KCNA - Kubernetes Fundamentals

Kubernetes Fundamentals (46%)

Download PDF Version

This domain covers the core concepts of Kubernetes and represents the largest portion of the KCNA exam.

Kubernetes Architecture

Control Plane Components

Component	Description
kube-apiserver	Front-end for the Kubernetes control plane, exposes the Kubernetes API
etcd	Consistent and highly-available key-value store for cluster data
kube-scheduler	Watches for newly created Pods and assigns them to nodes
kube-controller-manager	Runs controller processes (Node, Job, EndpointSlice, ServiceAccount)
cloud-controller-manager	Embeds cloud-specific control logic

Node Components

Component	Description
kubelet	Agent that runs on each node, ensures containers are running in a Pod
kube-proxy	Network proxy that maintains network rules on nodes
Container Runtime	Software responsible for running containers (containerd, CRI-O)

Kubernetes Objects

Workload Resources

Pod

The smallest deployable unit in Kubernetes.

```yaml
 apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.21
    ports:
    - containerPort: 80
```

Deployment

Manages ReplicaSets and provides declarative updates for Pods.

```yaml
 apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21
        ports:
        - containerPort: 80
```

ReplicaSet

Maintains a stable set of replica Pods running at any given time.

StatefulSet

Manages stateful applications with unique network identifiers and persistent storage.

DaemonSet

Ensures all (or some) nodes run a copy of a Pod.

Job & CronJob

- **Job**: Creates one or more Pods and ensures they successfully terminate
- **CronJob**: Creates Jobs on a repeating schedule

Service & Networking

Service Types

Type	Description
ClusterIP	Exposes the Service on a cluster-internal IP (default)
NodePort	Exposes the Service on each Node's IP at a static port
LoadBalancer	Exposes the Service externally using a cloud provider's load balancer
ExternalName	Maps the Service to a DNS name

```yaml
 apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
  type: ClusterIP
```

Ingress

Manages external access to services, typically HTTP/HTTPS.

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

Configuration

ConfigMap

Stores non-confidential configuration data in key-value pairs.

```
 apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DATABASE_HOST: "mysql.default.svc.cluster.local"
  LOG_LEVEL: "info"
```

Secret

Stores sensitive information like passwords, tokens, or keys.

```
 apiVersion: v1
kind: Secret
metadata:
  name: app-secret
type: Opaque
data:
  password: cGFzc3dvcmQxMjM=  # base64 encoded
```

Storage

PersistentVolume (PV)

A piece of storage in the cluster provisioned by an administrator.

PersistentVolumeClaim (PVC)

A request for storage by a user.

```
 apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: standard
```

Namespaces

Namespaces provide a mechanism for isolating groups of resources within a single cluster.

Default Namespaces: - `default` - Default namespace for objects with no other namespace - `kube-system` - For objects created by the Kubernetes system - `kube-public` - Readable by all users, reserved for cluster usage - `kube-node-lease` - For lease objects associated with each node

Labels and Selectors

Labels

Key-value pairs attached to objects for identification.

```
metadata:
  labels:
    app: nginx
    environment: production
    tier: frontend
```

Selectors

Used to filter objects based on labels.

```
selector:
  matchLabels:
    app: nginx
  matchExpressions:
  - key: environment
    operator: In
    values:
    - production
    - staging
```

Kubernetes API

API Groups

Group	Resources
core (v1)	Pods, Services, ConfigMaps, Secrets, Namespaces
apps/v1	Deployments, ReplicaSets, StatefulSets, DaemonSets
batch/v1	Jobs, CronJobs
networking.k8s.io/v1	Ingress, NetworkPolicy
rbac.authorization.k8s.io/v1	Roles, ClusterRoles, RoleBindings

API Versioning

- **Alpha (v1alpha1)**: Disabled by default, may be buggy
- **Beta (v1beta1)**: Enabled by default, well-tested
- **Stable (v1)**: Production-ready, backward compatible

kubectl Basics

Common Commands

```
# Get resources
kubectl get pods
kubectl get deployments
kubectl get services
kubectl get nodes

# Describe resources
kubectl describe pod <pod-name>
kubectl describe node <node-name>

# Create resources
kubectl apply -f manifest.yaml
kubectl create deployment nginx --image=nginx

# Delete resources
kubectl delete pod <pod-name>
kubectl delete -f manifest.yaml

# Logs and debugging
kubectl logs <pod-name>
kubectl exec -it <pod-name> -- /bin/bash

# Scaling
kubectl scale deployment nginx --replicas=5
```

Key Concepts to Remember

1. **Declarative vs Imperative**: Kubernetes prefers declarative configuration
2. **Desired State**: Controllers continuously work to match actual state to desired state
3. **Self-healing**: Kubernetes automatically replaces failed containers
4. **Horizontal Scaling**: Add more Pods to handle increased load
5. **Service Discovery**: Services provide stable endpoints for Pods
6. **Rolling Updates**: Deployments can update Pods without downtime

Practice Questions

1. What is the smallest deployable unit in Kubernetes?
2. Which component is responsible for scheduling Pods to nodes?
3. What is the default Service type in Kubernetes?
4. How do you store sensitive data in Kubernetes?
5. What is the purpose of a namespace?

Back to KCNA Overview | Next: Container Orchestration →