KCNA Domain 1: Kubernetes Fundamentals - 20 Questions

Coverage Areas

- **Kubernetes Resources**: 8 questions
- **Kubernetes Architecture**: 4 questions
- **Kubernetes API**: 3 questions
- **Containers**: 3 questions
- **Scheduling**: 2 questions

Question 1

Category: KCNA - Kubernetes Fundamentals

What is the primary function of a Pod in Kubernetes?

A. To store configuration data for applications

B. To provide network access between different clusters

C. To run one or more containers that share storage and network

D. To manage user authentication and authorization

Correct Answer: C

A Pod is the smallest deployable unit in Kubernetes and serves as a wrapper for one or more containers. Containers within a Pod share the same network (IP address and port space) and storage volumes. They can communicate with each other using localhost and share mounted volumes. This design allows tightly coupled containers to work together as a single unit while maintaining the benefits of containerization.

Why other options are incorrect:

Option A: Configuration data storage is handled by ConfigMaps and Secrets, not Pods.

Option B: Network access between clusters is managed by cluster networking and service mesh technologies, not Pods.

Option D: User authentication and authorization are handled by Kubernetes RBAC (Role-Based Access Control) and authentication systems, not Pods.

References:

- https://kubernetes.io/docs/concepts/workloads/pods/

Question 2

Category: KCNA - Kubernetes Fundamentals

Which Kubernetes resource would you use to ensure that exactly 3 replicas of your application are always running?

A. Pod

B. Service

C. Deployment

D. ConfigMap

Correct Answer: C

A Deployment manages the desired state of your application, including the number of replicas. When you specify `replicas: 3` in a Deployment, it ensures that exactly 3 Pod instances are running at all times. If a Pod fails or is deleted, the Deployment automatically creates a new one to maintain the desired count. Deployments also provide rolling updates and rollback capabilities.

Why other options are incorrect:

Option A: A Pod is a single instance and doesn't manage multiple replicas. You would need to manually create and manage multiple Pods.

Option B: A Service provides network access to Pods but doesn't control how many Pod instances are running.

Option D: A ConfigMap stores configuration data and has no capability to manage running applications or Pod replicas.

References:

- https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Question 3

Category: KCNA - Kubernetes Fundamentals

What type of information should be stored in a ConfigMap?

A. Database passwords and API keys

B. Non-sensitive configuration data like environment variables

C. Container images and their versions

D. Network routing information

Correct Answer: B

ConfigMaps are designed to store non-sensitive configuration data in key-value pairs. This includes environment variables, configuration files, command-line arguments, and other configuration information that applications need. ConfigMaps allow you to separate configuration from application code, making applications more portable and easier to manage across different environments.

Why other options are incorrect:

Option A: Sensitive data like passwords and API keys should be stored in Secrets, not ConfigMaps, as ConfigMaps are not encrypted and are easily readable.

Option C: Container images and versions are specified in Pod specifications and managed by container registries, not stored in ConfigMaps.

Option D: Network routing is handled by Services, Ingress controllers, and network policies, not ConfigMaps.

References:

- https://kubernetes.io/docs/concepts/configuration/configmap/

Question 4

Category: KCNA - Kubernetes Fundamentals

Which component of the Kubernetes control plane stores all cluster data?

A. kube-apiserver

B. kube-scheduler

C. etcd

D. kubelet

Correct Answer: C

etcd is a distributed key-value store that serves as Kubernetes' backing store for all cluster data. It stores the configuration data, state data, and metadata for all Kubernetes objects. All cluster components read from and write to etcd through the API server. etcd provides consistency and high availability for the cluster's persistent state.

Why other options are incorrect:

Option A: The kube-apiserver exposes the Kubernetes API and validates/processes requests, but it doesn't store data - it reads from and writes to etcd.

Option B: The kube-scheduler makes decisions about Pod placement but doesn't store cluster data.

Option D: The kubelet runs on worker nodes and manages Pod lifecycle, but it doesn't store cluster-wide data.

References:

- https://kubernetes.io/docs/concepts/overview/components/

Question 5

Category: KCNA - Kubernetes Fundamentals

What is the difference between a container image and a running container?

A. There is no difference; they are the same thing

B. A container image is a template, while a running container is an active instance of that image

C. A container image runs applications, while a running container stores data

D. A container image is larger than a running container

Correct Answer: B

A container image is a lightweight, standalone package that includes everything needed to run an application: code, runtime, system tools, libraries, and settings. It serves as a template or blueprint. A running container is an active instance created from that image - it's the actual execution environment where the application runs. You can create multiple running containers from the same image.

Why other options are incorrect:

Option A: Container images and running containers are fundamentally different - one is a static template, the other is a running process.

Option C: The relationship is reversed - container images are static templates that contain applications, while running containers are the active execution of those applications.

Option D: Size comparison is not the defining difference, and typically running containers may use additional memory and temporary storage beyond the image size.

References:

- https://kubernetes.io/docs/concepts/containers/images/

Question 6

Category: KCNA - Kubernetes Fundamentals

Which command would you use to view all Pods running in the current namespace?

A. kubectl describe pods

B. kubectl get pods

C. kubectl create pods

D. kubectl delete pods

Correct Answer: B

The `kubectl get pods` command lists all Pods in the current namespace, showing their status, age, and other basic information in a tabular format. This is the standard command for viewing Pod resources and getting a quick overview of what's running in your namespace.

Why other options are incorrect:

Option A: `kubectl describe pods` provides detailed information about specific Pods but requires specifying a Pod name, and shows verbose details rather than a simple list.

Option C: `kubectl create pods` is used to create new Pods, not view existing ones.

Option D: `kubectl delete pods` is used to remove Pods, not view them.

References:

- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Question 7

Category: KCNA - Kubernetes Fundamentals

What is a Namespace used for in Kubernetes?

A. To provide persistent storage for applications

B. To create logical isolation and organization of cluster resources

C. To manage container images

D. To establish network connections between Pods

Correct Answer: B

Namespaces provide a way to organize and logically isolate cluster resources. They allow multiple teams, projects, or environments to share a single cluster while maintaining separation. Resources in one namespace are isolated from resources in other namespaces by default. Namespaces also enable applying resource quotas, access controls, and policies at the namespace level.

Why other options are incorrect:

Option A: Persistent storage is provided by PersistentVolumes and PersistentVolumeClaims, not Namespaces.

Option C: Container images are managed by container registries and referenced in Pod specifications, not managed by Namespaces.

Option D: Network connections between Pods are established by the cluster networking system and Services, not Namespaces.

References:

- https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Question 8

Category: KCNA - Kubernetes Fundamentals

What happens when you apply a YAML file using `kubectl apply -f deployment.yaml`?

A. It only creates new resources and fails if they already exist

B. It creates new resources or updates existing ones to match the YAML specification

C. It deletes all existing resources and creates new ones

D. It only validates the YAML syntax without making changes

Correct Answer: B

The `kubectl apply` command creates new resources if they don't exist, or updates existing resources to match the desired state specified in the YAML file. This is a declarative approach where you describe what you want (the desired state) and Kubernetes makes the necessary changes to achieve that state. Apply tracks the configuration and allows for future updates and management of the resources.

Why other options are incorrect:

Option A: This describes `kubectl create`, which fails if resources already exist. `kubectl apply` can handle both creation and updates.

Option C: Apply doesn't delete and recreate resources - it updates them in place when possible to minimize disruption.

Option D: Syntax validation happens automatically, but apply actually creates or updates the resources rather than just validating.

References:

- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Question 9

Category: KCNA - Kubernetes Fundamentals

Which Kubernetes object would you use to expose your application to other applications within the cluster?

A. Deployment

B. Service

C. ConfigMap

D. Secret

Correct Answer: B

A Service provides a stable network endpoint for accessing a group of Pods within the cluster. Services abstract away the individual Pod IP addresses and provide load balancing across the selected Pods. Other applications can reliably connect to your application using the Service's name and port, even as the underlying Pods are created, destroyed, or moved around the cluster.

Why other options are incorrect:

Option A: A Deployment manages the application Pods but doesn't provide network access to them.

Option C: A ConfigMap stores configuration data and doesn't provide network connectivity.

Option D: A Secret stores sensitive data and doesn't provide network access to applications.

References:

- https://kubernetes.io/docs/concepts/services-networking/service/

Question 10

Category: KCNA - Kubernetes Fundamentals

What is the role of the kube-apiserver?

A. To schedule Pods on worker nodes

B. To run containers on worker nodes

C. To provide the API interface for all cluster operations

D. To store cluster configuration data

Correct Answer: C

The kube-apiserver is the central component that exposes the Kubernetes API. All cluster operations go through the API server - whether from kubectl commands, other control plane components, or applications. It validates and processes all API requests, handles authentication and authorization, and serves as the front-end for the cluster's shared state stored in etcd.

Why other options are incorrect:

Option A: Scheduling Pods is the responsibility of the kube-scheduler, not the API server.

Option B: Running containers on worker nodes is handled by the kubelet and container runtime, not the API server.

Option D: Storing cluster data is the function of etcd. The API server reads from and writes to etcd but doesn't store the data itself.

References:

- https://kubernetes.io/docs/concepts/overview/components/

Question 11

Category: KCNA - Kubernetes Fundamentals

How do you specify which container image to use in a Pod?

A. In the metadata section of the Pod specification

B. In the spec.containers.image field of the Pod specification

C. In a separate ImageSpec resource

D. In the labels section of the Pod specification

Correct Answer: B

The container image is specified in the `spec.containers.image` field of the Pod specification. Each container in the Pod has its own image field that specifies which container image to use. The image field typically contains the image name and tag (e.g., "nginx:1.21" or "myapp:latest").

Why other options are incorrect:

Option A: The metadata section contains information like name, labels, and annotations, but not the container image specification.

Option C: There is no separate ImageSpec resource in Kubernetes. Images are specified directly in the Pod specification.

Option D: Labels are used for organizing and selecting objects, not for specifying container images.

References:

- https://kubernetes.io/docs/concepts/workloads/pods/

Question 12

Category: KCNA - Kubernetes Fundamentals

What is the purpose of labels in Kubernetes?

A. To encrypt communication between Pods

B. To organize and select Kubernetes objects

C. To store application configuration data

D. To define resource limits for containers

Correct Answer: B

Labels are key-value pairs attached to Kubernetes objects (like Pods, Services, Deployments) that are used to organize and select groups of objects. They enable you to categorize resources and perform operations on groups of objects. For example, you can select all Pods with the label "app=frontend" or all resources with "environment=production". Labels are fundamental to how Controllers and Services identify which objects they should manage.

Why other options are incorrect:

Option A: Labels don't provide encryption. Network security is handled by network policies and other security mechanisms.

Option C: Configuration data is stored in ConfigMaps and Secrets, not in labels. Labels are metadata for organization and selection.

Option D: Resource limits are defined in the resources section of container specifications, not in labels.

References:

- https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

Question 13

Category: KCNA - Kubernetes Fundamentals

Which component runs on every worker node in a Kubernetes cluster?

A. etcd

B. kube-apiserver

C. kubelet

D. kube-scheduler

Correct Answer: C

The kubelet is the primary node agent that runs on every worker node in a Kubernetes cluster. It's responsible for managing the lifecycle of Pods on that node, including pulling container images, starting and stopping containers, monitoring container health, and reporting status back to the control plane. The kubelet ensures that containers described in Pod specifications are running and healthy.

Why other options are incorrect:

Option A: etcd typically runs only on control plane nodes as part of the cluster's data store, not on every worker node.

Option B: The kube-apiserver runs on control plane nodes to provide the API interface, not on every worker node.

Option D: The kube-scheduler runs on control plane nodes to make Pod placement decisions, not on worker nodes.

References:

- https://kubernetes.io/docs/concepts/overview/components/

Question 14

Category: KCNA - Kubernetes Fundamentals

What is a ReplicaSet responsible for?

A. Storing persistent data for applications

B. Providing network access to Pods

C. Ensuring a specified number of Pod replicas are running

D. Managing user permissions in the cluster

Correct Answer: C

A ReplicaSet ensures that a specified number of identical Pod replicas are running at any given time. If Pods fail, are deleted, or if there are too few running, the ReplicaSet creates new Pods. If there are too many Pods running, it deletes the excess ones. ReplicaSets use label selectors to identify which Pods they manage and continuously monitor to maintain the desired state.

Why other options are incorrect:

Option A: Persistent data storage is managed by PersistentVolumes and storage systems, not ReplicaSets.

Option B: Network access to Pods is provided by Services, not ReplicaSets.

Option D: User permissions are managed by RBAC (Role-Based Access Control) components, not ReplicaSets.

References:

- https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

Question 15

Category: KCNA - Kubernetes Fundamentals

What does it mean when a Pod is in "Pending" status?

A. The Pod is running successfully

B. The Pod has been accepted by the cluster but hasn't been scheduled to a node yet

C. The Pod has completed its task and is shutting down

D. The Pod has failed and cannot be restarted

Correct Answer: B

A Pod in "Pending" status means that the Pod has been accepted by the Kubernetes cluster and stored in etcd, but it hasn't been scheduled to run on a node yet. This could be because the scheduler is still working to find a suitable node, or because there are no nodes available that meet the Pod's requirements (such as resource requests, node selectors, or affinity rules).

Why other options are incorrect:

Option A: A successfully running Pod would have a status of "Running", not "Pending".

Option C: A Pod that has completed its task would have a status of "Succeeded" or "Completed", not "Pending".

Option D: A failed Pod would have a status of "Failed" or "CrashLoopBackOff", not "Pending".

References:

- https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/

Question 16

Category: KCNA - Kubernetes Fundamentals

How do containers within the same Pod communicate with each other?

A. Through the cluster DNS system

B. Using localhost (127.0.0.1) since they share the same network

C. Through a separate Service resource

D. They cannot communicate directly

Correct Answer: B

Containers within the same Pod share the same network namespace, which means they share the same IP address and port space. They can communicate with each other using localhost (127.0.0.1) on different ports. For example, if one container listens on port 8080 and another on port 9090, they can reach each other at localhost:8080 and localhost:9090 respectively.

Why other options are incorrect:

Option A: Cluster DNS is used for communication between different Pods or Services, not between containers within the same Pod.

Option C: Services are used to expose Pods to other Pods or external traffic, not for intra-Pod communication.

Option D: Containers within the same Pod can communicate directly through the shared network namespace.

References:

- https://kubernetes.io/docs/concepts/workloads/pods/

Question 17

Category: KCNA - Kubernetes Fundamentals

What is the primary difference between `kubectl create` and `kubectl apply`?

A. create is for Pods only, apply is for all resources

B. create fails if the resource exists, apply creates or updates the resource

C. create is faster than apply

D. There is no difference between them

Correct Answer: B

`kubectl create` is an imperative command that creates a new resource and fails if a resource with the same name already exists. `kubectl apply` is a declarative command that creates a resource if it doesn't exist or updates it if it does exist, based on the desired state specified in the configuration file. Apply also tracks the applied configuration for future management operations.

Why other options are incorrect:

Option A: Both commands can work with all types of Kubernetes resources, not just specific ones.

Option C: Performance differences are not the primary distinction between these commands. The key difference is their behavior with existing resources.

Option D: There are significant differences in how these commands handle existing resources and configuration management.

References:

- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Question 18

Category: KCNA - Kubernetes Fundamentals

What information is stored in a Secret?

A. Public configuration data like application settings

B. Sensitive data like passwords, tokens, and keys

C. Container images and their metadata

D. Network routing rules for the cluster

Correct Answer: B

Secrets are designed to store and manage sensitive information such as passwords, OAuth tokens, SSH keys, TLS certificates, and API keys. Secrets are base64 encoded and can be encrypted at rest (depending on cluster configuration). They provide a more secure way to handle sensitive data compared to putting it directly in Pod specifications or container images.

Why other options are incorrect:

Option A: Public configuration data should be stored in ConfigMaps, not Secrets. Secrets are specifically for sensitive information.

Option C: Container images and metadata are managed by container registries and the container runtime, not stored in Secrets.

Option D: Network routing rules are managed by Services, Ingress controllers, and network policies, not stored in Secrets.

References:

- https://kubernetes.io/docs/concepts/configuration/secret/

Question 19

Category: KCNA - Kubernetes Fundamentals

When you delete a Deployment, what happens to the Pods it manages?

A. The Pods continue running independently

B. The Pods are also deleted along with the Deployment

C. The Pods are converted to standalone Pods

D. Only the Pod metadata is deleted, containers keep running

Correct Answer: B

When you delete a Deployment, Kubernetes also deletes the ReplicaSet that the Deployment manages, which in turn deletes all the Pods that belong to that ReplicaSet. This is because the Pods are owned by the ReplicaSet (through owner references), and Kubernetes automatically cleans up owned resources when the owner is deleted. This cascading deletion ensures that no orphaned resources are left behind.

Why other options are incorrect:

Option A: The Pods don't continue running independently. They are deleted as part of the cascading deletion process.

Option C: The Pods are not converted to standalone Pods. They are completely removed from the cluster.

Option D: The entire Pod objects are deleted, including their containers and all associated resources.

References:

- https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Question 20

Category: KCNA - Kubernetes Fundamentals

What is the purpose of the `selector` field in a Service?

A. To choose which namespace the Service operates in

B. To determine which Pods the Service should route traffic to

C. To select which ports the Service should expose

D. To specify the Service type (ClusterIP, NodePort, etc.)

Correct Answer: B

The `selector` field in a Service specification determines which Pods the Service should route traffic to. The selector uses label matching to identify target Pods - any Pod that has labels matching the Service's selector will receive traffic from the Service. This creates a dynamic relationship where Pods can be added or removed from the Service simply by changing their labels.

Why other options are incorrect:

Option A: The namespace is determined by where the Service is created, not by the selector field.

Option C: Ports are specified in the `ports` field of the Service specification, not in the selector.

Option D: The Service type is specified in the `type` field, not in the selector field.

References:

- https://kubernetes.io/docs/concepts/services-networking/service/
