KCNA - Cloud Native Application Delivery

Cloud Native Application Delivery (8%)

Download PDF Version

This domain covers CI/CD, GitOps, and application deployment strategies in cloud native environments.

CI/CD Fundamentals

Continuous Integration (CI)

Automatically building and testing code changes.

Key Practices:

- Frequent code commits
- Automated builds
- Automated testing
- Fast feedback loops

Continuous Delivery (CD)

Automatically deploying code changes to staging/production.

Key Practices:

- Automated deployments
- Environment parity
- Rollback capabilities
- Release automation

CI/CD Pipeline Stages

```
┌──────────┐   ┌──────────┐   ┌──────────┐   ┌──────────┐   ┌──────────┐
│   Code   │ → │  Build   │ → │   Test   │ → │ Release  │ → │  Deploy  │
│  Commit  │   │          │   │          │   │          │   │          │
└──────────┘   └──────────┘   └──────────┘   └──────────┘   └──────────┘
```

GitOps

What is GitOps?

GitOps is a way of implementing Continuous Deployment for cloud native applications using Git as the single source of truth.

GitOps Principles

1. **Declarative**: System state is described declaratively
2. **Versioned**: Desired state is stored in Git
3. **Automated**: Changes are automatically applied
4. **Reconciled**: Software agents ensure actual state matches desired state

GitOps Workflow

```
┌─────────────┐     ┌─────────────┐     ┌─────────────┐
│  Developer  │     │     Git     │     │ Kubernetes  │
│             │     │  Repository │     │   Cluster   │
└──────┬──────┘     └──────┬──────┘     └──────┬──────┘
       │                   │                   │
       │  1. Push code     │                   │
       │──────────────────►│                   │
       │                   │                   │
       │                   │  2. Detect change │
       │                   │◄──────────────────│
       │                   │                   │
       │                   │  3. Pull & Apply  │
       │                   │──────────────────►│
       │                   │                   │
       │                   │  4. Reconcile     │
       │                   │◄──────────────────│
```

GitOps Tools

Tool	Description
Argo CD	Declarative GitOps CD for Kubernetes (CNCF)
Flux	GitOps toolkit for Kubernetes (CNCF)
Jenkins X	CI/CD for Kubernetes

Argo CD

What is Argo CD?

Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.

Key Features

- Automated deployment of applications
- Support for multiple config management tools (Kustomize, Helm, Jsonnet)
- SSO integration
- Rollback/roll-anywhere
- Health status analysis
- Web UI and CLI

Argo CD Architecture

```
┌─────────────────────────────────────────────────────┐
│                      Argo CD                        │
│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐  │
│  │ API Server  │  │ Repository  │  │Application  │  │
│  │             │  │   Server    │  │ Controller  │  │
│  └─────────────┘  └─────────────┘  └─────────────┘  │
└─────────────────────────────────────────────────────┘
         │                 │                 │
         ▼                 ▼                 ▼
   ┌─────────┐       ┌─────────┐       ┌─────────┐
   │   UI    │       │   Git   │       │   K8s   │
   │   CLI   │       │  Repos  │       │ Cluster │
   └─────────┘       └─────────┘       └─────────┘
```

Argo CD Application

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

Flux

What is Flux?

Flux is a set of continuous and progressive delivery solutions for Kubernetes.

Flux Components

Component	Description
Source Controller	Manages sources (Git, Helm, OCI)
Kustomize Controller	Reconciles Kustomize resources
Helm Controller	Manages Helm releases
Notification Controller	Handles events and alerts
Image Automation	Updates container images

Helm

What is Helm?

Helm is the package manager for Kubernetes.

Key Concepts

Concept	Description
Chart	Package of pre-configured Kubernetes resources
Release	Instance of a chart running in a cluster
Repository	Collection of charts
Values	Configuration for a chart

Helm Commands

```
# Add repository
helm repo add bitnami https://charts.bitnami.com/bitnami

# Search charts
helm search repo nginx

# Install chart
helm install my-release bitnami/nginx

# Upgrade release
helm upgrade my-release bitnami/nginx

# Rollback
helm rollback my-release 1

# Uninstall
helm uninstall my-release

# List releases
helm list
```

Helm Chart Structure

```
my-chart/
├── Chart.yaml          # Chart metadata
├── values.yaml         # Default configuration
├── templates/          # Kubernetes manifests
│   ├── deployment.yaml
│   ├── service.yaml
│   └── _helpers.tpl
└── charts/             # Dependencies
```

Deployment Strategies

Rolling Update

Gradually replaces old pods with new ones.

```
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
```

Blue-Green Deployment

Two identical environments, switch traffic between them.

```
┌─────────────┐     ┌─────────────┐
│   Blue      │     │   Green     │
│  (v1.0)     │     │  (v1.1)     │
│  [Active]   │     │  [Standby]  │
└──────┬──────┘     └──────┬──────┘
       │                   │
       └─────────┬─────────┘
                 │
          ┌──────▼──────┐
          │   Service   │
          │  (Switch)   │
          └─────────────┘
```

Canary Deployment

Gradually shift traffic to new version.

```
 Traffic Distribution:
├── v1.0: 90% ────────────────────────────────┐
└── v1.1: 10% ────┐                           │
                  │                           │
                  ▼                           ▼
            ┌─────────┐                 ┌─────────┐
            │ Canary  │                 │ Stable  │
            │  Pods   │                 │  Pods   │
            └─────────┘                 └─────────┘
```

A/B Testing

Route traffic based on specific criteria (headers, cookies).

Application Configuration

Kustomize

Kubernetes native configuration management.

```
 base/
├── deployment.yaml
├── service.yaml
└── kustomization.yaml

overlays/
├── dev/
│   └── kustomization.yaml
├── staging/
│   └── kustomization.yaml
└── prod/
    └── kustomization.yaml
 # kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - deployment.yaml
  - service.yaml
namePrefix: dev-
namespace: development
```

Key Concepts to Remember

1. **GitOps uses Git as single source of truth** for infrastructure
2. **Argo CD and Flux** are popular GitOps tools
3. **Helm** is the package manager for Kubernetes
4. **Rolling updates** are the default deployment strategy
5. **Canary deployments** allow gradual traffic shifting

Practice Questions

1. What is GitOps and what are its core principles?
2. What is the difference between Argo CD and Flux?
3. What is a Helm chart?
4. Describe the difference between blue-green and canary deployments.
5. What is Kustomize used for?

← Previous: Cloud Native Observability | Back to KCNA Overview