CI/CD Pipeline & IaC Architecture

CI/CD Pipeline & IaC Architecture

🚀 CI/CD Pipeline Overview

Pipeline Stages

1. **Validate** - Terraform/Ansible linting, security scans
2. **Build** - Container images, Helm charts, installer packages
3. **Test** - Integration tests, security testing
4. **Deploy** - Staging/Production with manual gates
5. **Monitor** - Health checks, performance tests

Key Features

- Multi-environment deployment (staging/production)
- Automated certificate management (Let's Encrypt + self-signed fallback)
- Security scanning (SAST, container scanning, secret detection)
- Infrastructure validation and testing
- Automated rollback capabilities

🏗️ Infrastructure as Code Structure

```
homelab-iac/
├── terraform/
│   ├── main.tf                # Core infrastructure
│   ├── variables.tf          # Variable definitions
│   ├── outputs.tf            # Output definitions
│   └── environments/         # Environment-specific configs
├── ansible/
│   ├── playbooks/deploy.yml   # Main deployment playbook
│   ├── roles/                # Modular automation roles
│   └── inventory/            # Environment inventories
├── config/
│   ├── global.tfvars         # Global defaults
│   ├── production.tfvars     # Production overrides
│   ├── staging.tfvars        # Staging overrides
│   └── local-overrides.tfvars # Local development
├── scripts/
│   ├── generate-certificates.sh  # Certificate automation
│   └── build-installer.sh        # Package builder
└── dist/                     # Generated artifacts
```

🔐 Certificate Management

Automated Certificate Handling

- **Let's Encrypt**: DNS-01 challenge via Cloudflare API
- **Self-signed fallback**: Auto-generated CA and server certificates
- **Distribution**: Kubernetes secrets, GitLab SSL, system CA store
- **Renewal**: Automated via cron jobs and GitLab CI

Certificate Types Generated

- Root CA certificate (self-signed mode)
- Wildcard certificate for `*.homelab.local`
- Service-specific certificates (GitLab, Keycloak, etc.)
- Client certificates for service authentication

⚙️ Configuration Override System

Three-Layer Configuration

1. **Global defaults** (`config/global.tfvars`)
2. **Environment overrides** (`config/{env}.tfvars`)
3. **Local overrides** (`config/local-overrides.tfvars`)

Override Examples

```
# Production: High availability
keycloak = {
  replicas = 3
  resources = {
    limits = { memory = "4Gi", cpu = "2000m" }
  }
}

# Staging: Resource efficient
keycloak = {
  replicas = 1
  resources = {
    limits = { memory = "1Gi", cpu = "500m" }
  }
}
```

📦 Deployment Artifacts

Generated Outputs

- **Complete installer**: `homelab-installer.tar.gz`
- **Certificate bundle**: `homelab-certificates-{date}.tar.gz`
- **Kubeconfig files**: Admin and kang user configs
- **Access documentation**: Service URLs and credentials

Installation Process

```
# Extract and run installer
tar -xzf homelab-installer.tar.gz
cd homelab-installer
sudo ./install.sh
```

🔄 CI/CD Integration

GitLab Pipeline Triggers

- **Push to main**: Full validation and staging deployment
- **Manual gates**: Production deployment requires approval

- **Scheduled**: Health checks and performance testing
- **Certificate renewal**: Automated via pipeline schedules

Environment Promotion

```yaml
# Staging deployment (automated)
deploy-staging:
  environment: staging
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH

# Production deployment (manual)
deploy-production:
  environment: production
  when: manual
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

🛡️ Security Features

Zero Trust Implementation

- Default deny network policies
- mTLS for service communication
- RBAC with least privilege
- Automated security scanning
- Certificate-based authentication

Compliance & Monitoring

- Infrastructure drift detection
- Security policy validation
- Audit logging and retention
- Automated backup and recovery

CI/CD & IaC Documentation v1.0 - DevOps Team