Security Practices in Jenkins Pipeline

# 1. Infrastructure Security

## Terraform Scanning — Checkov Integration

All Terraform code in the /terraform/ directory is scanned using Checkov before deployment.  
  
Pipeline Reference:  
stage('Terraform Security Scan') {  
 steps {  
 sh 'checkov -d terraform/ --quiet --skip-check CKV\_AWS\_999'  
 }  
}  
  
Purpose:  
- Detects insecure configurations (e.g., public S3 buckets, open security groups).  
- Enforces encryption, least privilege, and logging policies.  
  
Best Practices:  
- Terraform code stored in private GitHub repository.  
- Jenkins enforces scan-before-apply.  
- AWS credentials injected via Jenkins credentials and AWS Secrets Manager.

## Policy-as-Code

Security and compliance are enforced using Checkov and optionally OPA.  
Policies ensure:  
- No untagged resources.  
- Restricted CIDRs for network resources.  
- Encrypted storage and secrets.

## Access Control

AWS IAM roles follow least privilege.  
CI/CD roles have restricted actions.  
Access tokens managed via AWS Secrets Manager.  
GitHub PAM Authentication and Jenkins SSO used.  
Sensitive credentials stored in Jenkins Credentials Store.

# 2. Container Security

## Image Scanning — Trivy Integration

Trivy scans Docker images for vulnerabilities during build.  
  
Pipeline Reference:  
stage('Image Scan') {  
 steps {  
 sh 'trivy image --severity HIGH,CRITICAL myapp:latest || true'  
 }  
}  
  
Best Practices:  
- High/Critical CVEs cause warnings or failures.  
- Reports archived in Jenkins.  
- Minimal base images used.  
- Explicit versions enforced.  
- Rebuilds triggered on base image updates.

## Runtime Security

Kubernetes manifests use strict Pod Security Contexts:  
- runAsNonRoot: true  
- readOnlyRootFilesystem: true  
- allowPrivilegeEscalation: false  
  
RBAC limits container permissions.  
Future integration with Falco or Kyverno.

# 3. Secrets Management

Secrets are managed across CI/CD, Terraform, Kubernetes, and local dev:  
- Jenkins Credentials Store for AWS keys, GitHub tokens.  
- AWS Secrets Manager for Terraform.  
- Kubernetes Secrets & ConfigMaps.  
- .env files (git-ignored) for local dev.  
  
Best Practices:  
- Secrets never stored in Git.  
- .env files are git-ignored.  
- Automatic rotation via AWS Secrets Manager.  
- Jenkins credential usage audited.

# 4. Source Code Security — TruffleHog Integration

TruffleHog scans for hard-coded secrets before builds.  
  
Pipeline Reference:  
stage('Secret Scan') {  
 steps {  
 sh 'trufflehog filesystem --directory . --fail'  
 }  
}  
  
Best Practices:  
- Blocks commits/builds with secrets.  
- Uses entropy and regex detection.  
- Enforced on PRs and pre-deployment builds.

# 5. Security Monitoring & Reporting

All scan outputs stored as Jenkins artifacts.  
Reports published to Jenkins dashboard.  
Metrics exported to Prometheus/Grafana.  
Alerts configured for:  
- High/Critical CVEs  
- Terraform misconfigurations  
- Secrets in code

# 6. Compliance & Continuous Improvement

Tool Coverage and Compliance:  
- Checkov: Terraform, K8s (CIS, NIST, PCI DSS)  
- Trivy: Containers, IaC, SBOMs  
- TruffleHog: Secrets detection  
- AWS IAM: Identity & Key rotation  
- Jenkins Reports: CI/CD audit evidence

# Summary of Security Controls

|  |  |  |  |
| --- | --- | --- | --- |
| Area | Tool/Integration | Jenkins Stage | Status |
| IaC Scanning | Checkov | Terraform Security Scan | ✅ Active |
| Secrets Scanning | TruffleHog | Secret Scan | ✅ Active |
| Container Scanning | Trivy | Image Scan | ✅ Active |
| Secrets Management | AWS Secrets Manager, Jenkins Store | Runtime | ✅ Active |
| Auth Control | GitHub PAM | Jenkins Login | ✅ Enforced |