KCNA Domain 2: Container Orchestration - 20 Questions

Coverage Areas

- **Container Orchestration Fundamentals**: 5 questions
- **Runtime**: 4 questions
- **Security**: 4 questions
- **Networking**: 4 questions
- **Service Mesh**: 2 questions
- **Storage**: 1 question

Question 1

Category: KCNA - Container Orchestration

What is the primary benefit of container orchestration?

A. To create container images

B. To automatically manage the deployment, scaling, and operation of containerized applications

C. To store container images in a registry

D. To write application code

Correct Answer: B

Container orchestration provides automated management of containerized applications across a cluster of machines. This includes deploying applications, scaling them up or down based on demand, handling failures by restarting or rescheduling containers, managing updates and rollbacks, and ensuring applications run reliably. Orchestration platforms like Kubernetes eliminate the need for manual container management at scale.

Why other options are incorrect:

Option A: Creating container images is done by build tools like Docker, not orchestration platforms.

Option C: Storing container images is the function of container registries like Docker Hub or Harbor, not orchestration.

Option D: Writing application code is a development activity, not a function of container orchestration.

References:

- https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Question 2

Category: KCNA - Container Orchestration

What is the purpose of a taint on a Kubernetes node?

A. To mark a node as ready for scheduling

B. To prevent certain Pods from being scheduled on the node unless they have matching tolerations

C. To increase the node's CPU capacity

D. To provide network connectivity to the node

Correct Answer: B

Taints are applied to nodes to repel Pods from being scheduled on them unless the Pods have matching tolerations. This allows nodes to control which workloads can run on them. For example, you might taint nodes that are dedicated for specific applications, have special hardware, or are undergoing maintenance. Only Pods with the corresponding toleration can "tolerate" the taint and be scheduled on that node.

Why other options are incorrect:

Option A: Taints actually prevent scheduling rather than mark nodes as ready. A node without taints is generally ready for any Pod.

Option C: Taints don't affect the node's actual CPU capacity - they only control Pod scheduling based on toleration rules.

Option D: Network connectivity is handled by the cluster networking system, not by taints.

References:

- https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

Question 3

Category: KCNA - Container Orchestration

Which container runtime is commonly used with Kubernetes?

A. containerd

B. Apache

C. MySQL

D. Nginx

Correct Answer: A

containerd is a popular container runtime that Kubernetes can use to run containers. It's a lightweight, high-performance container runtime that handles the low-level operations of running containers, including image management, container lifecycle management, and resource isolation. Other container runtimes compatible with Kubernetes include CRI-O and Docker (through dockershim, though this is deprecated).

Why other options are incorrect:

Option B: Apache is a web server, not a container runtime.

Option C: MySQL is a database management system, not a container runtime.

Option D: Nginx is a web server and reverse proxy, not a container runtime.

References:

- https://kubernetes.io/docs/setup/production-environment/container-runtimes/

Question 4

Category: KCNA - Container Orchestration

What happens when a Pod with a toleration is scheduled on a node with a matching taint?

A. The Pod is immediately deleted

B. The Pod is scheduled normally and runs on the node

C. The taint is removed from the node

D. The Pod fails to start

Correct Answer: B

When a Pod has a toleration that matches a node's taint, the Pod can be scheduled on that node and will run normally. The toleration allows the Pod to "tolerate" the taint, effectively bypassing the scheduling restriction that the taint would normally impose. The taint remains on the node, and the Pod operates as it would on any other node.

Why other options are incorrect:

Option A: Matching tolerations allow Pods to run successfully, they don't cause deletion.

Option C: Taints are not removed when Pods with matching tolerations are scheduled. The taint remains to continue controlling future scheduling decisions.

Option D: A matching toleration ensures the Pod can start and run successfully on the tainted node.

References:

- https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

Question 5

Category: KCNA - Container Orchestration

What is the Container Runtime Interface (CRI)?

A. A network protocol for container communication

B. A standard API that allows Kubernetes to work with different container runtimes

C. A storage system for container images

D. A security framework for containers

Correct Answer: B

The Container Runtime Interface (CRI) is a standard API that allows Kubernetes to work with different container runtimes without needing to know the specific details of each runtime. This abstraction enables Kubernetes to support multiple container runtimes like containerd, CRI-O, and others through a consistent interface. The CRI defines how the kubelet communicates with container runtimes for operations like creating, starting, and stopping containers.

Why other options are incorrect:

Option A: CRI is an API specification, not a network protocol for container communication.

Option C: Container image storage is handled by container registries and image layers, not by CRI.

Option D: While container runtimes handle some security aspects, CRI itself is an interface specification, not a security framework.

References:

- https://kubernetes.io/docs/concepts/architecture/cri/

Question 6

Category: KCNA - Container Orchestration

What is a DaemonSet used for in Kubernetes?

A. To run exactly one Pod on every node in the cluster

B. To store configuration data for applications

C. To provide network access between Pods

D. To manage user authentication

Correct Answer: A

A DaemonSet ensures that all (or some) nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them automatically. As nodes are removed, those Pods are garbage collected. DaemonSets are typically used for cluster-wide services like log collection agents, monitoring agents, or network plugins that need to run on every node.

Why other options are incorrect:

Option B: Configuration data storage is handled by ConfigMaps and Secrets, not DaemonSets.

Option C: Network access between Pods is provided by Services and the cluster networking system, not DaemonSets.

Option D: User authentication is managed by authentication systems and RBAC, not DaemonSets.

References:

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

Question 7

Category: KCNA - Container Orchestration

What is the purpose of a Security Context in Kubernetes?

A. To store encrypted passwords

B. To define privilege and access control settings for Pods and containers

C. To create network connections between containers

D. To schedule Pods on specific nodes

Correct Answer: B

A Security Context defines privilege and access control settings for a Pod or container, including user ID, group ID, filesystem permissions, capabilities, and security profiles like SELinux or AppArmor. Security contexts help implement the principle of least privilege by controlling what actions containers can perform and what resources they can access at the operating system level.

Why other options are incorrect:

Option A: Encrypted passwords are stored in Secrets, not Security Contexts.

Option C: Network connections are managed by Services and networking components, not Security Contexts.

Option D: Pod scheduling is handled by the scheduler based on node selectors, affinity rules, and resource requirements, not Security Contexts.

References:

- https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Question 8

Category: KCNA - Container Orchestration

Which field in a Security Context is used to specify the user ID that containers should run as?

A. fsGroup

B. runAsUser

C. capabilities

D. seLinuxOptions

Correct Answer: B

The `runAsUser` field in a Security Context specifies the user ID (UID) that the container processes should run as. This is important for security because it determines what files the container can access and what operations it can perform on the host system. Running containers as non-root users (UID other than 0) is a security best practice.

Why other options are incorrect:

Option A: `fsGroup` defines the group ID for volume ownership and permissions, not the user ID for running processes.

Option C: `capabilities` controls Linux capabilities that are added or dropped from the container, not the user ID.

Option D: `seLinuxOptions` configures SELinux labels for the container, not the user ID.

References:

- https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Question 9

Category: KCNA - Container Orchestration

What is a Service Mesh?

A. A type of Kubernetes Service

B. An infrastructure layer that handles service-to-service communication

C. A container runtime for Kubernetes

D. A storage system for container data

Correct Answer: B

A Service Mesh is a dedicated infrastructure layer that handles service-to-service communication within a microservices architecture. It provides features like traffic management, security (mutual TLS), observability (metrics, logs, traces), and policy enforcement without requiring changes to application code. Popular service mesh implementations include Istio, Linkerd, and Consul Connect.

Why other options are incorrect:

Option A: While service meshes work with Kubernetes Services, they are not a type of Service but rather an additional infrastructure layer.

Option C: Service meshes are not container runtimes. They run alongside container runtimes to manage communication between services.

Option D: Service meshes don't provide storage for container data. They focus on communication, security, and observability.

References:

- https://kubernetes.io/docs/concepts/services-networking/service/
- https://istio.io/latest/docs/concepts/what-is-istio/

Question 10

Category: KCNA - Container Orchestration

What type of network communication does a Service Mesh typically secure?

A. Communication between users and applications

B. Communication between services within the cluster

C. Communication between clusters

D. Communication between containers and storage

Correct Answer: B

Service meshes primarily secure communication between services (microservices) within the cluster. They provide mutual TLS (mTLS) encryption, authentication, and authorization for service-to-service communication. This ensures that internal communications between different parts of your application are encrypted and properly authenticated, which is crucial in a microservices architecture.

Why other options are incorrect:

Option A: User-to-application communication is typically secured by ingress controllers, load balancers, and application-level security, not primarily by service meshes.

Option C: Inter-cluster communication is handled by cluster federation and multi-cluster networking solutions, not typically by service meshes.

Option D: Container-to-storage communication is secured by storage encryption and access controls, not by service meshes.

References:

- https://istio.io/latest/docs/concepts/security/

Question 11

Category: KCNA - Container Orchestration

What is the primary function of kube-proxy?

A. To schedule Pods on nodes

B. To manage network communication and load balancing for Services

C. To store cluster configuration data

D. To run containers on worker nodes

Correct Answer: B

kube-proxy is a network proxy that runs on each node and maintains network rules for Services. It handles the routing of traffic from Service IPs to the actual Pod IPs, implementing load balancing across multiple Pod endpoints. kube-proxy ensures that when you connect to a Service, your traffic gets routed to one of the healthy Pods backing that Service.

Why other options are incorrect:

Option A: Pod scheduling is the responsibility of the kube-scheduler, not kube-proxy.

Option C: Cluster configuration data is stored in etcd, not managed by kube-proxy.

Option D: Running containers is handled by the kubelet and container runtime, not kube-proxy.

References:

- https://kubernetes.io/docs/concepts/overview/components/
- https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/

Question 12

Category: KCNA - Container Orchestration

Which type of Service provides each node with a static port for external access?

A. ClusterIP

B. NodePort

C. LoadBalancer

D. ExternalName

Correct Answer: B

A NodePort Service allocates a static port (in the range 30000-32767 by default) on every node in the cluster. External traffic can reach the Service by connecting to any node's IP address on the allocated port. The traffic is then routed to the appropriate Pods. This provides a simple way to expose Services externally without requiring cloud provider integration.

Why other options are incorrect:

Option A: ClusterIP only provides internal cluster access and doesn't expose any external ports on nodes.

Option C: LoadBalancer provides external access through a cloud provider's load balancer, not through static ports on nodes.

Option D: ExternalName maps a Service to an external DNS name and doesn't provide port access on nodes.

References:

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

Question 13

Category: KCNA - Container Orchestration

What is a Network Policy in Kubernetes?

A. A rule that controls how Pods communicate with each other and external endpoints

B. A configuration for container runtime networking

C. A setting that defines node network interfaces

D. A specification for Service load balancing

Correct Answer: A

A Network Policy is a specification that controls how groups of Pods are allowed to communicate with each other and with other network endpoints. Network policies act as a firewall for Pods, allowing you to define rules for ingress (incoming) and egress (outgoing) traffic based on labels, namespaces, and IP addresses. This enables micro-segmentation and helps implement security controls in your cluster.

Why other options are incorrect:

Option B: Container runtime networking is configured separately from Network Policies, which operate at the Pod level.

Option C: Node network interfaces are configured at the infrastructure level, not controlled by Network Policies.

Option D: Service load balancing is handled by kube-proxy and Service configurations, not Network Policies.

References:

- https://kubernetes.io/docs/concepts/services-networking/network-policies/

Question 14

Category: KCNA - Container Orchestration

What happens if you don't specify a Security Context for a Pod?

A. The Pod will fail to start

B. The Pod will use default security settings, which may not be secure

C. The Pod will automatically run with maximum security

D. Kubernetes will prompt you to specify one

Correct Answer: B

If you don't specify a Security Context, the Pod will use default security settings provided by the container runtime and the node's configuration. These defaults may not be secure - for example, containers might run as root (UID 0) by default, which violates the principle of least privilege. It's a best practice to explicitly define Security Contexts to ensure containers run with appropriate security constraints.

Why other options are incorrect:

Option A: Pods can start without explicit Security Contexts, though they may not be secure.

Option C: Default settings are typically permissive rather than maximally secure, which is why explicit Security Contexts are recommended.

Option D: Kubernetes doesn't prompt for Security Context specification - it uses defaults if none are provided.

References:

- https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Question 15

Category: KCNA - Container Orchestration

What is the purpose of image pull secrets?

A. To encrypt container images

B. To provide authentication credentials for pulling images from private registries

C. To store application passwords

D. To configure network access for containers

Correct Answer: B

Image pull secrets provide authentication credentials needed to pull container images from private container registries. When your Pods reference images stored in private registries (like private Docker Hub repositories, AWS ECR, or corporate registries), Kubernetes needs credentials to authenticate and download those images. Image pull secrets store these credentials securely and can be referenced in Pod specifications.

Why other options are incorrect:

Option A: Image pull secrets don't encrypt images themselves - they provide access credentials. Image encryption is handled separately.

Option C: Application passwords are stored in regular Secrets, not specifically image pull secrets.

Option D: Network access configuration is handled by Network Policies and other networking components, not image pull secrets.

References:

- https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/

Question 16

Category: KCNA - Container Orchestration

Which scheduling constraint would you use to ensure a Pod runs on a node with specific hardware?

A. Tolerations

B. Node Selector

C. Image Pull Policy

D. Security Context

Correct Answer: B

Node Selector is the simplest way to constrain Pods to run on nodes with specific characteristics, including hardware requirements. You can label nodes with hardware-specific labels (like "disk=ssd" or "gpu=nvidia") and then use nodeSelector in your Pod specification to ensure Pods only get scheduled on nodes with those labels. This is useful for workloads that need specific hardware capabilities.

Why other options are incorrect:

Option A: Tolerations allow Pods to be scheduled on tainted nodes, but they don't select nodes based on hardware characteristics.

Option C: Image Pull Policy controls how container images are downloaded, not where Pods are scheduled.

Option D: Security Context controls security settings for containers, not scheduling decisions.

References:

- https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

Question 17

Category: KCNA - Container Orchestration

What is the default restart policy for Pods in Kubernetes?

A. Never

B. OnFailure

C. Always

D. RestartOnUpdate

Correct Answer: C

The default restart policy for Pods in Kubernetes is "Always". This means that containers within the Pod will be restarted whenever they exit, regardless of the exit code (whether they completed successfully or failed). This policy is appropriate for long-running services that should always be available. Other restart policies include "OnFailure" (restart only on non-zero exit codes) and "Never" (never restart).

Why other options are incorrect:

Option A: "Never" is a valid restart policy but not the default. It's used for jobs that should run once and not restart.

Option B: "OnFailure" is a valid restart policy but not the default. It's often used for batch jobs that should only restart on failure.

Option D: "RestartOnUpdate" is not a valid Kubernetes restart policy.

References:

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

Question 18

Category: KCNA - Container Orchestration

What is container image immutability?

A. Images cannot be deleted from registries

B. Images cannot be modified after they are built

C. Images cannot be pulled from registries

D. Images cannot be used in multiple Pods

Correct Answer: B

Container image immutability means that once a container image is built and tagged, it cannot be modified. If you need to make changes, you must build a new image with a new tag. This immutability provides consistency and predictability - you can be confident that an image tagged "myapp:1.2.3" will always contain exactly the same software and configuration, no matter when or where you deploy it.

Why other options are incorrect:

Option A: Images can be deleted from registries (though this should be done carefully to avoid breaking deployments).

Option C: Images are designed to be pulled from registries - that's their primary purpose.

Option D: Images are specifically designed to be reused across multiple Pods and deployments.

References:

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

Question 19

Category: KCNA - Container Orchestration

What is the purpose of a PersistentVolume (PV) in container orchestration?

A. To provide temporary storage that gets deleted when containers stop

B. To provide durable storage that persists beyond container and Pod lifecycles

C. To store container images

D. To manage network connections

Correct Answer: B

A PersistentVolume (PV) provides durable storage that persists beyond the lifecycle of individual containers and Pods. This is essential for stateful applications that need to preserve data even when containers are restarted, rescheduled, or updated. PVs are backed by various storage systems (NFS, cloud storage, local storage) and can be claimed by Pods through PersistentVolumeClaims (PVCs).

Why other options are incorrect:

Option A: This describes temporary/ephemeral storage like emptyDir volumes, not PersistentVolumes.

Option C: Container images are stored in container registries, not in PersistentVolumes.

Option D: Network connections are managed by Services and networking components, not storage volumes.

References:

- https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Question 20

Category: KCNA - Container Orchestration

How does a service mesh typically implement traffic encryption between services?

A. Using Network Policies

B. Using Secrets to store certificates

C. Using mutual TLS (mTLS) with automatic certificate management

D. Using container image encryption

Correct Answer: C

Service meshes typically implement traffic encryption using mutual TLS (mTLS) with automatic certificate management. In mTLS, both the client and server authenticate each other using certificates, and all communication is encrypted. The service mesh automatically handles certificate provisioning, rotation, and management, making it transparent to application developers while ensuring all service-to-service communication is secure.

Why other options are incorrect:

Option A: Network Policies control traffic flow but don't provide encryption - they're more like firewall rules.

Option B: While Secrets can store certificates, service meshes typically handle certificate management automatically without requiring manual Secret management.

Option D: Container image encryption secures images at rest, but doesn't encrypt traffic between running services.

References:

- https://istio.io/latest/docs/concepts/security/
- https://kubernetes.io/docs/concepts/services-networking/

Quality Assurance Summary

✅ **Domain Focus**: All 20 questions focused exclusively on Container Orchestration ✅ **Coverage Areas**: Balanced coverage across all sub-areas (Fundamentals, Runtime, Security, Networking, Service Mesh, Storage) ✅ **Question Level**: Foundational-level complexity with simple, direct questions ✅ **Question Format**: Multiple choice with single correct answers ✅ **Answer Options**: Plausible distractors of similar length ✅ **Explanations**: Clear explanations appropriate for foundational level learners ✅ **References**: Official Kubernetes and related documentation links ✅ **Content Accuracy**: All technical content verified against official documentation