CloudForge Kubernetes Integration Guide

Document ID: PRD-CF-025 **Last Updated:** 2024-03-01 **Owner:** CloudForge Product Team **Classification:** Public

---

Overview

CloudForge provides native Kubernetes support for deploying, managing, and scaling containerized applications. This guide covers Kubernetes integration features and best practices.

Supported Kubernetes Versions

Provider	Supported Versions
Amazon EKS	1.25, 1.26, 1.27, 1.28, 1.29
Google GKE	1.25, 1.26, 1.27, 1.28, 1.29
Azure AKS	1.25, 1.26, 1.27, 1.28, 1.29
Self-Managed	1.24+

Connecting a Cluster

Managed Kubernetes (EKS, GKE, AKS)

CloudForge can automatically provision and connect managed Kubernetes clusters.

Create New Cluster:

```
# cloudforge.yaml
resources:
  - type: kubernetes-cluster
    name: production-cluster
    provider: aws
    region: us-west-2
    version: "1.29"
```

```
    nodeGroups:
      - name: default
        instanceType: m5.large
        minSize: 2
        maxSize: 10
        desiredSize: 3
```

Connect Existing Cluster:

1. Go to **Infrastructure → Kubernetes**
2. Click **Connect Cluster**
3. Select provider and region
4. Choose existing cluster
5. CloudForge installs agent automatically

Self-Managed Kubernetes

For self-managed clusters:

1. Download CloudForge agent:

```
curl -fsSL https://get.cloudforge.novatech.com/agent | bash
```

2. Install agent with connection token:

```
cloudforge-agent install --token <your-token>
```

3. Verify connection:

```
cloudforge-agent status
```

Cluster Management

Node Groups

Define node pools for different workload types:

```yaml
nodeGroups:
  - name: general
    instanceType: m5.large
    minSize: 2
    maxSize: 10
    labels:
      workload: general

  - name: compute
    instanceType: c5.2xlarge
    minSize: 0
    maxSize: 20
    labels:
      workload: compute
    taints:
      - key: workload
        value: compute
        effect: NoSchedule

  - name: gpu
    instanceType: p3.2xlarge
    minSize: 0
    maxSize: 5
    labels:
      workload: gpu
    taints:
      - key: nvidia.com/gpu
        effect: NoSchedule
```

Auto-Scaling

Configure cluster auto-scaling:

```yaml
autoscaling:
  enabled: true
  metrics:
    - type: cpu
      target: 70
    - type: memory
      target: 80
  scaleDown:
    delay: 10m
    threshold: 50
```

Upgrades

CloudForge supports automated Kubernetes upgrades:

1. Go to **Cluster → Settings → Upgrades**
2. Select target version
3. Choose upgrade strategy:
   - **Rolling:** Node-by-node (recommended)
   - **Blue/Green:** New node group, drain old
4. Schedule upgrade window
5. Monitor progress

Deploying Applications

Using CloudForge Templates

```yaml
# cloudforge.yaml
applications:
  - name: web-api
    type: deployment
    image: myapp:latest
    replicas: 3
    resources:
      cpu: 500m
      memory: 512Mi
    ports:
      - 8080
    healthCheck:
      path: /health
      port: 8080
    autoscaling:
      minReplicas: 3
      maxReplicas: 10
      targetCPU: 70
```

Using Native Kubernetes Manifests

Deploy standard Kubernetes YAML:

```yaml
# cloudforge.yaml
kubernetes:
  manifests:
    - path: ./k8s/deployment.yaml
    - path: ./k8s/service.yaml
    - path: ./k8s/ingress.yaml
```

Using Helm Charts

Deploy Helm charts:

```yaml
# cloudforge.yaml
helm:
  releases:
    - name: nginx-ingress
      chart: ingress-nginx/ingress-nginx
      version: 4.8.0
      namespace: ingress-nginx
      values:
        controller:
          replicaCount: 2

    - name: prometheus
      chart: prometheus-community/prometheus
      version: 25.0.0
      namespace: monitoring
      valuesFile: ./helm/prometheus-values.yaml
```

Networking

Ingress Configuration

CloudForge supports multiple ingress controllers:

NGINX Ingress:

```yaml
ingress:
  type: nginx
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
  hosts:
    - host: api.example.com
```

```
      paths:
        - path: /
          service: web-api
          port: 8080
  tls:
    - hosts:
        - api.example.com
      secretName: api-tls
```

AWS ALB Ingress:

```
ingress:
  type: aws-alb
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
```

Service Mesh (Istio)

Enable Istio service mesh:

```
serviceMesh:
  enabled: true
  type: istio
  config:
    mtls: STRICT
    tracing:
      enabled: true
      samplingRate: 1.0
```

Network Policies

Define network policies:

```
networkPolicies:
  - name: api-policy
    podSelector:
      app: web-api
    ingress:
      - from:
          - podSelector:
              app: frontend
        ports:
```

```
        - port: 8080
  egress:
    - to:
        - podSelector:
            app: database
      ports:
        - port: 5432
```

Storage

Persistent Volumes

```
storage:
  - name: data-volume
    type: ebs
    size: 100Gi
    storageClass: gp3
    accessMode: ReadWriteOnce
```

Storage Classes

```
storageClasses:
  - name: fast-ssd
    provisioner: ebs.csi.aws.com
    parameters:
      type: gp3
      iops: "3000"
      throughput: "125"
    reclaimPolicy: Delete
    volumeBindingMode: WaitForFirstConsumer
```

Secrets Management

Integration with SecureVault

CloudForge integrates with SecureVault for secrets:

```yaml
secrets:
  vault:
    enabled: true
    path: secret/data/myapp
    keys:
      - name: DATABASE_URL
        key: db-connection-string
      - name: API_KEY
        key: external-api-key
```

Kubernetes Secrets

```yaml
secrets:
  kubernetes:
    - name: app-secrets
      data:
        DB_PASSWORD: ${SECUREVAULT:myapp/db/password}
```

Monitoring & Logging

Built-in Monitoring

CloudForge provides built-in Kubernetes monitoring:

- Node metrics (CPU, memory, disk)
- Pod metrics
- Container metrics
- Network metrics
- Custom metrics via Prometheus

Dashboard:

```yaml
monitoring:
  enabled: true
  prometheus:
    retention: 15d
  grafana:
    enabled: true
  alerts:
    - name: high-cpu
      condition: "avg(cpu_usage) > 80"
      duration: 5m
      severity: warning
```

Logging

```yaml
logging:
  enabled: true
  driver: fluentd
  destination: datadog
  filters:
    - namespace: production
      level: info
```

CI/CD Integration

DevPipeline Integration

```yaml
# .devpipeline.yaml
stages:
  - name: deploy
    steps:
      - name: Deploy to Kubernetes
        uses: cloudforge/deploy
        with:
          cluster: production-cluster
          manifest: ./cloudforge.yaml
          wait: true
          timeout: 10m
```

Rolling Updates

```yaml
deployment:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
  readinessProbe:
    httpGet:
      path: /ready
      port: 8080
    initialDelaySeconds: 10
    periodSeconds: 5
```

Canary Deployments

```
deployment:
  strategy:
    type: canary
    canary:
      steps:
        - weight: 10
          pause: 5m
        - weight: 50
          pause: 10m
        - weight: 100
      analysis:
        metrics:
          - name: error-rate
            threshold: 1
          - name: latency-p99
            threshold: 500
```

Security

Pod Security

```
security:
  podSecurityPolicy:
    enabled: true
    runAsNonRoot: true
    readOnlyRootFilesystem: true
    allowPrivilegeEscalation: false
```

RBAC

```
rbac:
  roles:
    - name: app-reader
      namespace: production
      rules:
        - apiGroups: [""]
          resources: ["pods", "services"]
          verbs: ["get", "list", "watch"]

  bindings:
```

```
    - role: app-reader
      subjects:
        - kind: ServiceAccount
          name: monitoring-sa
```

Image Security

```
security:
  imagePolicy:
    allowedRegistries:
      - registry.novatech.com
      - gcr.io
    scanImages: true
    blockHighVulnerabilities: true
```

Cost Optimization

Spot Instances

```
nodeGroups:
  - name: spot-workers
    instanceType: m5.large
    capacityType: SPOT
    spotMaxPrice: "0.05"
    minSize: 0
    maxSize: 20
```

Right-Sizing Recommendations

CloudForge analyzes resource usage and recommends: - Over-provisioned pods - Under-utilized nodes - Optimal instance types

View recommendations in **Cluster → Cost → Recommendations**

Troubleshooting

Common Issues

Pods not scheduling: - Check node resources - Review pod affinity/anti-affinity - Check taints and tolerations - Verify resource requests

Image pull errors: - Verify registry credentials - Check image name and tag - Ensure registry is accessible

Service not accessible: - Verify service selector matches pods - Check endpoint status - Review network policies - Verify ingress configuration

Debugging Commands

Via CloudForge CLI:

```
# Get pod status
cloudforge k8s pods --cluster production-cluster

# View logs
cloudforge k8s logs deployment/web-api --cluster production-cluster

# Execute into pod
cloudforge k8s exec deployment/web-api -- /bin/sh

# Describe resource
cloudforge k8s describe pod/web-api-xyz --cluster production-cluster
```

Best Practices

1. **Use namespaces** to isolate environments
2. **Set resource limits** on all containers
3. **Use liveness and readiness probes**
4. **Implement pod disruption budgets**
5. **Enable auto-scaling** for variable workloads
6. **Use managed node groups** when possible
7. **Regularly update** Kubernetes version
8. **Monitor cluster health** continuously

Related Documents: Getting Started (PRD-CF-001), API Reference (PRD-CF-010), CI/CD Best Practices (PRD-DP-010)