ArgoCD Automation and GitOps Patterns

ArgoCD Automation and GitOps Patterns

Overview

This document details the automated ArgoCD setup and GitOps patterns for the unified cluster-snek platform.

ArgoCD Bootstrap Process

1. Initial Installation

```
# Generated bootstrap manifest structure
argocd-bootstrap/
├── namespace.yaml
├── argocd-install.yaml
├── argocd-cm.yaml
├── argocd-rbac-cm.yaml
├── argocd-server-ingress.yaml
└── argocd-credentials-secret.yaml
```

2. Automated Configuration

Repository Credentials

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: repo-credentials
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
type: Opaque
stringData:
  type: git
  url: https://github.com/${GITHUB_ORG}/${REPO_NAME}
  username: ${GITHUB_USERNAME}
  password: ${GITHUB_TOKEN}
```

SSO Integration

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cm
data:
  url: https://argocd.${BASE_DOMAIN}
  oidc.config: |
    name: Keycloak
    issuer: https://keycloak.${BASE_DOMAIN}/realms/master
    clientId: argocd
    clientSecret: $oidc.keycloak.clientSecret
    requestedScopes: ["openid", "profile", "email", "groups"]
    requestedIDTokenClaims: {"groups": {"essential": true}}
```

ApplicationSet Patterns

1. Multi-Cluster Pattern

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: multi-cluster-apps
  namespace: argocd
spec:
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{name}}-apps'
    spec:
      project: default
      source:
        repoURL: https://github.com/${GITHUB_ORG}/{{name}}
        targetRevision: main
        path: apps/
      destination:
        server: '{{server}}'
        namespace: argocd
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

2. Service Deployment Pattern

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: service-deployments
spec:
  generators:
  - list:
      elements:
      - service: keycloak
        namespace: auth
      - service: gitlab
        namespace: gitlab
      - service: monitoring
        namespace: monitoring
  template:
    metadata:
      name: '{{service}}'
    spec:
      source:
        repoURL: https://github.com/${GITHUB_ORG}/services
        path: '{{service}}'
        targetRevision: main
      destination:
        server: https://kubernetes.default.svc
        namespace: '{{namespace}}'
```

3. Environment-Based Pattern

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: env-based-apps
spec:
  generators:
  - git:
      repoURL: https://github.com/${GITHUB_ORG}/config
      revision: HEAD
      directories:
      - path: environments/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
      source:
        repoURL: https://github.com/${GITHUB_ORG}/config
        path: '{{path}}'
      destination:
        server: https://kubernetes.default.svc
```

Webhook Configuration

GitHub Webhook Setup

```
# Generated webhook configuration
webhook_url: https://argocd.${BASE_DOMAIN}/api/webhook
webhook_secret: ${ARGOCD_WEBHOOK_SECRET}
events:
  - push
  - pull_request
```

GitLab Integration

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: gitlab-webhook
  namespace: argocd
stringData:
  webhook.gitlab.secret: ${GITLAB_WEBHOOK_SECRET}
```

RBAC Configuration

Role Definitions

```yaml
policy.csv: |
  # Admins
  g, argocd-admins, role:admin

  # Developers
  p, role:developer, applications, *, */*, allow
  p, role:developer, logs, get, */*, allow
  p, role:developer, exec, create, */*, allow

  # Viewers
  p, role:readonly, applications, get, */*, allow
  p, role:readonly, logs, get, */*, allow

  # Service accounts
  p, role:ci-cd, applications, sync, */*, allow
  p, role:ci-cd, applications, action/*, */*, allow
```

SSO Group Mapping

```
policy.default: role:readonly
policy.scopes: groups
scopes: '[groups]'
```

Notification Configuration

Slack Notifications

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-notifications-cm
data:
  service.slack: |
    token: $slack-token

  template.app-deployed: |
    message: |
      {{if eq .serviceType "slack"}}:white_check_mark:{{end}} Application {{.app.metadata.name}} is now running new version.

  template.app-health-degraded: |
    message: |
      {{if eq .serviceType "slack"}}:exclamation:{{end}} Application {{.app.metadata.name}} has degraded health.
```

Sync Policies

Progressive Rollout

```
syncPolicy:
  automated:
    prune: true
    selfHeal: true
  syncOptions:
  - CreateNamespace=true
  retry:
    limit: 5
    backoff:
      duration: 5s
      factor: 2
      maxDuration: 3m
```

Manual Approval

```
syncPolicy:
  automated: false
  syncOptions:
  - ApplyOutOfSyncOnly=true
```

Multi-Tenancy

Project Isolation

```yaml
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
spec:
  description: Team A applications
  sourceRepos:
  - 'https://github.com/org/team-a-*'
  destinations:
  - namespace: 'team-a-*'
    server: https://kubernetes.default.svc
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceWhitelist:
  - group: '*'
    kind: '*'
```

Disaster Recovery

Backup Configuration

```yaml
# ArgoCD application state backup
apiVersion: batch/v1
kind: CronJob
metadata:
  name: argocd-backup
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: backup
            image: argoproj/argocd:v2.8.0
            command:
            - /bin/bash
            - -c
            - |
              argocd app list -o yaml > /backup/apps.yaml
              argocd proj list -o yaml > /backup/projects.yaml
```

Best Practices

1. **Repository Structure**
   - Separate repos for apps and configs
   - Environment-specific branches

- Clear naming conventions

2. **Security**
   - Use sealed secrets for credentials
   - Enable RBAC
   - Implement webhook secrets
   - Regular credential rotation
3. **Performance**
   - Limit application refresh interval
   - Use resource hooks wisely
   - Monitor sync queue
4. **Monitoring**
   - Enable metrics
   - Configure alerts
   - Track sync failures
   - Monitor resource usage