Kubernetes Manifests & GitOps with ArgoCD

Table of Contents

1. Introduction to Kubernetes Manifests
2. Kubernetes Manifest Examples
- Deployment Manifest
- Service Manifest
- Ingress Manifest
3. Introduction to GitOps with ArgoCD
4. Setting Up GitOps with ArgoCD
- Installing ArgoCD
- Accessing the ArgoCD UI
5. Creating GitOps Components
- GitOps Repository Structure
- GitOps Agent Creation
- GitOps Cluster Creation
- GitOps Repository Creation
- GitOps Applications Creation
6. Deploying Applications with ArgoCD
- ArgoCD Application Manifest
- Applying the ArgoCD Application
- Verifying the Deployment

1. Introduction to Kubernetes Manifests

Kubernetes manifests are declarative YAML files that define the desired state of Kubernetes resources.

2. Kubernetes Manifest Examples

Deployment Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: my-app:latest
          ports:
            - containerPort: 80
```

Service Manifest

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
```

Ingress Manifest

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-app-ingress
spec:
  rules:
    - host: myapp.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: my-app-service
                port:
                  number: 80
```

3. Introduction to GitOps with ArgoCD

GitOps is a declarative approach where Kubernetes resources are managed through Git repositories.

4. Setting Up GitOps with ArgoCD

Installing ArgoCD

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifest
```

Accessing the ArgoCD UI

```
kubectl port-forward svc/argocd-server -n argocd 8080:443
kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | bas
```

5. Creating GitOps Components

GitOps Repository Structure

```
gitops-repo/
■■■ apps/
■   ■■■ my-app/
■   ■   ■■■ deployment.yaml
■   ■   ■■■ service.yaml
■   ■   ■■■ ingress.yaml
■■■ argocd/
■   ■■■ application.yaml
```

GitOps Agent Creation

```
# Deploy ArgoCD Agent
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifest

# Verify the Agent
kubectl get pods -n argocd
```

GitOps Cluster Creation

```
# Create a Kubernetes Cluster (Example with kind)
kind create cluster --name gitops-cluster

# Verify the Cluster
kubectl cluster-info
```

GitOps Repository Creation

```
# Create and Initialize a Git Repository for GitOps
mkdir gitops-repo && cd gitops-repo
git init
git remote add origin https://github.com/my-org/gitops-repo.git
```

```
# Push Initial Structure
git add .
git commit -m "Initial GitOps repo structure"
git push origin main
```

GitOps Applications Creation

```
# Deploy GitOps Applications via ArgoCD
kubectl apply -f argocd/application.yaml -n argocd

# Sync the Application in ArgoCD
argocd app sync my-app
```

6. Deploying Applications with ArgoCD

ArgoCD Application Manifest

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  destination:
    namespace: default
    server: https://kubernetes.default.svc
  source:
    repoURL: https://github.com/my-org/gitops-repo.git
    path: apps/my-app
    targetRevision: main
  syncPolicy:
    automated:
      selfHeal: true
      prune: true
```

Applying the ArgoCD Application

```
kubectl apply -f application.yaml -n argocd
```

Verifying the Deployment

```
kubectl get deployments,services,ingress
```