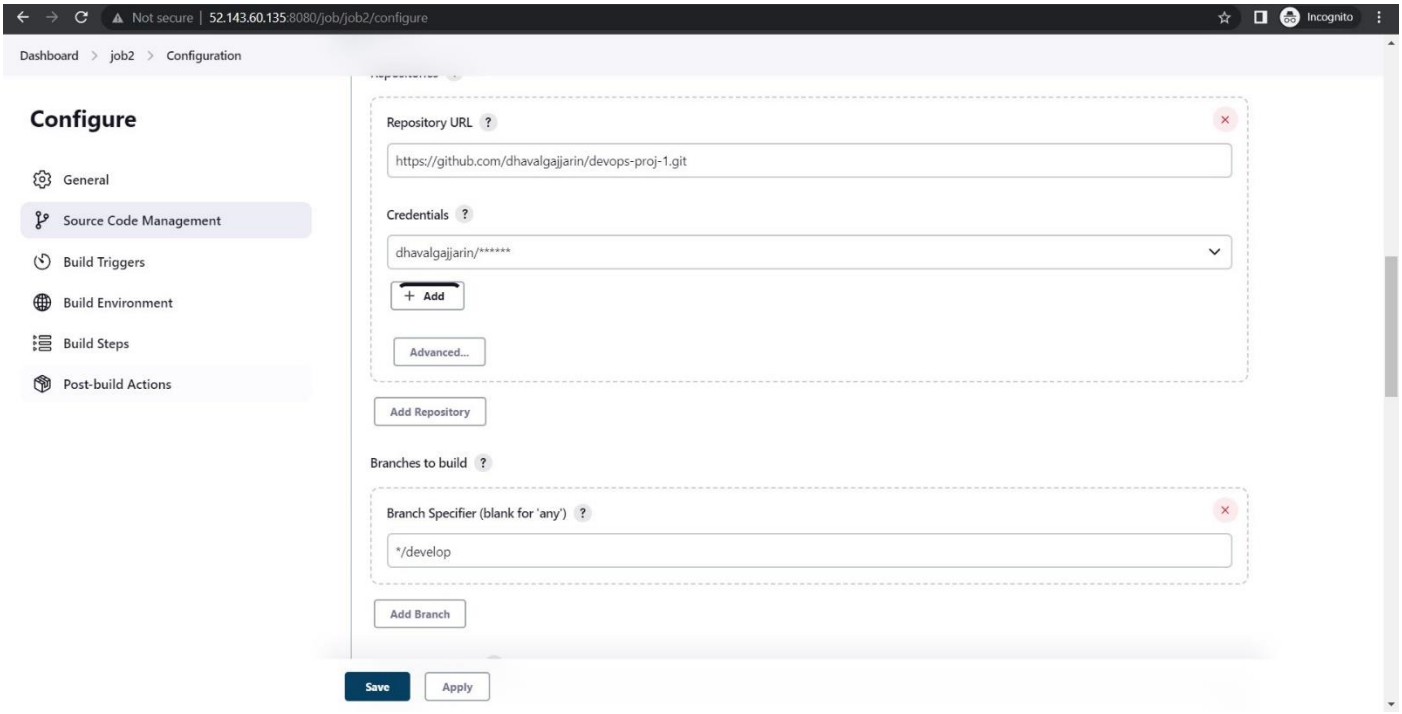

Not secure | 52.143.60.135:8080/job/job1/2/console
Incognito
Dashboard > job1 > #2
Status
Changes
Console Output
View as plain text
Edit Build Information
Git Build Data
Previous Build
Console Output
Started by user Dhaval
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/job1
The recommended git tool is: NONE
using credential Github
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/job1/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/dhavalgajjarin/devops-proj-1.git # timeout=10
Fetching upstream changes from https://github.com/dhavalgajjarin/devops-proj-1.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/dhavalgajjarin/devops-proj-1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 77f72fd2b3aa09eb860a74d7b1f01354c47e4412 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 77f72fd2b3aa09eb860a74d7b1f01354c47e4412 # timeout=10
Commit message: "Adding docker file"
> git rev-list --no-walk 77f72fd2b3aa09eb860a74d7b1f01354c47e4412 # timeout=10
[job1] $ /bin/sh -xe /tmp/jenkins11121704652496777406.sh
+ docker build /var/lib/jenkins/workspace/job1 -t build:v2
Sending build context to Docker daemon 255.5kB
Step 1/2 : FROM hshar/webapp
latest: Pulling from hshar/webapp
a48c500ed24e: Pulling fs layer

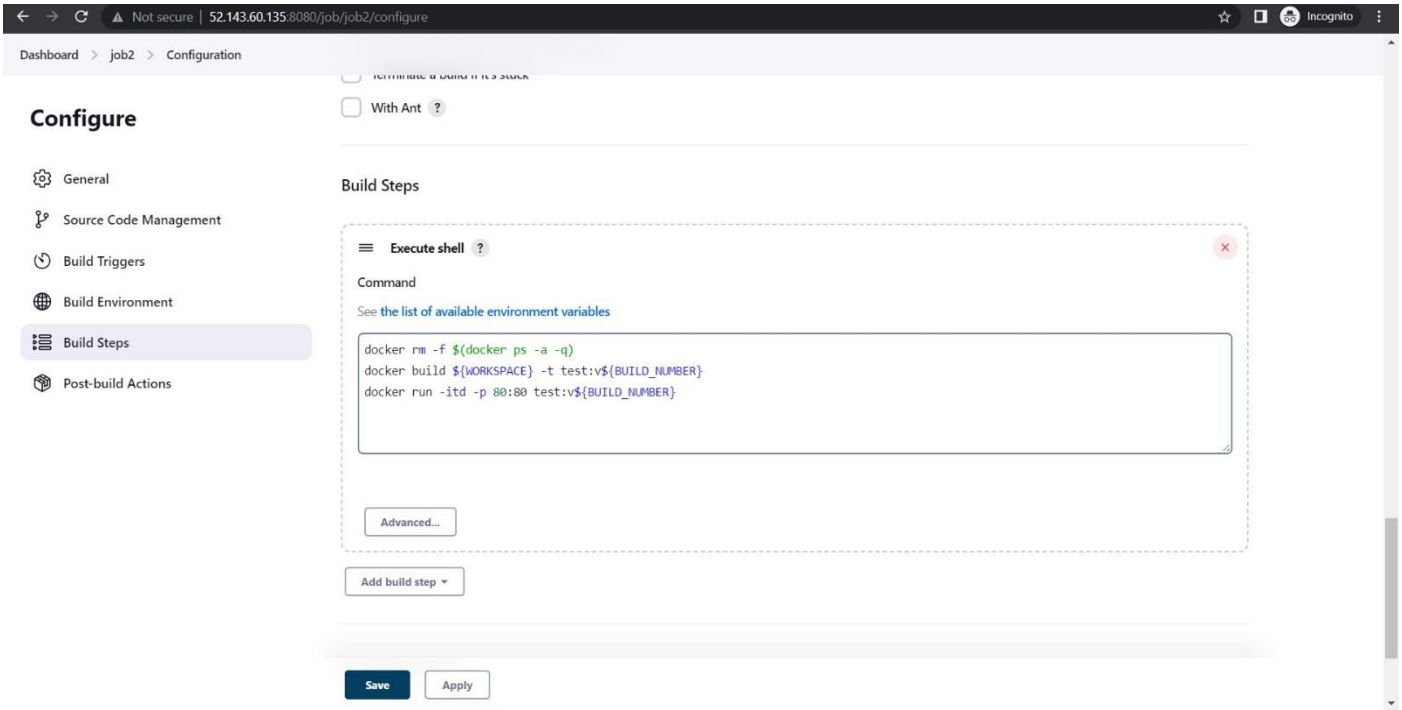
Configure webhook in GitHub so commit will trigger build.



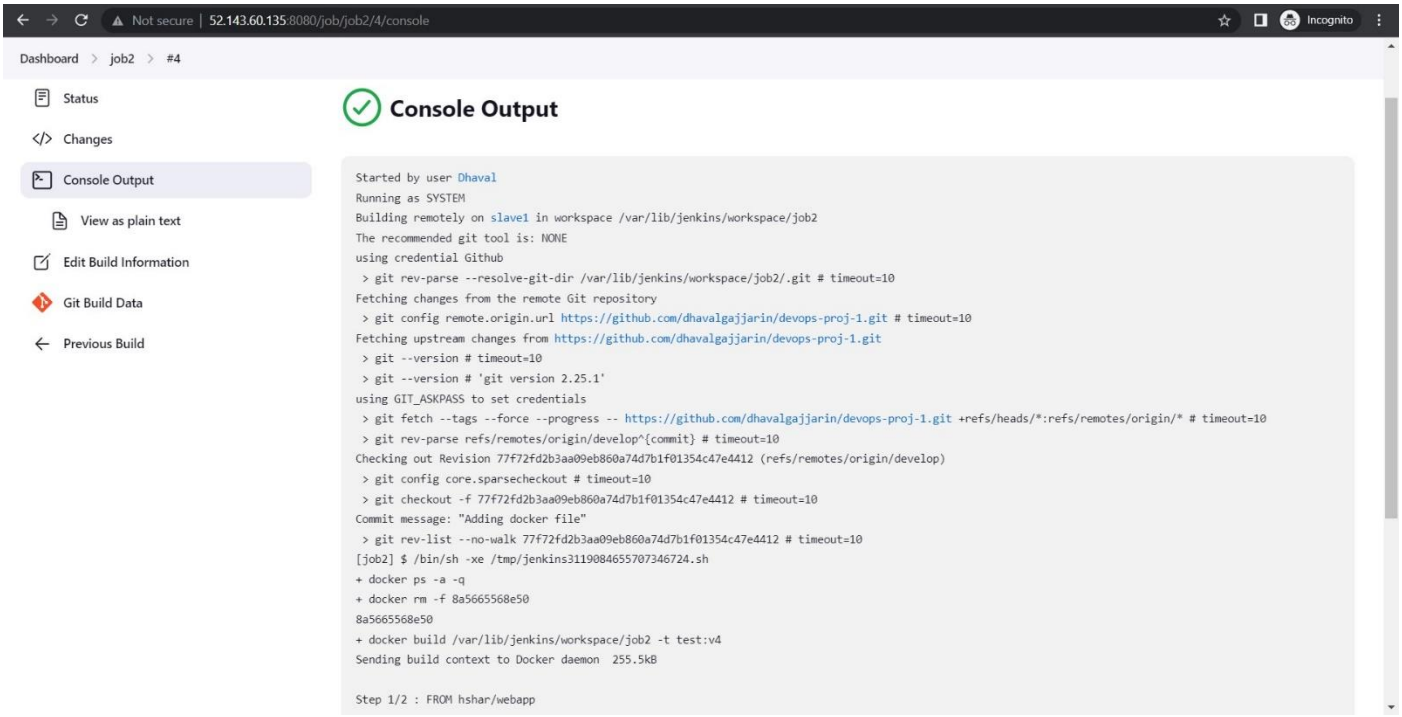
Create Jenkins Job2 which can checkout and build and deploy to the test environment.



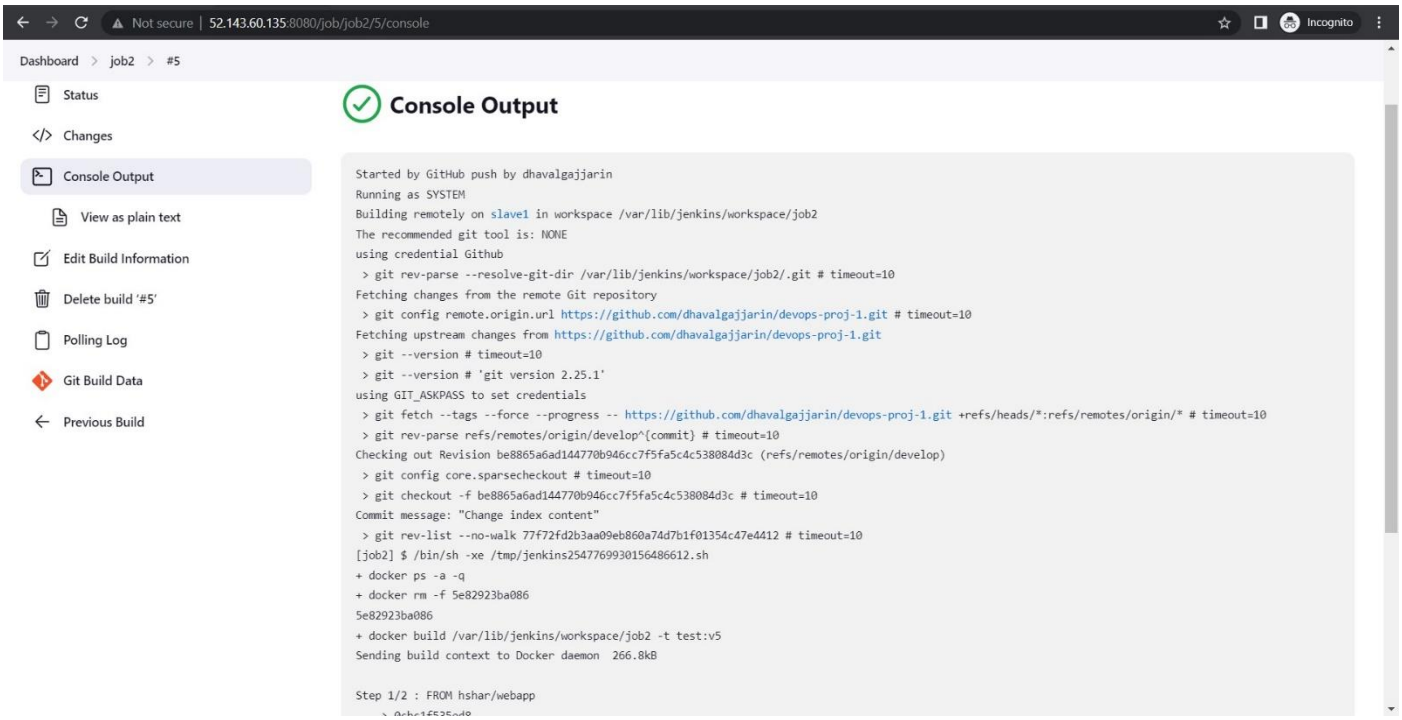

Not secure | 52.143.60.135:8080/job/job2/configure
Incognito
Dashboard > job2 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Repository URL
https://github.com/dhavalgajjarin/devops-proj-1.git
Credentials
dhavalgajjarin/******
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/develop
Add Branch
Save
Apply


Not secure | 52.143.60.135:8080/job/job2/configure
Incognito
Dashboard > job2 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
docker rm -f $(docker ps -a -q)
docker build ${WORKSPACE} -t test:v${BUILD_NUMBER}
docker run -itd -p 80:80 test:v${BUILD_NUMBER}
Advanced...
Add build step
Save
Apply

Manual build.

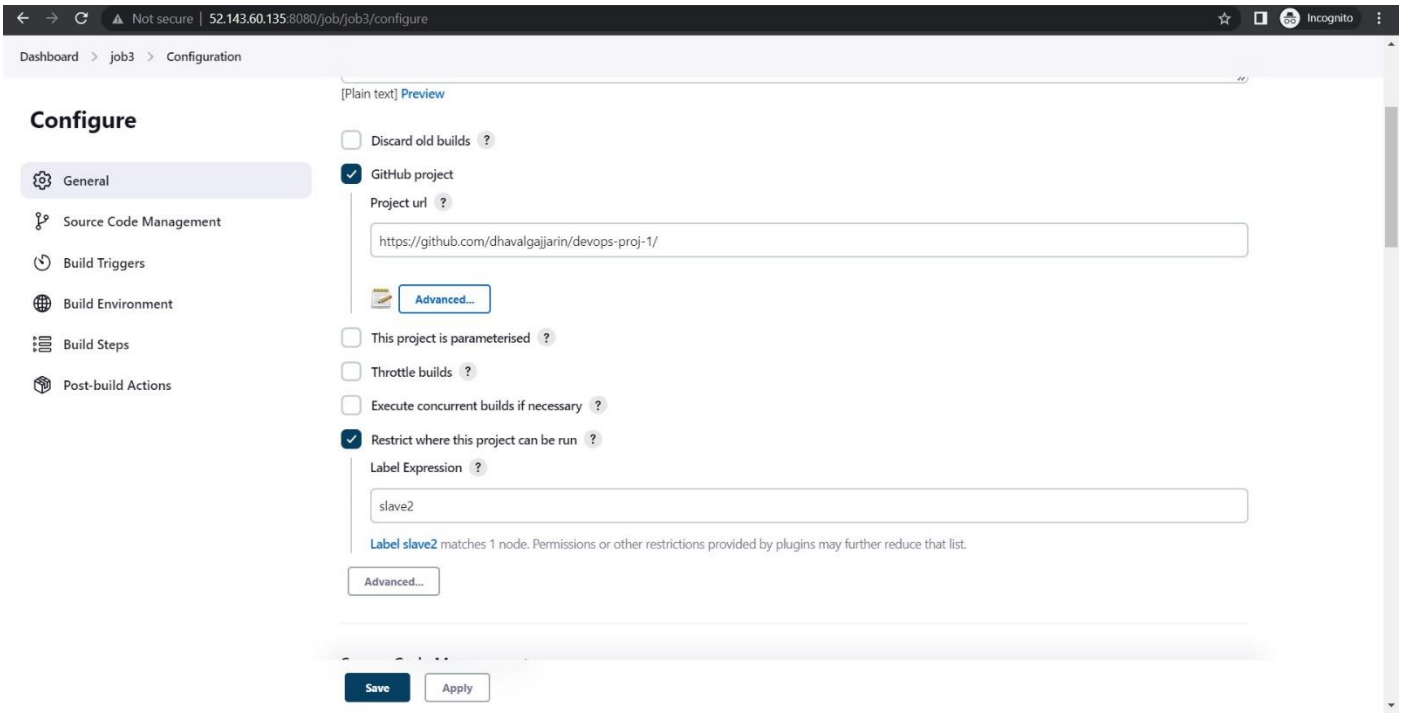


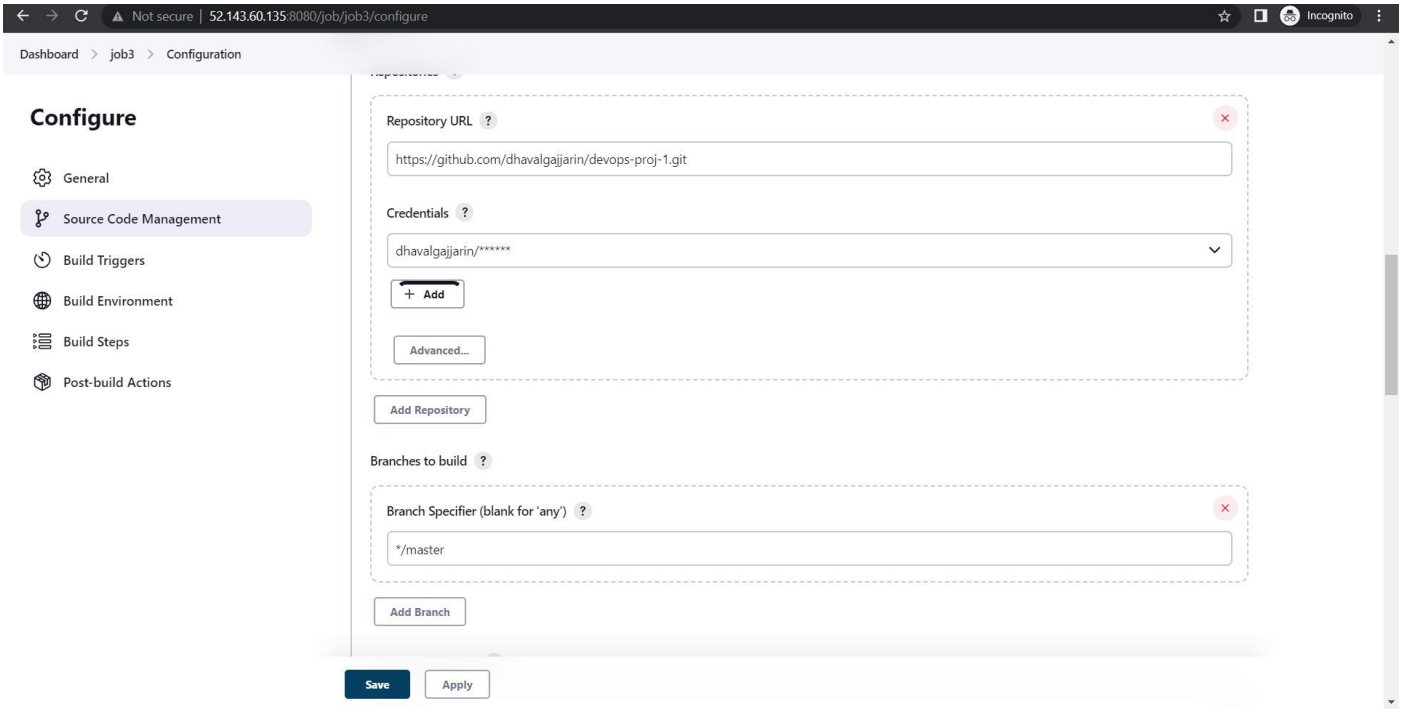
Automatic build using commit.



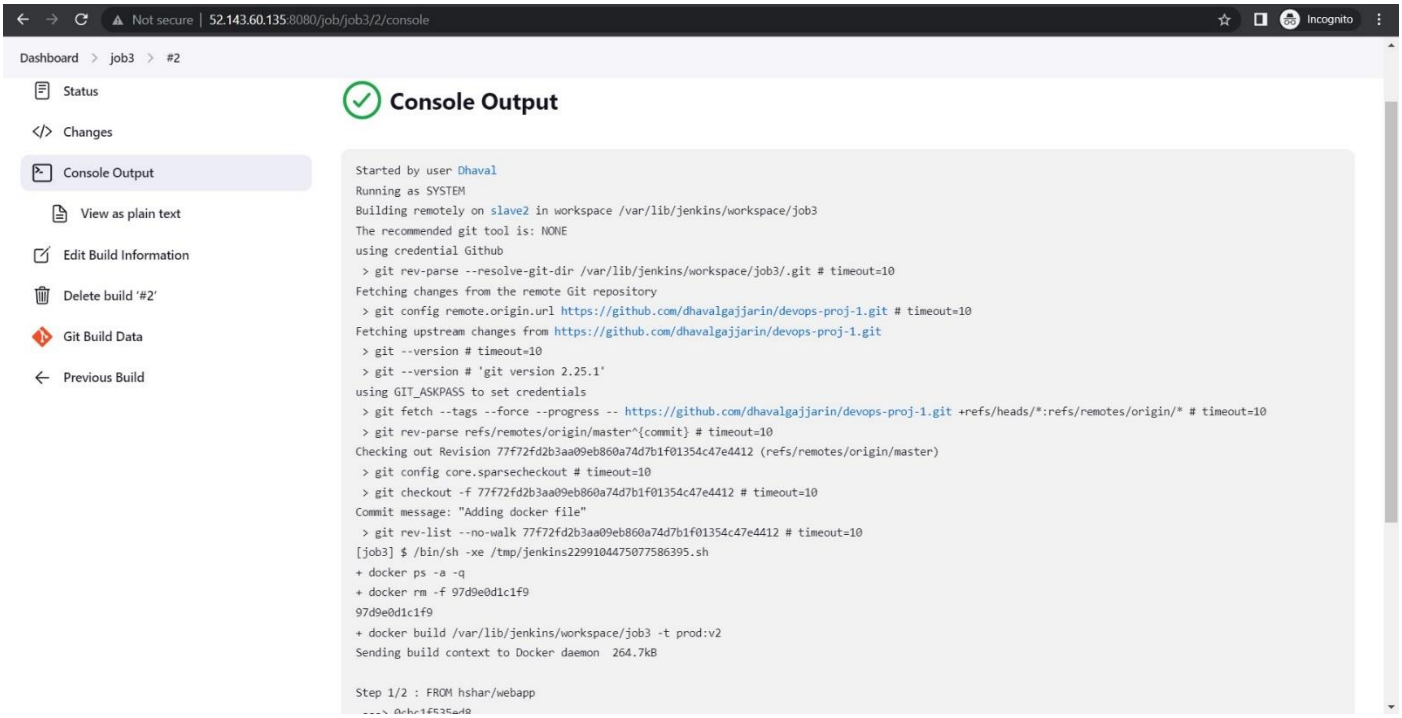
Open browser URL and test it.

Create Jenkins Job3 which can checkout and build and deploy to the production environment.

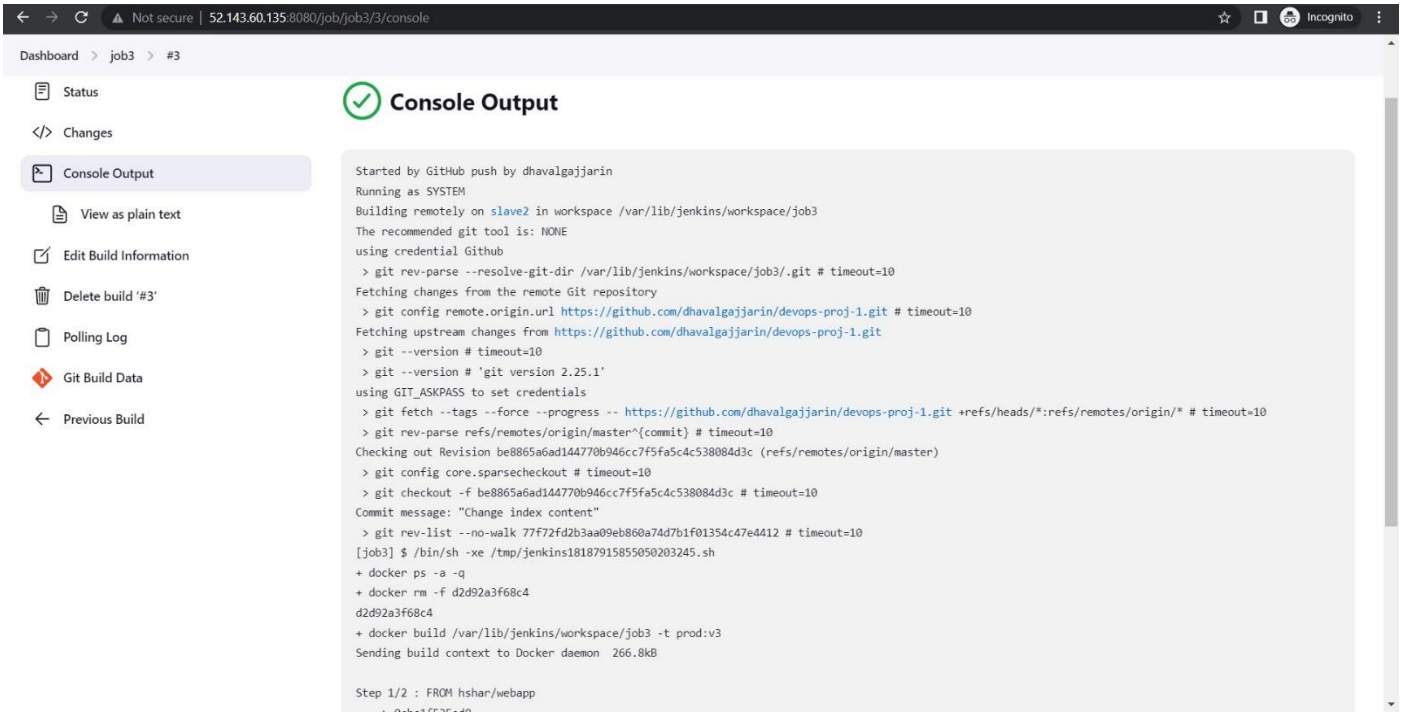




Manual build.



Automatic build using commit.



Open browser URL and test it.