SET A

1. Kubernetes Service Types

Question: You want to expose an application running inside your Kubernetes cluster to external users using a stable external IP. Which Kubernetes Service type should you use?

A. **ClusterIP**
B. **NodePort**
C. **LoadBalancer**
D. **ExternalName**

Answer: C

Explanation:

- **LoadBalancer (C)** provisions an external load balancer (if supported by the cloud provider) and assigns a public IP for accessing the service.
- **ClusterIP (A)** is the default and allows only internal access.
- **NodePort (B)** exposes the service on a static port on each node, but without an external load balancer.
- **ExternalName (D)** maps a service to an external DNS name, but does not expose the service via an IP.

---

2. Kubernetes Pod Scheduling Constraints

Question: You need to ensure that a specific pod always runs on a node with a GPU for machine learning workloads. Which Kubernetes feature should you use?

A. **Taints and Tolerations**
B. **NodeSelector**
C. **Pod Disruption Budget (PDB)**
D. **Pod Priority Classes**

Answer: B

Explanation:

- **NodeSelector (B)** allows you to specify a node with specific labels (e.g., `gpu=true`) to schedule pods on nodes with GPUs.

- **Taints and Tolerations (A)** prevent workloads from being scheduled on specific nodes unless explicitly tolerated.
- **PDB (C)** is used to define how many replicas of a pod can be disrupted during maintenance, not for scheduling.
- **Pod Priority Classes (D)** influence scheduling but do not restrict a pod to specific nodes.

---

3. Kubernetes Networking Model

Question: What is true about the Kubernetes networking model?

A. **Each pod gets a unique IP address and can communicate with other pods without NAT.**
B. **Pods communicate using host machine IPs, and port mapping is required.**
C. **Containers within the same pod cannot talk to each other directly.**
D. **All network policies are enabled by default for inter-pod communication.**

Answer: A

Explanation:

- **Kubernetes assigns a unique IP to each pod (A)**, and all pods in a cluster can communicate without NAT.
- **Host machine IPs (B)** are used only in certain cases like NodePort services, not for pod-to-pod communication.
- **Containers inside the same pod (C)** share the same network namespace and can communicate via `localhost`.
- **Network policies (D)** are not enforced by default; they must be explicitly created.

---

4. Kubernetes Rolling Update vs. Recreate

Question: What happens during a **rolling update** of a Deployment in Kubernetes?

A. **All existing pods are terminated immediately and new ones are created.**
B. **Pods are replaced one at a time to ensure no downtime.**
C. **Only some replicas are restarted while others remain running.**
D. **A new Deployment is created, and the old Deployment is deleted.**

Answer: B

Explanation:

- **Rolling updates (B)** ensure availability by replacing pods incrementally.
- **Recreate strategy (A)** would terminate all pods first, leading to downtime.
- **Option C** is partially correct but doesn't explicitly state a controlled, sequential rollout.
- **Option D** is incorrect—Deployments are updated, not replaced.

---

5. Kubernetes ConfigMap vs Secret

Question: How do ConfigMaps differ from Secrets in Kubernetes?

A. **ConfigMaps are stored as base64-encoded data, whereas Secrets store plain text.**
B. **Secrets are stored securely and can be encrypted, while ConfigMaps store non-sensitive configuration.**
C. **Both ConfigMaps and Secrets require pods to restart for changes to take effect.**
D. **ConfigMaps can only be mounted as environment variables, while Secrets cannot.**

Answer: B

Explanation:

- **Secrets are encoded and can be encrypted for security (B)**, whereas ConfigMaps hold non-sensitive configuration.
- **ConfigMaps store plain text (A is incorrect)**, while Secrets are base64-encoded by default.
- **Option C** is incorrect—pods don't always need a restart; some applications can reload configurations dynamically.
- **Option D** is incorrect—both can be used as environment variables or mounted as files.

---

6. Kubernetes Persistent Volume Access Modes

Question: Which Kubernetes **Persistent Volume (PV) access mode** allows multiple nodes to **read and write** to the same volume at the same time?

A. **ReadWriteOnce (RWO)**
B. **ReadOnlyMany (ROX)**
C. **ReadWriteMany (RWX)**
D. **ReadWriteOnly (RWO)**

Answer: C

Explanation:

- **ReadWriteMany (RWX) (C)** allows multiple nodes to read and write simultaneously.
- **ReadWriteOnce (RWO) (A)** means only one node can mount the volume in read/write mode.
- **ReadOnlyMany (ROX) (B)** allows multiple nodes to mount but only for read access.
- **D is an incorrect option (no such mode as ReadWriteOnly).**

---

7. Kubernetes Horizontal Pod Autoscaler (HPA)

Question: You want to **automatically scale** the number of pods in a deployment based on CPU utilization. Which Kubernetes feature should you use?

A. **Cluster Autoscaler**
B. **Vertical Pod Autoscaler (VPA)**
C. **Horizontal Pod Autoscaler (HPA)**
D. **PodDisruptionBudget (PDB)**

Answer: C

Explanation:

- **HPA (C)** automatically scales the number of pods based on resource utilization.
- **Cluster Autoscaler (A)** scales nodes, not pods.
- **Vertical Pod Autoscaler (B)** adjusts pod resource requests but does not scale replicas.
- **PodDisruptionBudget (D)** controls disruption limits but does not handle autoscaling.

---

8. Kubernetes PodSecurityPolicy

Question: You need to **prevent pods from running as root** in your Kubernetes cluster. Which feature should you use?

A. **PodDisruptionBudget (PDB)**
B. **SecurityContext in pod definition**
C. **RBAC (Role-Based Access Control)**
D. **ServiceAccount**

Answer: B

Explanation:

- **SecurityContext (B)** can enforce that pods do not run as root.
- **PodDisruptionBudget (A)** controls pod eviction, not security settings.

- **RBAC (C)** manages permissions but does not enforce security settings on pod execution.
- **ServiceAccount (D)** assigns identities to pods but does not enforce runtime security.

---

9. Kubernetes Init Containers

Question: What is the primary purpose of **Init Containers** in Kubernetes?

A. **To provide long-running background services within a pod.**
B. **To execute setup tasks before the main containers start.**
C. **To replace the main application container in case of failure.**
D. **To handle persistent storage for the pod.**

Answer: B

Explanation:

- **Init Containers (B)** run before the main application containers to perform initialization tasks like setting up configuration files.
- **They do not run as background services (A)**—they terminate after completion.
- **They do not replace failed containers (C).**
- **They do not directly handle storage (D).**

---

10. Kubernetes DaemonSet vs Deployment

Question: You need to ensure that a specific pod is deployed to **every node in the cluster**. Which Kubernetes resource should you use?

A. **Deployment**
B. **StatefulSet**
C. **DaemonSet**
D. **ReplicaSet**

Answer: C

Explanation:

- **DaemonSets (C)** ensure that a pod runs on every node in the cluster (e.g., log collectors, monitoring agents).
- **Deployments (A)** manage replicas but do not guarantee placement on every node.
- **StatefulSets (B)** are used for stateful applications with stable network identities.

- **ReplicaSets (D)** ensure a specific number of pod replicas but do not control node placement.

SET B

1. Kubernetes Network Policy

Question: You want to ensure that only a specific pod in **namespace A** can access a pod running in **namespace B**. Which Kubernetes feature should you use?

A. **Ingress Controller**
B. **RBAC (Role-Based Access Control)**
C. **NetworkPolicy**
D. **ServiceAccount**

Answer: C

Explanation:

- **NetworkPolicy (C)** controls traffic flow between pods at the network level and can restrict access between namespaces.
- **Ingress Controller (A)** handles external traffic, not inter-pod communication.
- **RBAC (B)** manages API access, not network security.
- **ServiceAccount (D)** is used for API authentication, not network policies.

---

2. Kubernetes StatefulSets

Question: What is the key difference between a **StatefulSet** and a **Deployment** in Kubernetes?

A. **StatefulSets use a unique identity and persistent storage for each pod.**
B. **Deployments always ensure that pods run on different nodes.**
C. **StatefulSets only support one replica per pod definition.**
D. **Deployments cannot automatically scale replicas.**

Answer: A

Explanation:

- **StatefulSets (A)** maintain a stable identity and persistent storage for each pod, making them ideal for databases and applications that require ordered deployment.
- **Deployments (B)** do not guarantee pods will be placed on different nodes.
- **StatefulSets allow multiple replicas (C is incorrect).**
- **Deployments (D) support autoscaling with Horizontal Pod Autoscaler.**

---

3. Kubernetes Readiness vs. Liveness Probes

Question: What is the main purpose of a **readiness probe** in a Kubernetes pod?

A. **To check if a pod is running and ready to receive traffic.**
B. **To restart the pod if it becomes unresponsive.**
C. **To update DNS entries for the pod.**
D. **To determine if a pod has been scheduled on a node.**

Answer: A

Explanation:

- **Readiness probes (A)** indicate if a pod is ready to serve traffic.
- **Liveness probes (B)** determine if a pod is healthy or needs a restart.
- **DNS updates (C) happen automatically in Kubernetes.**
- **Pod scheduling (D) is determined by the scheduler, not probes.**

---

4. Kubernetes API Resource Hierarchy

Question: Which Kubernetes resource is the **parent** of a ReplicaSet?

A. **Pod**
B. **Deployment**
C. **DaemonSet**
D. **ConfigMap**

Answer: B

Explanation:

- **Deployments (B)** create and manage **ReplicaSets**, which in turn manage **pods**.
- **DaemonSets (C)** ensure that a pod runs on every node.
- **ConfigMaps (D)** store configuration but are not part of pod scheduling.

---

5. Kubernetes Job vs. CronJob

Question: You need to run a **one-time** data migration task in Kubernetes. Which resource should you use?

A. **Job**
B. **CronJob**
C. **StatefulSet**
D. **Deployment**

Answer: A

Explanation:

- **Jobs (A)** are for one-time batch tasks.
- **CronJobs (B)** schedule recurring jobs.
- **StatefulSets (C)** manage persistent applications.
- **Deployments (D)** are for long-running applications.

---

6. Kubernetes Service Discovery

Question: How do Kubernetes pods discover and communicate with services inside a cluster?

A. **Using built-in DNS resolution for Service names.**
B. **Manually assigning static IP addresses to pods.**
C. **By using host-based routing configured in ConfigMaps.**
D. **By exposing each pod with a NodePort.**

Answer: A

Explanation:

- Kubernetes **automatically assigns DNS names** to services, allowing pods to resolve service names without needing static IPs.
- **Static IPs (B)** are not used since pods are ephemeral.
- **ConfigMaps (C)** store configurations but do not control networking.
- **NodePort (D)** exposes services externally but is not required for internal pod communication.

---

7. Kubernetes RBAC Roles

Question: Which **RBAC resource** is used to restrict access to **only specific resources** in a namespace?

A. **ClusterRole**
B. **Role**
C. **RoleBinding**
D. **PodSecurityPolicy**

Answer: B

Explanation:

- **Roles (B)** apply permissions within a specific namespace.
- **ClusterRoles (A)** grant permissions cluster-wide.
- **RoleBindings (C)** assign roles to users but do not define permissions.
- **PodSecurityPolicy (D)** is used for security settings, not access control.

---

8. Kubernetes Horizontal Pod Autoscaler (HPA) Metrics

Question: What metric does the **Horizontal Pod Autoscaler (HPA)** use by default to scale pods?

A. **Memory utilization**
B. **CPU utilization**
C. **Disk I/O**
D. **Pod restarts**

Answer: B

Explanation:

- **HPA scales based on CPU utilization (B) by default.**
- **Memory utilization (A)** requires custom metrics.
- **Disk I/O (C) and pod restarts (D)** are not used for autoscaling.

---

9. Kubernetes Cluster Autoscaler

Question: You need to automatically **add and remove nodes** from a Kubernetes cluster based on pod scheduling needs. Which feature should you use?

A. **Cluster Autoscaler**
B. **Horizontal Pod Autoscaler**
C. **Vertical Pod Autoscaler**
D. **NodeSelector**

Answer: A

Explanation:

- **Cluster Autoscaler (A)** adds/removes worker nodes based on demand.
- **HPA (B)** scales pods, not nodes.
- **VPA (C)** adjusts pod resource requests.
- **NodeSelector (D)** assigns pods to specific nodes but does not scale them.

---

10. Kubernetes Ingress Controllers

Question: You want to route incoming HTTP requests based on path rules (e.g., `/api` to one service, `/web` to another). Which resource should you use?

A. **Service of type LoadBalancer**
B. **Ingress**
C. **NetworkPolicy**
D. **ConfigMap**

Answer: B

Explanation:

- **Ingress (B)** handles HTTP routing based on host/path rules.
- **LoadBalancer (A)** exposes a single backend service.
- **NetworkPolicies (C)** control internal networking but do not handle HTTP routing.
- **ConfigMaps (D)** store configurations but do not handle traffic routing.

SET C

1. Kubernetes Pod-to-Pod Communication

Question: By default, how do pods communicate with each other in Kubernetes?

A. **Pods in the same namespace can communicate directly, but pods in different namespaces require NetworkPolicies.**
B. **All pods can communicate with each other across nodes and namespaces without any restrictions.**
C. **Pods communicate using NAT (Network Address Translation).**
D. **Pods must explicitly expose a service to communicate with each other.**

Answer: B

Explanation:

- Kubernetes **uses a flat network model** where **all pods can communicate without NAT** (B).
- **NetworkPolicies (A)** must be explicitly defined to restrict traffic.
- **Pods communicate without NAT (C)** because they are assigned unique IPs.
- **Services (D)** provide stable access but are not required for pod-to-pod communication.

---

2. Kubernetes Admission Controllers

Question: Which **admission controller** can prevent pods from being created if they do not have specific labels?

A. **ValidatingWebhook**
B. **MutatingWebhook**
C. **ResourceQuota**
D. **PodSecurityAdmission**

Answer: A

Explanation:

- **ValidatingWebhook (A)** can block resource creation based on policies (e.g., missing labels).
- **MutatingWebhook (B)** modifies requests but does not block them.
- **ResourceQuota (C)** enforces limits on resources but does not validate labels.
- **PodSecurityAdmission (D)** enforces pod security standards but does not check labels.

---

3. Kubernetes ServiceAccount Tokens

Question: What is the primary purpose of a **ServiceAccount** in Kubernetes?

A. **To provide network identity to a pod for inter-pod communication.**
B. **To grant API access to pods running inside the cluster.**
C. **To create persistent storage volumes for stateful workloads.**
D. **To assign a unique hostname to each pod.**

Answer: B

Explanation:

- **ServiceAccounts (B)** allow pods to authenticate to the Kubernetes API.
- **Pods communicate via networking, not identity (A).**
- **Storage (C)** is managed via PersistentVolumes.
- **Hostnames (D)** are assigned based on pod DNS configurations.

---

4. Kubernetes Persistent Volumes

Question: Which **StorageClass** reclaim policy allows Kubernetes to automatically delete the PersistentVolume when its corresponding PersistentVolumeClaim is deleted?

A. **Retain**
B. **Recycle**
C. **Delete**
D. **Recreate**

Answer: C

Explanation:

- **Delete (C)** removes the PersistentVolume when its PersistentVolumeClaim is deleted.
- **Retain (A)** keeps the volume even after the claim is deleted.
- **Recycle (B)** is deprecated in favor of external storage solutions.
- **Recreate (D)** is not a valid reclaim policy.

---

5. Kubernetes etcd Backup and Restore

Question: What is the recommended method to back up an **etcd** database in a Kubernetes cluster?

A. **Use `kubectl get all -o yaml` and save the output.**
B. **Take a snapshot using `etcdctl snapshot save`.**
C. **Export the Kubernetes ConfigMap data.**
D. **Run `kubectl backup cluster` (non-existent command).**

Answer: B

Explanation:

- **`etcdctl snapshot save` (B)** creates a point-in-time backup of the etcd database.
- **Kubectl YAML exports (A)** do not include all cluster states.
- **ConfigMaps (C)** do not store full cluster state.
- **Option D is an invalid command.**

---

6. Kubernetes Pod Security Standards

Question: You need to **enforce least privilege security** by ensuring that pods do **not** run as root. How can you enforce this policy?

A. **Using NetworkPolicies**
B. **Setting a SecurityContext in the Pod spec**
C. **Using a Kubernetes ServiceAccount**
D. **By enabling Pod Priority Classes**

Answer: B

Explanation:

- **SecurityContext (B)** allows you to enforce pod security policies such as preventing root access.
- **NetworkPolicies (A)** control network access but not security privileges.
- **ServiceAccounts (C)** manage API authentication, not runtime security.
- **Pod Priority Classes (D)** control scheduling priority, not security.

---

7. Kubernetes Audit Logs

Question: Where does Kubernetes store **API audit logs** by default?

A. **etcd database**
B. **Systemd logs on worker nodes**
C. **/var/log/kube-apiserver-audit.log**
D. **/var/lib/kubernetes/logs**

Answer: C

Explanation:

- Kubernetes **audit logs are stored in `/var/log/kube-apiserver-audit.log` (C)** when enabled.
- **etcd (A)** stores cluster state, not logs.
- **Systemd logs (B)** only log system-related events.
- **Path D is incorrect and does not exist by default.**

---

8. Kubernetes Node Drain

Question: What happens when you run `kubectl drain <node>`?

A. **All running pods on the node are forcibly deleted immediately.**
B. **Pods with ReplicaSets are rescheduled on other nodes, while standalone pods are evicted.**
C. **All DaemonSet pods are evicted.**
D. **New pods can still be scheduled on the drained node.**

Answer: B

Explanation:

- **Drain (B)** evicts pods managed by ReplicaSets but cannot evict DaemonSet pods unless explicitly specified.
- **Pods are not deleted immediately (A), they are gracefully evicted.**
- **DaemonSet pods are not evicted by default (C).**
- **New pods are prevented from scheduling (D is incorrect).**

---

9. Kubernetes Pod Disruption Budget (PDB)

Question: You want to ensure that at least **one pod of a deployment** remains running during node maintenance. How can you enforce this?

A. **Set a minimum `replicas: 1` in the Deployment.**
B. **Create a PodDisruptionBudget (PDB).**
C. **Use a PriorityClass to protect the pod.**
D. **Use a Horizontal Pod Autoscaler.**

Answer: B

Explanation:

- **PodDisruptionBudget (B)** ensures a minimum number of available replicas during voluntary disruptions.
- **Replica count (A)** does not prevent disruptions.
- **PriorityClasses (C)** affect scheduling priority, not disruptions.
- **HPA (D)** scales based on metrics, not maintenance events.

---

10. Kubernetes Secrets Security

Question: How can you **secure Kubernetes Secrets** stored in a cluster?

A. **Store secrets as ConfigMaps.**
B. **Use `kubectl create secret` and enable encryption at rest.**
C. **Pass secrets as environment variables in pod specs.**
D. **Store secrets in plaintext inside manifests.**

Answer: B

Explanation:

- **Option B is the correct approach—Kubernetes Secrets should be encrypted at rest using KMS or custom encryption providers.**
- **ConfigMaps (A) are not designed for sensitive data.**
- **Environment variables (C) expose secrets in pod specifications.**
- **Plaintext in manifests (D) is insecure and violates security best practices.**

SET D

1. Kubernetes Role-Based Access Control (RBAC)

Question: You need to allow a developer to **view and list pods** in a namespace but prevent them from deleting pods. Which RBAC resource should you create?

A. **RoleBinding with a ClusterRole of `admin`**
B. **ClusterRoleBinding with the `view` role**
C. **Role with `get`, `list`, and `watch` permissions on pods**
D. **RoleBinding that grants `delete` permissions only for the developer's pod**

Answer: C

Explanation:

- **A Role (C)** scoped to a namespace with `get`, `list`, and `watch` permissions is correct.
- **A ClusterRoleBinding (B)** applies to the whole cluster, which is not needed here.
- **Option D grants delete permissions, which is what we want to avoid.**

---

2. Kubernetes Static Pods

Question: How do **static pods** differ from regular Kubernetes pods?

A. **Static pods are created by the kube-apiserver, while regular pods are created by kubelet.**
B. **Static pods are managed directly by the kubelet and do not have an associated Deployment or ReplicaSet.**
C. **Static pods can be rescheduled to another node if the node fails.**
D. **Static pods do not support logs or monitoring via `kubectl logs`.**

Answer: B

Explanation:

- **Static pods are managed by the kubelet (B) and do not go through the API server.**
- **They are not rescheduled (C) because there is no controller managing them.**
- **Logs can still be accessed using `kubectl logs` (D is incorrect).**

---

3. Kubernetes Ingress TLS Configuration

Question: You need to secure an **Ingress resource** with TLS encryption. What steps should you take?

A. **Enable HTTPS in the Ingress controller and provide a Secret containing the TLS certificate.**
B. **Modify the Service definition to use TLS instead of HTTP.**
C. **Store the TLS certificate in a ConfigMap and reference it in the Ingress spec.**
D. **Configure NetworkPolicy to enforce TLS between pods.**

Answer: A

Explanation:

- **Option A is correct—you must enable HTTPS and provide a Kubernetes Secret containing the TLS certificate.**
- **Ingress handles TLS, not the Service (B is incorrect).**
- **ConfigMaps (C) are not used for TLS certificates.**
- **NetworkPolicies (D) control traffic but do not enforce TLS at the ingress level.**

---

4. Kubernetes Node Selectors vs. Affinity

Question: What is the key difference between **nodeSelector** and **nodeAffinity**?

A. **nodeSelector supports expressions, while nodeAffinity only supports exact matches.**
B. **nodeAffinity is more flexible and supports `preferred` and `required` scheduling rules, whereas nodeSelector only supports exact label matches.**
C. **nodeSelector requires multiple labels, whereas nodeAffinity requires only one label.**
D. **nodeAffinity does not allow node selection based on labels.**

Answer: B

Explanation:

- **nodeSelector (B) only supports exact label matches.**
- **nodeAffinity (B) supports expressions and `preferred` vs. `required` rules.**
- **nodeSelector does not require multiple labels (C is incorrect).**

---

5. Kubernetes Troubleshooting - CrashLoopBackOff

Question: A pod is stuck in `CrashLoopBackOff`. What is the best command to investigate the issue?

A. `kubectl describe pod <pod-name>`
B. `kubectl get events --all-namespaces`
C. `kubectl logs <pod-name>`
D. **All of the above**

Answer: D

Explanation:

- `kubectl describe pod` **(A) provides detailed pod information and reasons for restarts.**
- `kubectl get events` **(B) helps check for related issues in the namespace.**
- `kubectl logs` **(C) retrieves logs for debugging the application itself.**

---

6. Kubernetes Network Plugins (CNI)

Question: What is the role of a **Container Network Interface (CNI)** in Kubernetes?

A. **To provide pod-to-pod networking across nodes**
B. **To expose Kubernetes services externally**
C. **To encrypt all network traffic between pods**
D. **To configure TLS certificates for communication between nodes**

Answer: A

Explanation:

- **CNI plugins (A) handle pod networking within and across nodes.**
- **Services (B) use different mechanisms, not CNI.**
- **Encryption (C) can be handled by additional tools like Istio or service meshes.**
- **TLS certificates (D) are not managed by CNI.**

---

7. Kubernetes Pod Eviction

Question: What happens when a node reaches its memory limit?

A. **The node shuts down automatically.**
B. **Kubernetes evicts low-priority pods to free up memory.**
C. **All pods on the node are restarted.**
D. **Pods are automatically moved to another node.**

Answer: B

Explanation:

- **Kubernetes evicts pods (B) based on priority and QoS classes.**
- **Nodes do not shut down automatically (A).**
- **Pods are not restarted unless a controller handles them (C).**
- **Pods are not "moved" but rather restarted elsewhere if scheduling allows (D).**

---

8. Kubernetes etcd Key-Value Store

Question: What type of data is stored in **etcd**?

A. **Pod logs**
B. **Cluster configuration, API objects, and state data**
C. **Docker images used by the nodes**
D. **PersistentVolumeClaim data**

Answer: B

Explanation:

- **etcd stores the entire Kubernetes cluster state and configuration (B).**
- **Pod logs (A) are stored separately in logging systems.**
- **Docker images (C) are stored in container registries, not etcd.**
- **PersistentVolumeClaims (D) reference storage, but the data itself is elsewhere.**

---

9. Kubernetes Helm Package Manager

Question: What is the purpose of **Helm** in Kubernetes?

A. **To manage application deployments using pre-defined templates**
B. **To monitor Kubernetes logs and metrics**
C. **To provide a network overlay for pod communication**
D. **To perform automatic scaling of workloads**

Answer: A

Explanation:

- **Helm (A) helps package, install, and manage Kubernetes applications.**
- **Logging (B) is handled by tools like Fluentd, Loki, and ELK.**
- **Networking (C) is handled by CNI plugins.**
- **Scaling (D) is handled by the HPA or Cluster Autoscaler.**

---

10. Kubernetes API Server Authentication

Question: Which method is NOT a valid authentication mechanism for the **Kubernetes API Server**?

A. **X.509 Client Certificates**
B. **ServiceAccount Tokens**
C. **NetworkPolicies**
D. **OIDC (OpenID Connect) Tokens**

Answer: C

Explanation:

- **NetworkPolicies (C) control networking but do not handle authentication.**
- **X.509 certificates (A), ServiceAccount tokens (B), and OIDC tokens (D) are all valid authentication methods.**

SET E

1. Kubernetes Role-Based Access Control (RBAC)

Question: You need to grant a user permission to create, modify, and delete deployments in a specific namespace. Which resource should you use?

A. **ClusterRoleBinding**
B. **RoleBinding with a Role that allows access to deployments**
C. **ServiceAccount with elevated privileges**
D. **NetworkPolicy that allows API access from the user's workstation**

Answer: B

Explanation:

- **RoleBinding (B) grants permissions within a specific namespace.**
- **ClusterRoleBinding (A) applies cluster-wide, which is unnecessary here.**
- **ServiceAccounts (C) are for workloads, not user access.**
- **NetworkPolicies (D) do not control API access.**

---

2. Kubernetes Control Plane Components

Question: Which Kubernetes component is responsible for **scheduling pods onto nodes**?

A. **kube-apiserver**
B. **kubelet**
C. **kube-controller-manager**
D. **kube-scheduler**

Answer: D

Explanation:

- **kube-scheduler (D) assigns pods to nodes based on resource availability and constraints.**
- **kube-apiserver (A) is the front-end of the control plane.**
- **kubelet (B) runs on each node and manages pods.**
- **kube-controller-manager (C) manages controllers like ReplicaSet and Node Controller.**

---

3. Kubernetes Network Policies

Question: You need to block **all ingress traffic** to a set of pods except from a specific namespace. What should you use?

A. **Ingress Controller**
B. **NetworkPolicy with `Ingress` rules**
C. **NodeSelector to restrict pod placement**
D. **ServiceAccount-based authentication**

Answer: B

Explanation:

- **NetworkPolicy (B) controls pod-to-pod traffic based on namespace and label selectors.**
- **Ingress Controllers (A) handle external traffic.**
- **NodeSelector (C) controls pod scheduling but does not block traffic.**
- **ServiceAccounts (D) manage API access, not network traffic.**

---

4. Kubernetes High Availability

Question: How do you make an **etcd cluster highly available** in a Kubernetes control plane?

A. **Run a single etcd instance on the master node.**
B. **Deploy etcd in a multi-node configuration with an odd number of instances.**
C. **Use a PersistentVolume to store etcd data on a single node.**
D. **Configure a network policy that ensures only kube-apiserver can communicate with etcd.**

Answer: B

Explanation:

- **etcd should be deployed with an odd number of nodes (B) to maintain quorum.**
- **Single-node etcd (A) is a single point of failure.**
- **PersistentVolumes (C) do not provide HA.**
- **NetworkPolicies (D) enhance security but do not affect HA.**

---

5. Kubernetes Readiness Probe

Question: What happens when a **readiness probe** fails for a running pod?

A. **The pod is restarted immediately.**
B. **The pod is removed from the Service's endpoints but continues running.**
C. **The pod is evicted from the node.**
D. **All containers in the pod are automatically stopped.**

Answer: B

Explanation:

- **A failing readiness probe (B) removes the pod from service load balancers but does not restart it.**
- **Liveness probes (A) trigger restarts, not readiness probes.**

- **Pods are not evicted (C) when readiness fails.**
- **Containers remain running (D).**

---

6. Kubernetes Persistent Volume Claims

Question: What happens when a **PersistentVolumeClaim (PVC)** is deleted but the underlying PersistentVolume (PV) uses the **Retain** reclaim policy?

A. **The PersistentVolume is deleted automatically.**
B. **The PersistentVolume is detached but remains available for manual recovery.**
C. **The PersistentVolume is recycled for reuse by other PVCs.**
D. **All data on the PersistentVolume is erased immediately.**

Answer: B

Explanation:

- **Retain (B) keeps the PersistentVolume after PVC deletion for manual reuse.**
- **Delete (A) is a different reclaim policy that removes the PV.**
- **Recycle (C) is deprecated.**
- **Data is not erased automatically (D).**

---

7. Kubernetes Pod Priority and Preemption

Question: A high-priority pod cannot start due to insufficient resources, but there are lower-priority pods running. What will Kubernetes do?

A. **Wait indefinitely until more nodes are added.**
B. **Evict lower-priority pods to free up resources.**
C. **Restart the lower-priority pods with reduced resource limits.**
D. **Ignore priority settings and schedule the pod when possible.**

Answer: B

Explanation:

- **Kubernetes preemptively evicts lower-priority pods (B) to make room.**
- **It does not wait indefinitely (A).**
- **Pods are not restarted with modified limits (C).**
- **Priority is respected (D is incorrect).**

---

8. Kubernetes API Aggregation Layer

Question: What is the purpose of the **API Aggregation Layer** in Kubernetes?

A. **To expose additional APIs without modifying the core Kubernetes API server.**
B. **To increase the performance of API requests by caching responses.**
C. **To enforce security policies for external API calls.**
D. **To provide an alternative to etcd for storing API objects.**

Answer: A

Explanation:

- **API Aggregation (A) allows new APIs to be added without modifying the API server.**
- **It does not cache responses (B).**
- **Security policies (C) are handled elsewhere.**
- **etcd remains the primary data store (D).**

---

9. Kubernetes CRI (Container Runtime Interface)

Question: Which container runtime is NOT compatible with Kubernetes' **Container Runtime Interface (CRI)**?

A. **containerd**
B. **Docker (without CRI plugin)**
C. **CRI-O**
D. **gVisor**

Answer: B

Explanation:

- **Docker (B) is not CRI-compatible by default and requires the `cri-dockerd` plugin.**
- **containerd (A) and CRI-O (C) are CRI-compliant.**
- **gVisor (D) is a sandbox but integrates with CRI runtimes.**

---

10. Kubernetes Troubleshooting - API Server

Question: The **kube-apiserver** is down. What command can you run on a control plane node to check its status?

A. `kubectl get pods -n kube-system`
B. `systemctl status kube-apiserver`
C. `kubectl logs kube-apiserver`
D. `etcdctl cluster-health`

Answer: B

Explanation:

- `systemctl status kube-apiserver` **(B) checks the API server service status.**
- `kubectl` **(A, C) depends on the API server, which is down.**
- `etcdctl` **(D) checks etcd, not the API server.**