Here's a step-by-step project outline where AWS infrastructure is built using Terraform and incorporates Jenkins, Ansible, Kubernetes, and Docker.

**Project Overview:**

The goal of this project is to:

* Use **Terraform** to provision AWS resources.
* Set up **Jenkins** as a CI/CD tool.
* Use **Ansible** for configuration management.
* Deploy **Docker** containers in **Kubernetes** clusters.

**Technologies:**

* **AWS**: Cloud provider for hosting infrastructure.
* **Terraform**: Infrastructure as Code (IaC) tool to automate AWS resource creation.
* **Jenkins**: CI/CD tool to automate the deployment process.
* **Ansible**: Configuration management tool to configure servers.
* **Docker**: Containerization platform to package applications.
* **Kubernetes**: Orchestration tool to manage Docker containers.

**Step 1: Set up AWS Infrastructure with Terraform**

1. **Install Terraform** and configure AWS credentials.
2. **Define AWS resources** (VPC, EC2 instances, Security Groups, EKS (Elastic Kubernetes Service) cluster) in Terraform.
3. **Provision infrastructure** by running Terraform scripts.

**Sample main.tf for AWS EKS Cluster:**

hcl

Copy code

provider "aws" {

region = "us-west-2"

}

resource "aws\_vpc" "main\_vpc" {

cidr\_block = "10.0.0.0/16"

tags = {

Name = "main\_vpc"

}

}

resource "aws\_subnet" "subnet1" {

vpc\_id = aws\_vpc.main\_vpc.id

cidr\_block = "10.0.1.0/24"

}

resource "aws\_eks\_cluster" "eks\_cluster" {

name = "eks-cluster"

role\_arn = aws\_iam\_role.eks\_role.arn

vpc\_config {

subnet\_ids = [aws\_subnet.subnet1.id]

}

}

* + This will create a VPC and an EKS cluster to manage the Kubernetes environment.

1. **Deploy the infrastructure**:

bash

Copy code

terraform init

terraform apply

**Step 2: Set up Jenkins for CI/CD**

1. **Install Jenkins** on an EC2 instance or use AWS Elastic Beanstalk.
2. **Create a Jenkins Pipeline** to automate infrastructure provisioning and application deployment.

**Sample Jenkinsfile**:

groovy

Copy code

pipeline {

agent any

stages {

stage('Terraform Init') {

steps {

sh 'terraform init'

}

}

stage('Terraform Apply') {

steps {

sh 'terraform apply -auto-approve'

}

}

stage('Deploy to Kubernetes') {

steps {

sh 'kubectl apply -f deployment.yaml'

}

}

}

*}*

* + *This* Jenkins pipeline will automate the Terraform provisioning and Kubernetes deployment.

**Step 3: Configure Ansible for Server Configuration**

1. **Install Ansible** on your control machine.
2. **Create an Ansible playbook** to configure AWS EC2 instances (installing Docker, setting up security, etc.).

**Sample playbook.yml:**

yaml

Copy code

---

- hosts: aws\_instances

become: yes

tasks:

- name: Install Docker

apt:

name: docker.io

state: present

- name: Start Docker Service

service:

name: docker

state: started

enabled: yes

1. **Run the playbook** from your control machine to configure EC2 instances:

bash

Copy code

ansible-playbook -i inventory.ini playbook.yml

**Step 4: Set up Docker and Kubernetes**

1. **Build Docker images** of your application and push them to a Docker registry (DockerHub or AWS ECR).

**Dockerfile** example:

Dockerfile

Copy code

FROM node:14

WORKDIR /app

COPY . .

RUN npm install

CMD ["npm", "start"]

Build and push the Docker image:

bash

Copy code

docker build -t myapp:latest .

docker tag myapp:latest <dockerhub\_username>/myapp:latest

docker push <dockerhub\_username>/myapp:latest

1. **Create Kubernetes manifests** to deploy your Docker container.

**deployment.yaml:**

yaml

Copy code

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp-deployment

spec:

replicas: 2

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: myapp

image: <dockerhub\_username>/myapp:latest

ports:

- containerPort: 3000

1. **Deploy to Kubernetes**:

bash

Copy code

kubectl apply -f deployment.yaml

**Step 5: Automate the Entire Process with Jenkins**

* Jenkins will now be responsible for running the Terraform scripts, configuring EC2 instances with Ansible, building Docker containers, and deploying them to Kubernetes.

**Project Structure:**

css

Copy code

project/

├── ansible/

│ └── playbook.yml

├── jenkins/

│ └── Jenkinsfile

├── terraform/

│ ├── main.tf

│ ├── outputs.tf

│ └── variables.tf

└── k8s/

└── deployment.yaml

**Conclusion:**

This project covers provisioning AWS infrastructure using Terraform, configuring servers with Ansible, automating the deployment process using Jenkins, and managing containers using Docker and Kubernetes. You can extend it further by integrating monitoring tools like Prometheus/Grafana or adding additional security and networking layers.

ansible-playbook main.yml -i 172.31.21.193, --private-key /home/ubuntu/.ssh/authorized\_keys -u ubuntu

/home/ubuntu/.ssh

---

# tasks file for jenkins

- hosts: 172.31.21.193

become: true

tasks:

- name: add jenkins key

apt\_key: # ansible module to add key

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

state: present

- name: add jenkins repo

apt\_repository: #ansible module to add repo

repo: 'deb https://pkg.jenkins.io/debian-stable binary/'

state: present

- name: install java #install java as jenkins need jdk

apt:

name: openjdk-11-jre

state: present

- name: install jenkins #install jenkins

apt:

name: jenkins

state: present

- name: start jenkins service

service:

name: jenkins

state: started

- name: enable jenkins to start at boot time

service:

name: jenkins

enabled: yes

- name: Install Docker

apt:

name: docker.io

state: present

- name: Start Docker Service

service:

name: docker

state: started

enabled: yes

git config --global user.email "you@example.com"