BLUE-GREEN DEPLOYMENT ON AWS WITH JENKINS, ANSIBLE & ELB

1. PROJECT OVERVIEW.

Objective: Implement a **Blue-Green Deployment strategy** on AWS using **Jenkins, Ansible, and ELB**, ensuring zero downtime, automated rollback, and seamless application updates.

2. Architecture & Workflow

Architecture Components:

- Jenkins: Automates CI/CD pipeline
- Ansible: Manages infrastructure and deployments
- AWS EC2: Hosts the application
- AWS ELB (Elastic Load Balancer): Routes traffic to Blue or Green environment
- Auto Scaling Group (ASG): Ensures high availability
- Docker: Runs the application in containers
- Git: Stores application and infrastructure code

Workflow Steps:

1. **Develop & Commit Code**: Developer pushes changes to Git.
2. **Jenkins Triggers Build**: Detects changes and starts the pipeline.
3. **Build & Test**: Jenkins builds a Docker image and runs tests.
4. **Deploy to New Environment (Green)**: Ansible deploys the new version.
5. **Traffic Switching via ELB**: Once validated, ELB shifts traffic to Green.
6. **Rollback Mechanism**: If issues are detected, ELB reverts to Blue.

3. Tech Stack & Prerequisites

Tech Stack:

- AWS (EC2, ELB, Auto Scaling, IAM)
- Jenkins (Pipeline as Code)
- Ansible (Infrastructure Automation)
- Docker (Containerization)
- Git (Version Control)

Prerequisites:

1. AWS Account with IAM roles configured
2. Jenkins installed on a master node
3. Ansible installed on the Jenkins server
4. Docker installed on EC2 instances
5. Git repository with application source code

4. Infrastructure Setup

Step 1: Create EC2 Instances

Launch two EC2 instances (blue and green) with the required configurations.

Step 2: Configure Security Groups

- Open ports **22 (SSH), 80 (HTTP), 443 (HTTPS)**.
- Allow Jenkins server to access EC2 instances.

Step 3: Setup AWS ELB

- Create an **Application Load Balancer (ALB)**.
- Attach Blue and Green instances as target groups.

6. CI/CD Pipeline Configuration

Jenkins Pipeline Script

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/user/repository.git'
            }
        }
        stage('Build & Test') {
            steps {
                sh 'docker build -t myapp:latest .'
                sh 'docker run --rm myapp:latest pytest'
            }
        }
        stage('Deploy to Green') {
            steps {
                sh 'ansible-playbook -i inventory deploy.yml --extra-vars
"target=green"'
            }
        }
        stage('Switch Traffic to Green') {
            steps {
                sh 'ansible-playbook -i inventory switch.yml'
            }
        }
        stage('Cleanup Old Version') {
```

```
        steps {
            sh 'ansible-playbook -i inventory cleanup.yml'
        }
    }
  }
}
```

6. Ansible Playbooks

Inventory File (inventory):

```
[blue]
blue-instance-ip ansible_ssh_user=ec2-user
[green]
green-instance-ip ansible_ssh_user=ec2-user
```

Deploy Application (deploy.yml):

```
- hosts: "{{ target }}"
  become: yes
  tasks:
    - name: Pull Docker Image
      command: docker pull myapp:latest

    - name: Run Application
      command: docker run -d -p 80:80 myapp:latest
```

Switch Traffic (switch.yml):

```
- hosts: localhost
  tasks:
    - name: Update ELB Target Group
      command: aws elbv2 modify-target-group --target-group-arn GREEN_TARGET_GROUP_ARN --action deregister
```

Rollback (rollback.yml):

```
- hosts: localhost
  tasks:
    - name: Revert Traffic to Blue
      command: aws elbv2 modify-target-group --target-group-arn BLUE_TARGET_GROUP_ARN --action register
```

7. Blue-Green Switching & Rollback Strategy

Switching Traffic

- ELB shifts traffic from **Blue to Green** after validation.
- DNS updates propagate within seconds.

Rollback Plan

- If an issue occurs, **switch back to Blue**.
- Old version remains running for quick recovery.

8. If you want Security check in this projects so follow theses topics.

- Implement **IAM roles** for least privilege access.
- Use **AWS Secrets Manager** for credentials.
- Enable **CloudWatch monitoring** for performance insights.
- Scan Docker images with **Trivy** for vulnerabilities.
- Automate compliance checks with **AWS Config**.

"SUCCESS ISN'T ABOUT LUCK; IT'S ABOUT CONSISTENCY, LEARNING, AND EXECUTION. KEEP PUSHING FORWARD!"