📌 DevOps Project Guide – End-to-End Explanation & Setup
This guide will walk you through each component of the Automated AWS Infrastructure & CI/CD Deployment Project. I'll break it down into:

1️⃣ Infrastructure Setup with Terraform
2️⃣ Configuration Management with Ansible
3️⃣ CI/CD Pipeline with Jenkins & Maven
4️⃣ Containerization & Kubernetes Deployment
5️⃣ Helm Charts for Kubernetes
6️⃣ Monitoring & Logging Setup
7️⃣ Security & IAM Best Practices

1️⃣ Infrastructure Setup with Terraform
Terraform automates the creation of AWS infrastructure. The key resources include:

✅ EC2 Instances – Deploy Jenkins, Kubernetes Master & Worker Nodes
✅ S3 Bucket – Store Terraform state securely
✅ DynamoDB Table – Lock state files to prevent conflicts
✅ IAM Roles & Policies – Secure access to AWS resources

📜 Terraform Code (AWS Resources)
The script provisions an EC2 instance for Jenkins, a backend S3 bucket for Terraform state, and a DynamoDB table for state locking.

hcl
Copy
Edit
```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "jenkins_server" {
  ami           = "ami-052c08d70def0ac62"
  instance_type = "t2.micro"
  key_name      = "my-key"
  security_groups = ["jenkins-security-group"]

  tags = {
    Name = "Jenkins-Server"
  }
}

resource "aws_s3_bucket" "terraform_state" {
  bucket = "my-terraform-state-bucket"
}

resource "aws_dynamodb_table" "terraform_locks" {
  name         = "terraform-lock"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "LockID"

  attribute {
    name = "LockID"
    type = "S"
  }
}

terraform {
  backend "s3" {
    bucket         = "my-terraform-state-bucket"
    key            = "state/terraform.tfstate"
    region         = "us-east-1"
    dynamodb_table = "terraform-lock"
```

```
  }
}
```
🔹 Steps to Apply Terraform

Install Terraform (terraform init)
Plan the infrastructure (terraform plan)
Apply the changes (terraform apply -auto-approve)
2️⃣ Configuration Management with Ansible
Ansible automates the setup of Jenkins on the EC2 instance.

📜 Ansible Playbook for Jenkins Setup
This script installs Java, Jenkins, and enables it to start automatically.

yaml
Copy
Edit
```
- name: Install Jenkins on EC2
  hosts: jenkins_servers
  become: yes
  tasks:
    - name: Install Java
      apt:
        name: openjdk-11-jdk
        state: present

    - name: Install Jenkins
      shell: |
        wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
        sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
        sudo apt update && sudo apt install jenkins -y

    - name: Start and enable Jenkins
      systemd:
        name: jenkins
        state: started
        enabled: yes
```
🔹 Steps to Apply Ansible Playbook

Install Ansible (sudo apt install ansible -y)
Create an inventory file (hosts.ini) listing the Jenkins server
Run the playbook (ansible-playbook -i hosts.ini jenkins-setup.yml)
3️⃣ CI/CD Pipeline with Jenkins & Maven
This Jenkins pipeline automates:

✅ Code Checkout – Fetches code from GitHub
✅ Build – Uses Maven to build the application
✅ Artifact Storage – Uploads the JAR file to AWS S3
✅ Deployment – Deploys the application to Kubernetes

📜 Jenkinsfile
groovy
Copy
Edit
```
pipeline {
  agent any

  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/example/my-app.git'
      }
    }
```

```
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }

    stage('Store Artifact') {
      steps {
        sh 'aws s3 cp target/my-app.jar s3://my-artifact-bucket/'
      }
    }

    stage('Deploy to Kubernetes') {
      steps {
        sh 'kubectl apply -f k8s/deployment.yaml'
      }
    }
  }
}
```

4️⃣ Containerization & Kubernetes Deployment
The application is containerized using Docker and deployed into Kubernetes.

📜 Dockerfile
dockerfile
Copy
Edit
```
FROM openjdk:11
COPY target/my-app.jar /app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

5️⃣ Helm Charts for Kubernetes Deployment
Helm automates Kubernetes deployments.

📜 Helm values.yaml
yaml
Copy
Edit
```
image:
  repository: my-dockerhub-user/my-app
  tag: latest
```

🔷 Steps to Deploy with Helm

Install Helm (helm install my-app ./helm-chart)
Check deployment (kubectl get pods)

6️⃣ Monitoring & Logging Setup (ELK & Dynatrace)
✅ Elasticsearch collects logs from Kubernetes
✅ Kibana visualizes logs in real-time
✅ Dynatrace provides application monitoring

🔷 Steps to Set Up ELK

Deploy Elasticsearch (helm install elasticsearch elastic/elasticsearch)
Deploy Kibana (helm install kibana elastic/kibana)
Integrate logs with Fluentd

7️⃣ Security & IAM Best Practices
✅ IAM Roles – Restrict access to AWS resources
✅ S3 & DynamoDB – Store Terraform state securely

📜 Terraform Backend Storage
hcl
Copy
Edit
```
terraform {
  backend "s3" {
```

```
    bucket         = "my-terraform-state-bucket"
    key            = "state/terraform.tfstate"
    region         = "us-east-1"
    dynamodb_table = "terraform-lock"
  }
}
```

📌 Summary & Next Steps

🚀 This project covers Terraform, Ansible, Jenkins, Docker, Kubernetes, Helm, and ELK Stack to automate infrastructure, CI/CD, and monitoring.

✅ What you should do next?

Set up Terraform & apply configurations
Run the Ansible playbook to configure Jenkins
Configure Jenkins CI/CD pipeline
Build & deploy the application to Kubernetes
Monitor logs using ELK & Dynatrace

📌 Final Note

If you need GitHub repositories or additional guidance, let me know. I will provide step-by-step GitHub repository setup if needed. 🚀