# Securing Swiggy Clone App Deployment on AWS: A Comprehensive Guide to Building a DevSecOps Pipeline with Terraform, Jenkins, SonarQube, Trivy, Argocd, and EKS

## 1. Infrastructure Setup with Terraform

Terraform is a powerful Infrastructure as Code (IaC) tool used to provision and manage cloud resources. Here's how to set up your infrastructure:

**Steps:**

1. **Install Terraform:**
    - Download and install Terraform from its official website.
    - Verify installation using `terraform --version`.
2. **Write Terraform Configuration:**
    - Create a `main.tf` file to define AWS resources, such as VPCs, subnets, security groups, and EKS clusters.

Example snippet to create an EKS cluster:

```
resource "aws_eks_cluster" "swiggy_clone" {
  name     = "swiggy-clone-cluster"
  role_arn = aws_iam_role.eks_cluster_role.arn
  vpc_config {
    subnet_ids = aws_subnet.eks_subnet.*.id
  }
```

    - }

3. **Initialize Terraform:**
   - Run `terraform init` to download provider plugins.
4. **Plan and Apply Configuration:**
   - Execute `terraform plan` to preview the changes.
   - Deploy resources using `terraform apply`.

## 2. CI/CD Pipeline with Jenkins

Jenkins is an open-source automation server used for building and deploying applications.

**Steps:**

1. **Install Jenkins:**
   - Launch an EC2 instance and install Jenkins.
   - Configure Jenkins with necessary plugins such as Kubernetes, Git, and Pipeline.
2. **Create a Jenkins Pipeline:**
   - Write a Jenkinsfile to define your CI/CD pipeline.

Example pipeline stages:
```
pipeline {
 agent any
 stages {
  stage('Clone Repository') {
   steps {
    git 'https://github.com/your-repo/swiggy-clone-app.git'
   }
  }
  stage('Build') {
   steps {
    sh 'docker build -t swiggy-clone .'
   }
  }
 }
```

   - }
3. **Integrate with EKS:**
   - Use the Kubernetes plugin to deploy applications to your EKS cluster.

## 3. Code Quality Analysis with SonarQube

SonarQube is a tool to ensure your code adheres to quality standards.

**Steps:**

1. **Install SonarQube:**
   - Deploy SonarQube on an EC2 instance or use a containerized version.
2. **Configure Jenkins with SonarQube:**
   - Install the SonarQube plugin in Jenkins.

Update the Jenkins pipeline to include code analysis:

```
stage('Code Analysis') {
 steps {
  withSonarQubeEnv('SonarQube') {
   sh 'sonar-scanner'
  }
 }
```

   - }
3. **Analyze Results:**
   - Access the SonarQube dashboard to review code quality metrics and resolve any issues.

## 4. Vulnerability Scanning with Trivy

Trivy scans your container images for vulnerabilities.

**Steps:**

1. **Install Trivy:**
    - Download and install Trivy on your Jenkins server.
2. **Integrate Trivy in Pipeline:**

Add a vulnerability scanning stage in your Jenkinsfile:

```
stage('Scan Vulnerabilities') {
  steps {
    sh 'trivy image swiggy-clone:latest'
  }
```

    - }
3. **Review Reports:**
    - Analyze the scan report and address any critical vulnerabilities.

## 5. GitOps with ArgoCD

ArgoCD automates continuous delivery of Kubernetes applications.

**Steps:**

1. **Install ArgoCD:**
    - Deploy ArgoCD in your EKS cluster using the following command:
      kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
2. **Configure ArgoCD:**
    - Add your application repository to ArgoCD.
    - Define an Application manifest to deploy the Swiggy Clone App.
3. **Automate Deployments:**
    - ArgoCD will monitor the Git repository and automatically sync changes to the cluster.

## 6. Deploying to Amazon EKS

Amazon EKS simplifies Kubernetes management and scales your application effortlessly.

**Steps:**

1. **Access the Cluster:**
    - Use `kubectl` to connect to the EKS cluster.
2. **Deploy the Application:**
    - Create Kubernetes manifests for your application (e.g., Deployment, Service).

Example `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swiggy-clone
spec:
  replicas: 3
  selector:
    matchLabels:
      app: swiggy-clone
  template:
    metadata:
      labels:
        app: swiggy-clone
    spec:
      containers:
      - name: swiggy-clone
        image: swiggy-clone:latest
        ports:
```

-
  ```
        - containerPort: 80
  ```

    - Apply the manifest: `kubectl apply -f deployment.yaml`.
3. **Expose the Application:**

Use a LoadBalancer service to expose the app:

```
apiVersion: v1
kind: Service
metadata:
  name: swiggy-clone-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
```

    - app: swiggy-clone

## 7. Securing the Pipeline

Security is crucial in every stage of the pipeline.

**Steps:**

1. **IAM Roles and Policies:**
    - Use AWS IAM roles with least privilege to secure access to resources.
2. **Network Security:**
    - Restrict access to your EKS cluster using security groups and private subnets.
3. **Secrets Management:**
    - Use AWS Secrets Manager or Kubernetes Secrets to store sensitive information securely.
4. **Audit and Monitoring:**
    - Enable logging and monitoring with AWS CloudWatch and Kubernetes tools like Prometheus.