

Project 2-

=====

Step1:-

Create 1 instance named “Jenkins_Terraform_Ansible >

Install terraform

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform -y
```

sudo nano main.tf

```
provider "aws" {
  secret_key = ""
  access_key = ""
  region = "us-west-1"
}

resource "aws_instance" "K8-M" {
  ami = ""
  instance_type = "t2.medium"
  key_name = ""
  tags = {
    Name = "Kmaster"
  }
}

resource "aws_instance" "K8-S1" {
  ami = ""
  instance_type = "t2.medium"
  key_name = ""
  tags = {
    Name = "Kslave1"
  }
}
```

```
}
```

```
resource "aws_instance" "K8-S2" {
  ami = ""
  instance_type = "t2.medium"
  key_name = ""
  tags = {
    Name = "Kslave2"
  }
}
```

Terraform init

Terraform plan

Terraform apply

Instances (4) Info								
		Last updated	Connect	Instance state	Actions	Launch instances		
<input type="checkbox"/> Name ∅		Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	
<input type="checkbox"/>	Kslave2	i-0e5933ceeb5d08583	Running Q Q	t2.medium	2/2 checks passed View alarms +	us-east-1c		
<input type="checkbox"/>	Kmaster	i-0611cf003af131a55	Running Q Q	t2.medium	2/2 checks passed View alarms +	us-east-1c		
<input type="checkbox"/>	Jenkins_Terraform_Ansible	i-0a9ecb9339ce2c03d	Running Q Q	t2.medium	2/2 checks passed View alarms +	us-east-1c		
<input type="checkbox"/>	Kslave1	i-0e9ce41e7ebbea612	Running Q Q	t2.medium	Initializing View alarms +	us-east-1c		

You will get to see 3 ec2 instances running.

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.K8-S1: Creating...
aws_instance.K8-M: Creating...
aws_instance.K8-S2: Creating...
aws_instance.K8-S1: Still creating... [00m10s elapsed]
aws_instance.K8-M: Still creating... [00m10s elapsed]
aws_instance.K8-S2: Still creating... [00m10s elapsed]
aws_instance.K8-S1: Creation complete after 13s [id=i-0e9ce41e7ebbea612]
aws_instance.K8-M: Creation complete after 13s [id=i-0611cf003af131a55]
aws_instance.K8-S2: Creation complete after 13s [id=i-0e5933ceeb5d08583]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-28-63:~$
```

i-0a9ecb9339ce2c03d (Jenkins_Terraform_Ansible)
Public IPs: 54.226.70.54 Private IPs: 172.31.28.63

Step 2 :-

Install Ansible on the Jenkins_Terraform_Ansible

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

Step 3:-

Now create a cluster for KMaster

On Master:

Ssh-keygen

Sudo cat id_rsa.pub

Copy ssh key

Go on Kmaster

```
Cd .ssh  
Sudo nano authorized_keys  
Paste ssh key
```

Go on Master:

```
Cd /etc/ansible  
Ls  
Sudo nano hosts  
[test]  
Private Ip of Kmaster  
Ansible -m ping all
```

Step 4:-

Playbook Syntax:

Sudo nano playbook.yaml

```
---
- name: Installations on Master
  hosts: localhost
  become: true
  tasks:
    - name: Executing script on master
      script: Jenkins_terraform_ansible.sh

- name: Installations on test
  hosts: test
  become: true
  tasks:
    - name: Executing script on test
      script: K-master.sh
```

```
GNU nano 6.2
---
- name: Installations on Master
  hosts: localhost
  become: true
  tasks:
    - name: Executing script on master
      script: Jenkins_terraform_ansible.sh

- name: Installations on test
  hosts: test
  become: true
  tasks:
    - name: Executing script on test
      script: K-master.sh

^G Help          ^O Write Out      ^W Where Is      ^K Cut
^X Exit          ^R Read File       ^\ Replace       ^U Paste
```

i-0a9ecb9339ce2c03d (Jenkins_Terraform_Anible)
Public IPs: 54.226.70.54 Private IPs: 172.31.28.63

Step 5:-

Jenkins_terraform_ansible.sh

sudo apt update

sudo apt install openjdk-17-jre -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

```

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null sudo

apt-get update

sudo apt-get install jenkins -y

sudo apt-get install docker.io -y

```

The screenshot shows a terminal window with the AWS logo at the top. The command history is displayed:

```

GNU nano 6.2
sudo apt update
sudo apt install openjdk-17-jre -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
sudo apt-get install docker.io -y

```

At the bottom of the terminal, there are keyboard shortcuts:

- ^{^G} Help
- ^{^O} Write Out
- ^{^W} Where Is
- ^{^K} Cut
- ^{^T} E
- ^{^X} Exit
- ^{^R} Read File
- ^{^V} Replace
- ^{^U} Paste
- ^{^J} J

Below the terminal window, the instance ID and IP addresses are shown:

i-0a9ecb9339ce2c03d (Jenkins_Terraform_Ansible)
Public IPs: 54.226.70.54 Private IPs: 172.31.28.63

Step 6:-

Kmaster.sh

```

sudo apt update

sudo apt install openjdk-17-jre -y

sudo apt install docker.io -y

```

The screenshot shows a terminal window with the AWS logo at the top. The command history is displayed:

```

GNU nano 6.2
sudo apt update
sudo apt install openjdk-17-jre -y
sudo apt install docker.io -y

```

At the top right of the terminal window, it says "kmaster.sh *".

Step 7:-

RUN PLAYBOOK :

```
ansible-playbook play.yaml -syntax-check
```

```
ansible-playbook play.yaml --check
```

```
ansible-playbook play.yaml
```

```
ubuntu@ip-172-31-20-201:/etc/ansible$ sudo nano play.yaml
ubuntu@ip-172-31-20-201:/etc/ansible$ ansible-playbook play.yaml

PLAY [Installations on Master] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Executing script on master] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=2    changed=1    unreachable=0    failed=0    skipped=0
ubuntu@ip-172-31-20-201:/etc/ansible$ []

i-006ec4ade90e24b2a (jenkins-terra-ansible)
Public IPs: 54.164.117.172 Private IPs: 172.31.20.201
```

Step 8 :-

Open the Github repository and fork it.

Clone this new repository > Go to folder > Create a Dockerfile

Dockerfile syntax

```
FROM ubuntu
```

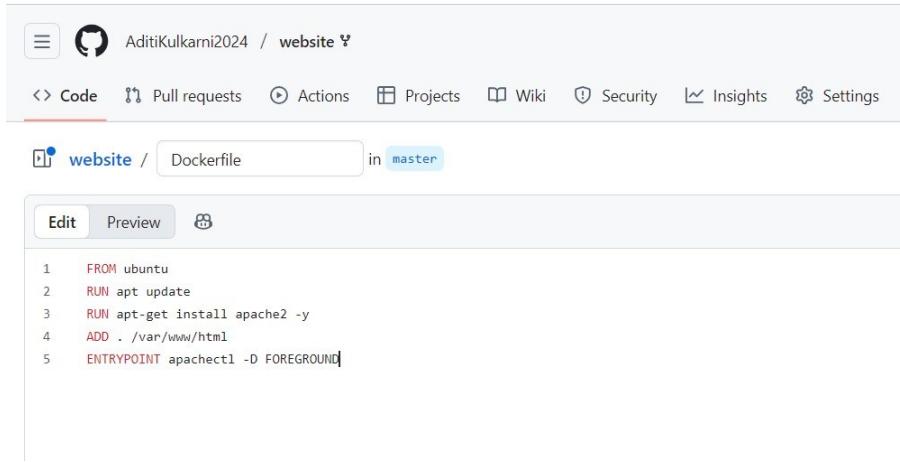
```
RUN apt update
```

```
RUN apt-get install apache2 -y
```

The screenshot shows a GitHub fork creation dialog. At the top, there's a search bar and navigation links for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. Below that, a section titled "Create a new fork" explains what a fork is: "A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks." It shows the current owner as "AditiKulkarni2024" and a repository name field containing "website". A note says, "By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further." There's a "Description" field with a character limit of 350. At the bottom, there are checkboxes for "Copy the `master` branch only" and "Contribute back to `https://github.com/aditi2024/website` by adding your own branch. Learn more.", and a note stating "You are creating a fork in your personal account." A "Create fork" button is at the very bottom right.

```
ADD . /var/www/html
```

```
ENTRYPOINT apachectl -D FOREGROUND
```



A screenshot of a GitHub repository page for a project named "website". The "Code" tab is selected. Below it, a "Dockerfile" tab is active, showing the following content:

```
1 FROM ubuntu
2 RUN apt update
3 RUN apt-get install apache2 -y
4 ADD . /var/www/html
5 ENTRYPOINT apachectl -D FOREGROUND
```

Step 9:- Create deploy.yaml file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: custom-deployment
  labels:
    app: custom
spec:
  replicas: 2
  selector:
    matchLabels:
      app: custom
  template:
    metadata:
      labels:
        app: custom
    spec:
      containers:
        - name: custom
          image: docker6767/image
          ports:
            - containerPort: 80
```



A screenshot of a terminal window titled "deploy.yaml". The content of the file is:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: custom-deployment
  labels:
    app: custom
spec:
  replicas: 2
  selector:
    matchLabels:
      app: custom
  template:
    metadata:
      labels:
        app: custom
    spec:
      containers:
        - name: custom
          image: docker6767/image
          ports:
            - containerPort: 80
```

matchLabels:

app: custom

template:

metadata:

labels:

app: custom

spec:

containers:

- name: custom

- image: docker6767/image

- ports:**

- containerPort: 80

Create SVC.yml

```
apiVersion: v1
kind: Service
metadata:
  name: my-custom-deployment
spec:
  type: NodePort
  ports:
```

```
- targetPort: 80
```

```
  port: 80
  nodePort: 30008
```

```
selector:
```

```
  app: custom
```

Git status

Git add .

Git commit -m "add dockerfile"

Git branch > you will get a master branch

The screenshot shows a GitHub repository page for 'AditiKulkarni2024 / website'. The repository has 1 branch and 0 tags. It contains files like Dockerfile, images, and index.html. The Dockerfile was created by AditiKulkarni2024 and updated 7 years ago. The repository has 3 commits and 0 forks. On the right, there are sections for About (No description), Activity (0 stars, 0 watches, 0 forks), Releases (No releases, Create a new), and Packages.

The screenshot shows the GitHub Webhooks settings page. It includes fields for Payload URL (http://54.164.117.172:8080/github-webhook, with a warning about missing a scheme), Content type (application/x-www-form-urlencoded), and Secret. Under SSL verification, 'Enable SSL verification' is selected. A section for triggering events includes options for Just the push event, Send me everything, and Let me select individual events. The Active checkbox is checked, and an 'Add webhook' button is at the bottom.

Step 10:- Create Nodes-

Nodes								
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	Built-In Node	Linux (amd64)	In sync	3.24 GiB	0 B	3.24 GiB	0ms	
	slave	Linux (amd64)	In sync	3.24 GiB	0 B	3.24 GiB	97ms	

Icon: S M L Legend

Create a Pipeline job

Jenkins / project-job / Configuration

Configure

- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
- Trigger builds remotely (e.g., from scripts) ?

Triggers (selected)

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

```
Script ?  
1 pipeline {  
2     agent none  
3  
4     environment {  
5         // Make sure the credentials ID is correct and doesn't have line breaks  
6         DOCKERHUB_CREDENTIALS  credentials('a33c448b-a34e-42f2-b98d-e9e7e890a018')  
7     }  
8     stages {  
9 }
```

Buttons: Save | Apply

```
pipeline{
    agent none
    environment {

DOCKERHUB_CREDENTIALS=credentials('

    }

stages{
    stage('Hello'){
        agent{
            label 'KMaster'

        }
    }

steps{
    echo 'Hello World'

}
}

stage('git'){
    agent{
        label 'KMaster'

    }
}

steps{
    git'URL'
}
```

```

}

}

stage('docker') {
    agent {
        label 'KMaster'
    }
    steps {
        sh 'sudo docker build /home/ubuntu/jenkins/workspace/FinalProject -t docker6767/image'

        sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'

        sh 'sudo docker push docker6767/image'
    }
}

```

The screenshot shows a Jenkins pipeline configuration and its build history.

Pipeline Configuration:

```

} } stage('Kubernetes') { agent { label 'KMaster' } steps {

```

Job Details:

- Status:** project-job
- Permalinks:** project-job
- Actions:**
 - Changes
 - Build Now
 - Configure
 - Delete Pipeline
 - GitHub
 - Stages
 - Rename
 - Pipeline Syntax
 - GitHub Hook Log

Build History:

Build #	Timestamp	Status
#2	6:34 pm	Failed (red)
#1	6:31 pm	Failed (red)

```
sh 'kubectl create -f deploy.yml'
```

```
sh 'kubectl create -f svc.yml'

}

}

}

}
```

The screenshot shows a terminal window within the AWS CloudWatch interface. The terminal output is as follows:

```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-16-21:~$ history
 1 sudo apt update -y
 2 sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
 3 wget -O https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
 4 gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint
 5 echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(grep -oP '^U_CODENAME=.*' /etc/os-release || lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
 6 sudo apt update
 7 sudo apt-get install terraform
 8 terraform --version
 9 sudo nano main.tf
10 sudo nano main.tf
11 terraform init
12 terraform plan
13 sudo nano main.tf
14 terraform init
15 terraform plan
16 sudo nano main.tf
17 terraform init
18 terraform plan
19 sudo nano main.tf
20 terraform init
21 terraform plan
22 terraform apply
23 history
```

Below the terminal output, the session details are shown:

i-0eb1c0ac734bf93c8 (Jenkins-Terraform-Ansible)
Public IPs: 54.208.117.223 Private IPs: 172.31.16.21