al-nafi INTERNATIONAL COLLEGE

Al Razzaq Program - Part 2

Kubernetes and Cloud Native Associate (KCNA)

Author:

Digital Transformation Team

Kubernetes and Cloud Native Associate (KCNA)

Certification Overview:
KCNA is an entry-level certification from the **Cloud Native Computing Foundation (CNCF)** designed to validate foundational knowledge of **Kubernetes**, **cloud-native architectures**, and the **CNCF ecosystem**. It prepares candidates for hands-on certifications and real-world DevOps or cloud-native roles.

1. Kubernetes Fundamentals (46%)

This domain forms the core of the KCNA exam and provides the groundwork for Kubernetes architecture and operations.

Key Concepts:

- **Kubernetes Definition:**
  Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers.
- **Core Components:**
  - **Pods:** The smallest deployable unit in Kubernetes containing one or more containers.
  - **Nodes:** Machines (VMs or physical) that run containerized applications.
  - **Namespaces:** Logical partitions within a Kubernetes cluster.
  - **Deployments:** Manage stateless applications and control the rollout of updates.
  - **ReplicaSets:** Ensure specified numbers of pod replicas are running.
  - **Services:** Expose applications to external/internal traffic.

Control Plane Components:

- **API Server:** Central management entity that exposes the Kubernetes API.

AL NAFI E Learning Pvt ltd

- **Scheduler:** Assigns Pods to Nodes.
- **Controller Manager:** Manages controllers like ReplicationController, NodeController.
- **etcd:** Consistent and highly available key-value store used as Kubernetes' backing store.

Worker Node Components:

- **Kubelet:** Communicates with the API server, ensures container execution.
- **Kube-proxy:** Manages network communication inside and outside the cluster.
- **Container Runtime:** Software responsible for running containers (e.g., containerd, CRI-O).

Basic Kubernetes Objects and Configuration:

- **YAML:** Declarative language for Kubernetes configurations.
- **kubectl:** CLI tool for managing Kubernetes clusters.

Practical kubectl Operations:

- Deploying and scaling applications.
- Inspecting cluster health and logs.
- Debugging with logs and events.

Additional Concepts:

- **Networking:** Kubernetes uses a flat network structure for communication between pods and services.
- **State & Desired State:** Kubernetes reconciles current state with the desired state.
- **Container Lifecycle:** Pod phases (Pending, Running, Succeeded, Failed, Unknown).

Al NAFI E Learning Pvt ltd

© Al Nafi All Rights Reserved

Any printed document should be considered as an uncontrolled copy

2. Container Orchestration (22%)

This section introduces orchestration concepts and how Kubernetes improves operational efficiency.

Orchestration Overview:

- **Need for Orchestration:** Simplifies container management, provides automation for deployment, scaling, and maintenance.
- **Kubernetes vs. Docker Swarm:** Kubernetes offers richer features, broader adoption, and community support.

Benefits of Orchestration:

- **High Availability**
- **Scalability**
- **Load Balancing**
- **Rolling Updates and Rollbacks**

Resource Configuration:

- Define CPU and memory `requests` (guaranteed) and `limits` (maximum usage).

Pod Design Patterns:

- **Sidecar:** Helper container enhancing primary container (e.g., logging).
- **Ambassador:** Acts as a proxy.
- **Adapter:** Translates metrics/logs formats.

Metadata & Filtering:

- **Labels:** Key-value pairs for selection.
- **Annotations:** Arbitrary non-identifying metadata.
- **Selectors:** Query labels to manage groups of resources.

AL NAFI E Learning Pvt ltd

3. Cloud Native Architecture (16%)

Introduces the design principles and ecosystem that support scalable and maintainable cloud-native applications.

Core Definitions:

- **Cloud Native:** Architectures built with microservices, dynamic orchestration, and containerization.

Key Concepts:

- **CNCF Landscape:** Projects like Kubernetes, Prometheus, Helm, etc.
- **Microservices vs. Monoliths:** Small, independently deployable components vs. single large application.
- **12-Factor Apps:** Best practices for building SaaS applications (e.g., stateless processes, config in environment).

Design Principles:

- **Loosely Coupled Systems**
- **Declarative APIs**
- **Infrastructure as Code (IaC):** Manage infrastructure via machine-readable definition files.
- **GitOps:** Using Git as the source of truth for infrastructure and applications.

Networking Components:

- **Ingress Controllers:** Manage external access to services.
- **API Gateways:** Centralized API traffic handling.
- **Service Meshes:** Abstract communication between microservices (e.g., Istio).

AL NAFI E Learning Pvt ltd

4. Cloud Native Observability (8%)

Focuses on monitoring, logging, and tracing in cloud-native environments.

Purpose:

- Observability ensures operational insight into system health and behavior.

Core Concepts:

- **Metrics:** Numeric values describing system performance.
- **Logs:** Time-stamped records of events.
- **Tracing:** Tracks request flows across services.

Key Tools:

- **Prometheus:** Metrics collection and alerting.
- **Grafana:** Visualization and dashboards.
- **Fluentd:** Log collection and forwarding.
- **Jaeger:** Distributed tracing system.

Practices:

- Alerting for threshold breaches.
- Visual dashboards for performance metrics.

AL NAFI E Learning Pvt ltd

5. Cloud Native Application Delivery (8%)

Addresses how cloud-native applications are developed and deployed using modern tooling.

CI/CD Overview:

- **Continuous Integration (CI):** Automate code integration and testing.
- **Continuous Delivery (CD):** Automate deployment to staging or production environments.

Key Components:

- **Git:** Version control and source of truth in GitOps workflows.

- **Container Registries:** Store and distribute container images (e.g., Docker Hub, Harbor).
- **Helm:** Package manager for Kubernetes; simplifies app deployment via reusable charts.

Case Study: KCNA in Action – Building a Cloud-Native Bank

Scenario:
A FinTech startup is launching a digital bank using cloud-native technologies. They hire a KCNA-certified associate to design the initial architecture.

Solutions Implemented:

- Created Kubernetes clusters for scalable microservices.
- Implemented Prometheus & Grafana for observability.
- Applied GitOps for deployment pipelines with ArgoCD.
- Defined YAML manifests for Helm charts.
- Set up logging with Fluentd and tracing with Jaeger.

Results:

- Reduced deployment errors by 70%
- Improved time to market by 40%
- Gained real-time visibility into services

Reference:

- CNCF Case Studies
- Weaveworks GitOps
- Prometheus Documentation

AL NAFI E-Learning Pvt ltd

Summary

The KCNA certification empowers candidates with a robust foundational knowledge of Kubernetes, container orchestration, and the broader CNCF ecosystem. It covers:

- Core Kubernetes concepts (Pods, Services, Control Plane)
- Container orchestration and scheduling benefits
- Microservices and cloud-native design patterns
- Observability through metrics, logs, and tracing
- CI/CD pipelines and GitOps practices

This knowledge base is crucial for progressing to **CKA**, **CKAD**, or specialized cloud-native roles. KCNA serves as a launching pad to understand how modern applications are built, delivered, and maintained in dynamic, distributed environments.

AL NAFI E Learning Pvt ltd

© Al Nafi All Rights Reserved