DevOps Shack

200 Kubernetes Interview Questions and answers

1. What is Kubernetes, and why is it used?

Answer: Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It ensures that application workloads are highly available, resilient, and can handle dynamic scaling.

- Key Features:
  - Automated scheduling and resource optimization.
  - Self-healing capabilities (restarting failed containers).
  - Horizontal scaling of applications.
  - Service discovery and load balancing.
  - Rolling updates and rollbacks.

2. Explain the architecture of Kubernetes.

Answer: Kubernetes has a master-worker architecture:

1. Control Plane (Master Node):
   - API Server: Exposes Kubernetes APIs for communication.
   - Scheduler: Assigns workloads to worker nodes based on resources and policies.
   - Controller Manager: Manages controllers like replication, deployment, etc.
   - etcd: Stores cluster configuration data (key-value store).
2. Worker Nodes:

- Kubelet: Ensures that containers are running as expected.
- Kube-proxy: Handles network rules for communication.
- Container Runtime: Executes containers (e.g., Docker, containerd).

3. What is a Pod in Kubernetes?

Answer: A Pod is the smallest deployable unit in Kubernetes. It can host one or more tightly coupled containers that share:

- Storage: Shared volumes.
- Network: A single IP address.
- Lifecycle: Containers in a pod start and stop together.

Example Use Case: A web server and a logging agent running together in a pod.

4. How does Kubernetes handle networking for Pods?

Answer: Kubernetes provides a flat networking model:

1. Every Pod gets a unique IP address.
2. Pods can communicate with each other directly without NAT.
3. Kubernetes uses CNI plugins (e.g., Calico, Flannel) to manage networking.

Key Components:

- Service: Exposes a stable IP and DNS for pods.
- Ingress: Manages external HTTP/HTTPS traffic.
- NetworkPolicy: Enforces traffic restrictions between pods.

5. What is a ReplicaSet, and how is it different from a Deployment?

Answer:

- ReplicaSet ensures a specified number of pod replicas are running at all times.
- Deployment is a higher-level abstraction that manages ReplicaSets and supports rolling updates and rollbacks.

Example: Use a Deployment for versioned app releases.

6. What are ConfigMaps and Secrets? How are they different?

Answer:

- ConfigMaps: Store non-confidential configuration data (e.g., config files, environment variables).
- Secrets: Store sensitive data (e.g., passwords, tokens) in a base64-encoded format.

Difference: Secrets are encrypted at rest, while ConfigMaps are not.

7. What is the purpose of a Kubernetes Service?

Answer: A Service provides a stable network endpoint to access a set of Pods.

- Types:
    - ClusterIP: Internal communication.
    - NodePort: Exposes on a static port of the worker nodes.
    - LoadBalancer: Integrates with cloud providers' load balancers.
    - ExternalName: Maps to an external DNS name.

8. What is the role of a PersistentVolume (PV) and PersistentVolumeClaim (PVC)?

Answer:

- PersistentVolume (PV): Represents a storage resource provisioned in the cluster.
- PersistentVolumeClaim (PVC): Requests specific storage characteristics.

Workflow:

1. Admin creates a PV.
2. User creates a PVC.
3. Kubernetes binds the PVC to a matching PV.

9. How does Kubernetes perform Rolling Updates?

Answer: Kubernetes uses Deployments to perform rolling updates, gradually replacing old pod versions with new ones to minimize downtime.

Key Steps:

1. `kubectl apply` the new Deployment.
2. Kubernetes spins up new pods while terminating the old ones.
3. Users experience no downtime if properly configured.

10. What is a Kubernetes Namespace?

Answer: Namespaces allow you to logically divide a Kubernetes cluster into virtual clusters for:

- Resource isolation.
- Separate environments (e.g., dev, staging, production).

Example: `kubectl get pods --namespace=<namespace-name>`

11. What are Kubernetes Labels and Selectors?

Answer:

- Labels: Key-value pairs attached to Kubernetes objects (e.g., Pods, Nodes) for identification and grouping.
- Selectors: Used to query objects based on labels.

Example:

```
metadata:

  Labels:
    app: frontend
```

Selectors:

```
kubectl get pods -l app=frontend
```

12. What is the difference between a DaemonSet and a Deployment?

Answer:

- DaemonSet ensures that a copy of a pod runs on all (or specific) nodes.
    - Used for system-level services like logging and monitoring.
- Deployment ensures a specified number of replicas of an application pod.

13. What are the different types of Kubernetes volumes?

Answer: Kubernetes supports many volume types:

1. EmptyDir: Temporary storage, deleted when a pod is deleted.
2. HostPath: Maps a host machine directory.
3. PersistentVolume: Long-term storage.
4. ConfigMap/Secret: Inject configuration or secrets.
5. NFS, AWS EBS, Azure Disk, GCE Persistent Disk: Cloud-specific volumes.

14. What is a Kubernetes Ingress?

Answer: Ingress is an API object that manages external HTTP/S access to services.

Features:

- URL-based routing.
- SSL/TLS termination.
- Load balancing.

Example Ingress YAML:

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

15. What are Resource Quotas in Kubernetes?

Answer: Resource quotas limit the resource usage in a namespace, controlling CPU, memory, and object counts.

Example YAML:

```
apiVersion: v1

kind: ResourceQuota

metadata:

  name: cpu-mem-quota

spec:

  hard:

    requests.cpu: "4"

    requests.memory: "16Gi"

    limits.cpu: "8"

    limits.memory: "32Gi"
```

16. Explain Kubernetes Horizontal Pod Autoscaler (HPA).

Answer: HPA scales the number of pods in a Deployment/ReplicaSet based on metrics like CPU, memory, or custom metrics.

Steps:

1. Enable the Metrics Server.
2. Apply an HPA object.

Example:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
Metadata:
  name: example-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

17. What is a Custom Resource Definition (CRD)?

Answer: CRDs extend Kubernetes by defining custom resources that behave like built-in resources.

Use Case: Create objects like `MySQLCluster` or `KafkaTopic`.

Example CRD YAML:

```
apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

metadata:

  name: widgets.example.com

spec:

  group: example.com

  names:

    kind: Widget

    listKind: WidgetList

    plural: widgets

    singular: widget

  scope: Namespaced

  versions:

  - name: v1
```

```
    served: true

    storage: true
```

18. What are Kubernetes Admission Controllers?

Answer: Admission Controllers are plugins that intercept API requests to validate or modify them.

Types:

- Validating Admission Controllers: Validate requests (e.g., deny pods without specific labels).
- Mutating Admission Controllers: Modify requests (e.g., inject sidecars).

19. How does Kubernetes implement High Availability (HA)?

Answer:

1. Control Plane HA:
   - Use multiple master nodes.
   - Use etcd clustering for redundancy.
   - Load balancer for API server endpoints.
2. Worker Node HA:
   - Schedule multiple replicas of pods.
   - Use taints, tolerations, and node affinity.

20. How do you debug a Kubernetes Pod?

Answer:

Check pod status:

```
kubectl get pods
```

Describe the pod:

```
kubectl describe pod <pod-name>
```

View logs:

```
kubectl logs <pod-name>
```

21. How do you monitor a Kubernetes cluster?

Answer: Monitoring a Kubernetes cluster involves tracking system health, performance, and logs.

Key Tools:

- Prometheus: For metrics collection.
- Grafana: For visualizing metrics.
- ELK Stack (Elasticsearch, Logstash, Kibana): For log aggregation and analysis.
- Kubernetes Dashboard: Native UI for basic monitoring.
- K9s: Terminal-based cluster monitoring tool.

22. What are Taints and Tolerations in Kubernetes?

Answer:

- Taints: Prevent pods from being scheduled on a node unless the pod has a matching toleration.
- Tolerations: Allow pods to tolerate a node's taints.

Example: Taint a node:

```
kubectl taint nodes node1 key=value:NoSchedule
```

Add a toleration in a pod spec:

```
Tolerations:
```

```
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

23. What is a StatefulSet, and when is it used?

Answer: A StatefulSet manages stateful applications, ensuring each pod has:

- A unique and stable identity.
- Persistent storage tied to the pod's lifecycle.

Use Case: Databases, Kafka, Elasticsearch.

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  serviceName: "mysql-service"
  replicas: 3
  selector:
    matchLabels:
      app: mysql
  Template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        ports:
        - containerPort: 3306
```

24. How do you implement Kubernetes Secrets securely?

Answer:

Use `kubectl create secret` to generate secrets:

```
kubectl create secret generic my-secret
--from-literal=username=admin --from-literal=password=secret
```

Reference it in a Pod spec:

```
env:
- name: DB_USER
```

```
  valueFrom:
    secretKeyRef:
      name: my-secret
      key: username
```

1. Use encryption for secrets at rest with EncryptionConfig.
2. Restrict access using RBAC.

25. What is Kubernetes RBAC?

Answer: Role-Based Access Control (RBAC) restricts access to resources in Kubernetes based on roles and bindings.

Components:

- Role/ClusterRole: Defines permissions.
- RoleBinding/ClusterRoleBinding: Assigns roles to users or groups.

Example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
```

```
  resources: ["pods"]

  verbs: ["get", "list", "watch"]
```

26. What is the difference between NodePort and LoadBalancer services?

Answer:

- **NodePort: Exposes the service on a static port across all nodes.**
  - **Accessible via `<NodeIP>:<NodePort>`.**
- **LoadBalancer: Automatically provisions a cloud provider's external load balancer.**

27. How do you troubleshoot Kubernetes networking issues?

Answer:

Check Pod Networking:

```
kubectl exec -it <pod> -- curl <target-service>
```

Inspect Services:

```
kubectl get svc
```

1. **Verify Network Policies: Ensure correct ingress/egress rules.**

Test DNS Resolution:

```
kubectl exec -it <pod> -- nslookup <service-name>
```

28. What is a Kubernetes Job?

Answer: A Job is a controller that ensures a specified number of pods complete successfully.

Use Case: Batch processing tasks, data migration.

Example YAML:

```
apiVersion: batch/v1
kind: Job
Metadata:
  name: example-job
spec:
  template:
    spec:
      containers:
      - name: job-container
        image: busybox
        command: ["echo", "Hello, Kubernetes!"]
      restartPolicy: Never
  backoffLimit: 4
```

29. What are NetworkPolicies in Kubernetes?

Answer: NetworkPolicies define how pods communicate with each other and external services.

Example YAML:

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
metadata:
  name: allow-web
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: frontend
    ports:
    - protocol: TCP
```

```
      port: 80
```

30. How do you implement Helm in Kubernetes?

Answer: Helm is a package manager for Kubernetes, used to manage applications via Charts.

1. Install Helm CLI.

Add a chart repository:

```
helm repo add stable https://charts.helm.sh/stable
```

Install a chart:

```
helm install my-release stable/nginx
```

Upgrade a release:

```
helm upgrade my-release stable/nginx
```

31. What is the difference between a Deployment and a Job?

Answer:

- Deployment: Manages long-running applications and ensures pod availability.
- Job: Runs short-lived tasks until completion.

32. How do you handle Kubernetes logs?

Answer:

View logs for a pod:

```
kubectl logs <pod-name>
```

Stream logs:

```
kubectl logs -f <pod-name>
```

1. Centralized logging tools:

    - Fluentd: For log collection.
    - ELK Stack: For aggregation and search.

33. What is a Kubernetes Kubeconfig file?

Answer: The Kubeconfig file stores cluster access credentials.

Default Location:

```
~/.kube/config
```

Switch between contexts:

```
kubectl config use-context <context-name>
```

34. What is Kubernetes API Aggregation Layer?

Answer: It allows the addition of APIs to extend Kubernetes functionality without modifying the core.

Use Case: Add custom metrics for autoscaling.

35. How do you handle version upgrades in Kubernetes?

Answer:

1. Backup etcd.
2. Upgrade the control plane components (API server, scheduler, etc.).
3. Upgrade kubelets on worker nodes.
4. Test workloads for compatibility.

36. How do you scale applications in Kubernetes?

Answer:

Manual Scaling:

```
kubectl scale deployment <deployment-name>
--replicas=<desired-replicas>
```

Horizontal Pod Autoscaler (HPA): Automatically adjusts the number of pods based on CPU, memory, or custom metrics.

```
apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

  name: hpa-example

spec:

  scaleTargetRef:

    apiVersion: apps/v1

    kind: Deployment

    name: example-app

  minReplicas: 2

  maxReplicas: 10

  metrics:

  - type: Resource

    resource:

      name: cpu
      Target:

        type: Utilization

        averageUtilization: 50
```

1. Cluster Autoscaler: Automatically adjusts the size of the cluster by scaling nodes.

37. How do you implement Kubernetes Blue-Green Deployments?

Answer: Blue-Green Deployments involve running two environments (Blue = current version, Green = new version) and switching traffic once the Green version is verified.

Steps:

1. Deploy the new version (`Green`) alongside the old (`Blue`).
2. Route traffic to the `Green` version via service or ingress.
3. Delete the old version after testing.

Example: Two deployments for `Blue` and `Green`:

```
apiVersion: apps/v1

kind: Deployment

metadata:

  name: blue-deployment

spec:

  replicas: 2

  template:
    Metadata:

      Labels:

        app: my-app

        version: blue

apiVersion: apps/v1

kind: Deployment

metadata:

  name: green-deployment

spec:

  replicas: 2

  template:

    metadata:

      labels:

        app: my-app

        version: green
```

38. What are the best practices for managing secrets in Kubernetes?

Answer:

1. **Use Kubernetes Secrets with limited RBAC access.**
2. **Enable Encryption at Rest:**

- Update `kube-apiserver` with `--encryption-provider-config`.

3. Use external tools like HashiCorp Vault or AWS Secrets Manager.
4. Avoid hardcoding sensitive data in manifests.

39. What are Init Containers in Kubernetes?

Answer: Init containers run before the main application container in a pod. They are used for initialization tasks such as setting up configurations or waiting for a dependency.

Example YAML:

```
apiVersion: v1
kind: Pod
metadata:
  name: init-container-example
spec:
  initContainers:
  - name: init-myservice
    image: busybox
    command: ["sh", "-c", "echo Initializing... && sleep 5"]
  containers:
  - name: my-app
```

```
    image: nginx
```

40. What are Kubernetes Probes, and what are their types?

Answer: Probes are used to monitor the health of a container.

Types:

1. **Liveness Probe: Checks if the container is alive and should be restarted.**
2. **Readiness Probe: Checks if the container is ready to serve traffic.**
3. **Startup Probe: Ensures the container has started successfully.**

Example:

```
livenessProbe:

  httpGet:

    path: /healthz

    port: 8080

  initialDelaySeconds: 3

  periodSeconds: 5
```

41. How do you configure Kubernetes Storage Classes?

Answer: Storage Classes define the provisioner, parameters, and reclaim policy for persistent storage.

Example YAML:

```
apiVersion: storage.k8s.io/v1

kind: StorageClass
```

```
Metadata:

  name: fast

provisioner: kubernetes.io/aws-ebs

parameters:

  type: gp2

reclaimPolicy: Retain

volumeBindingMode: Immediate
```

42. What is the role of kube-proxy in Kubernetes?

Answer: kube-proxy is a network proxy that runs on each node in the cluster and:

- **Maintains network rules for Pods.**
- **Manages virtual IPs and load-balancing traffic to backend pods.**

43. What is Kubernetes Federation?

Answer: Federation allows the management of multiple Kubernetes clusters as a single entity.

Use Cases:

- **Disaster recovery.**
- **Multi-region deployments.**
- **Centralized management.**

44. What are Kubernetes DaemonSets, and when are they used?

Answer: A DaemonSet ensures that a copy of a pod runs on all or specific nodes.

Use Cases:

- Node monitoring (e.g., Prometheus Node Exporter).
- Log aggregation (e.g., Fluentd).

45. How do you debug a CrashLoopBackOff error?

Answer:

Describe the pod:

```
kubectl describe pod <pod-name>
```

Check pod logs:

```
kubectl logs <pod-name>
```

Start an interactive shell:

```
kubectl exec -it <pod-name> -- /bin/bash
```

1. Verify configuration (e.g., secrets, volumes).

46. What is Kubernetes API Server, and what role does it play?

Answer: The API server is the core component of the control plane, exposing Kubernetes APIs. It:

- Serves as the communication hub for cluster components.
- Authenticates and authorizes API requests.

47. What is Kubernetes etcd, and why is it important?

Answer: etcd is a distributed key-value store that:

- Stores all cluster configuration data.
- Provides high availability using Raft consensus.

Best Practices:

- Backup etcd regularly.

- Use encryption for data in etcd.

48. How do you implement Kubernetes Rolling Updates?

Answer: Rolling updates replace pods gradually without downtime.

Command:

```
kubectl set image deployment/<deployment-name>
<container-name>=<new-image>
```

Example YAML:

```
strategy:

  type: RollingUpdate

  rollingUpdate:

    maxSurge: 1

    maxUnavailable: 1
```

49. What is the difference between Deployment and ReplicaSet?

Answer:

- ReplicaSet: Ensures a specified number of pod replicas are running.
- Deployment: Manages ReplicaSets, supports rolling updates, and rollbacks.

50. How do you enforce pod security in Kubernetes?

Answer:

1. PodSecurityPolicy (Deprecated): Define security settings for pods.
2. Pod Security Admission:

- Set namespace labels (`restricted`, `baseline`, `privileged`).

3. Use tools like OPA Gatekeeper.

51. What is the Kubernetes Scheduler, and how does it work?

Answer: The Kubernetes Scheduler assigns pods to nodes based on available resources and scheduling policies.

Steps:

1. Evaluates pending pods.
2. Identifies eligible nodes based on constraints (e.g., resource requests, taints/tolerations).
3. Prioritizes nodes using weights (e.g., least loaded node).
4. Binds the pod to the chosen node.

52. What is a Kubernetes Service Mesh?

Answer: A Service Mesh manages service-to-service communication within a cluster, providing:

- Traffic control (e.g., canary deployments).
- Observability (e.g., tracing and metrics).
- Security (e.g., mutual TLS).

Popular Service Meshes:

- Istio
- Linkerd
- Consul

53. What is Kubernetes PersistentVolume Reclaim Policy?

Answer: The Reclaim Policy determines what happens to a PersistentVolume (PV) when its claim is deleted.

Types:

1. Retain: Retains the data for manual reuse.
2. Delete: Deletes the storage resource.
3. Recycle (deprecated): Clears the volume for reuse.

54. What are Kubernetes Node Affinity and Anti-Affinity?

Answer:

- Node Affinity: Ensures pods are scheduled on specific nodes.
- Node Anti-Affinity: Prevents pods from being scheduled on specific nodes.

Example YAML:

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: disktype
          operator: In
          values:
          - ssd
```

55. What are Kubernetes Horizontal Pod Autoscaler Metrics?

Answer: The Horizontal Pod Autoscaler (HPA) uses the following metrics:

1. Resource Metrics: CPU and memory usage.
2. Custom Metrics: Application-specific metrics via Prometheus Adapter.
3. External Metrics: Metrics from external systems (e.g., cloud services).

56. How does Kubernetes handle Stateful Applications?

Answer: Stateful applications require:

1. StatefulSets: Ensures unique identity and persistent storage.
2. PersistentVolumeClaims: For persistent data storage.
3. Headless Services: Provides stable DNS entries for pods.

57. What is Kubernetes Eviction, and why does it occur?

Answer: Eviction occurs when a node cannot meet resource demands.

Triggers:

1. Resource Pressure: CPU, memory, or disk usage exceeds thresholds.
2. Node Maintenance: Manual eviction for updates.
3. Pod Priority: Lower-priority pods are evicted first.

58. What is the difference between ConfigMap and Environment Variables?

Answer:

- ConfigMap: Stores non-sensitive configuration data.
- Environment Variables: Inject configuration directly into the container.

ConfigMap Example:

```
apiVersion: v1

kind: ConfigMap
```

```
Metadata:

  name: app-config
data:
  app-mode: production
```

Use in a Pod:

```
env:
- name: APP_MODE
  valueFrom:
    configMapKeyRef:
      name: app-config
      key: app-mode
```

59. What is Kubernetes Vertical Pod Autoscaler (VPA)?

Answer: VPA adjusts the resource requests and limits of running pods.

Use Case: Workloads with varying resource needs.

60. What are Kubernetes Admission Webhooks?

Answer: Admission webhooks modify or validate Kubernetes API requests.

Types:

1. Mutating Webhooks: Modify requests.
2. Validating Webhooks: Validate requests.

Example: Injecting a sidecar container using a mutating webhook.

61. What is the difference between a Service and an Ingress in Kubernetes?

Answer:

- Service: Exposes a set of pods internally or externally.
- Ingress: Manages HTTP/HTTPS routing to services, supports SSL termination.

62. How do you perform Canary Deployments in Kubernetes?

Answer:

1. Deploy a small percentage of pods with the new version.
2. Gradually increase the number of new pods while monitoring.

Example with Service:

- Split traffic using a load balancer or ingress.

63. What are Kubernetes RuntimeClasses?

Answer: RuntimeClasses allow Kubernetes to use different container runtimes (e.g., gVisor, Kata Containers) for enhanced security or performance.

64. What is Kubernetes API Priority and Fairness?

Answer: This feature ensures fair API request handling, prioritizing critical operations under high load.

65. What are Kubernetes Feature Gates?

Answer: Feature Gates enable or disable experimental Kubernetes features.

Enable Example:

```
kube-apiserver --feature-gates=IPv6DualStack=true
```

66. What is Kubernetes CRI (Container Runtime Interface)?

Answer: The CRI standardizes interactions between Kubernetes and container runtimes.

Supported Runtimes:

- **Docker (deprecated for Kubernetes).**
- **Containerd.**
- **CRI-O.**

67. How do you manage cluster upgrades in Kubernetes?

Answer:

1. **Upgrade the control plane components.**
2. **Upgrade worker nodes using kubeadm or automation tools like Kops.**
3. **Validate workloads after the upgrade.**

68. What is Kubernetes Pod Priority and Preemption?

Answer:

- **Priority: Determines the importance of a pod.**
- **Preemption: Higher-priority pods evict lower-priority pods under resource constraints.**

69. How does Kubernetes manage DNS for Pods?

Answer: Kubernetes integrates CoreDNS to provide DNS resolution for services and pods.

Pod DNS Example:

```
<service-name>.<namespace>.svc.cluster.local
```

70. What is Kubernetes Dual-Stack Networking?

Answer: Dual-Stack Networking allows pods and services to use both IPv4 and IPv6 addresses.

Enable Dual-Stack:

```
apiServer:

  extraArgs:

    feature-gates: "IPv6DualStack=true"
```

71. How do you secure the Kubernetes API Server?

Answer:

1. **Use RBAC to control API access.**
2. **Enable audit logging.**
3. **Use TLS for secure communication.**
4. **Restrict API server access to trusted networks.**

72. What are Kubernetes Sidecars?

Answer: Sidecars are helper containers that run alongside the main application container in the same pod.

Use Cases:

- **Logging agents.**
- **Proxies (e.g., Envoy for service mesh).**
- **Configuration updaters.**

73. What are Kubernetes Resource Requests and Limits?

Answer:

- Requests: Minimum guaranteed resources.

- Limits: Maximum resources a pod can use.

Example YAML:

```
resources:

  requests:

    memory: "64Mi"

    cpu: "250m"

  limits:

    memory: "128Mi"

    cpu: "500m"
```

74. What are Kubernetes Pod Disruption Budgets (PDB)?

Answer: A Pod Disruption Budget ensures a minimum number or percentage of pods remain available during voluntary disruptions (e.g., node updates).

Example YAML:

```
apiVersion: policy/v1

kind: PodDisruptionBudget

metadata:

  name: my-pdb

spec:
  minAvailable: 2
  selector:
    matchLabels:
      app: my-app
```

75. What is the Kubernetes ClusterIP Service?

Answer: ClusterIP is the default Kubernetes Service type, exposing the service only within the cluster.

Key Features:

- **Provides an internal virtual IP.**
- **Used for internal communication between pods.**

Example YAML:

```
apiVersion: v1
kind: Service
metadata:
  name: my-clusterip-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP



      port: 80

      targetPort: 8080
```

76. What is Kubernetes Node Maintenance Mode?

Answer: Node maintenance mode allows administrators to safely drain workloads from a node for updates or troubleshooting.

Steps:

Cordon the node (prevents new pods from being scheduled):

```
kubectl cordon <node-name>
```

Drain the node (safely evict pods):

```
kubectl drain <node-name> --ignore-daemonsets
--delete-emptydir-data
```

Perform maintenance, then uncordon:

```
kubectl uncordon <node-name>
```

77. What is Kubernetes Metrics Server?

Answer: The Metrics Server provides resource usage metrics for Kubernetes, enabling features like HPA.

Installation:

```
kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/l
atest/download/components.yaml
```

78. How do you handle Failed Pods in Kubernetes?

Answer:

Check pod events:

```
kubectl describe pod <pod-name>
```

View logs:

```
kubectl logs <pod-name>
```

Debug using ephemeral containers:

```
kubectl debug <pod-name> --image=busybox
```

Inspect Node Status:

```
kubectl get nodes

kubectl describe node <node-name>
```

79. What is Kubernetes HPA vs VPA?

Answer:

- Horizontal Pod Autoscaler (HPA): Scales the number of pods based on metrics (e.g., CPU).
- Vertical Pod Autoscaler (VPA): Adjusts pod resource requests and limits dynamically.

80. How do you manage Kubernetes Config Drift?

Answer:

- Use GitOps tools like ArgoCD or FluxCD to manage configurations.
- Regularly audit resources using kubectl diff or CI/CD pipelines.

81. What are Kubernetes Finalizers?

Answer: Finalizers are used to ensure cleanup tasks are performed before deleting a resource.

Example Use Case:

- Clean up cloud resources (e.g., AWS S3 buckets).

82. How do you implement Kubernetes Multi-Cluster Management?

Answer:

1. Use tools like Rancher, Kubefed, or Anthos.
2. Federate resources using Kubernetes Federation.
3. Centralize observability and monitoring.

83. What is Kubernetes Pod Overhead?

Answer: Pod overhead accounts for additional resources consumed by the pod infrastructure, ensuring accurate scheduling.

Enable Overhead in RuntimeClass:

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: my-runtime
overhead:
  podFixed:
    cpu: "100m"
```

```
    memory: "128Mi"
```

84. What is Kubernetes Kubelet, and what are its responsibilities?

Answer: The Kubelet is an agent running on each node, responsible for:

- Ensuring container health.
- Managing pod lifecycle.
- Reporting node status to the control plane.

85. What is the difference between Kubernetes PersistentVolume and PersistentVolumeClaim?

Answer:

- PersistentVolume (PV): Represents physical storage.
- PersistentVolumeClaim (PVC): Request for storage by a pod.

86. What are Kubernetes CSI Drivers?

Answer: The Container Storage Interface (CSI) standardizes storage plugins for Kubernetes.

Examples:

- AWS EBS CSI Driver.
- Azure Disk CSI Driver.

87. How do you monitor Kubernetes Logs?

Answer:

View logs for a specific pod:

```
kubectl logs <pod-name>
```

1. Aggregate logs using:
   - Fluentd
   - ELK Stack
   - Loki + Grafana

88. What are Kubernetes Token Review APIs?

Answer: Token Review APIs validate external authentication tokens (e.g., OIDC, service accounts).

Example: Authenticate using OIDC tokens.

89. What is the Kubernetes CronJob?

Answer: A CronJob schedules periodic tasks based on cron syntax.

Example YAML:

```
apiVersion: batch/v1

kind: CronJob

metadata:

  name: example-cronjob

spec:

  schedule: "*/5 * * * *"

  jobTemplate:

    spec:
```

```
Template:



  spec:
    containers:
    - name: hello
      image: busybox
      args:
      - /bin/sh
      - -c
      - "echo Hello, Kubernetes!"
    restartPolicy: OnFailure
```

90. What is the Kubernetes Default Service Account?

Answer: Each namespace has a default service account automatically attached to pods. It can be used to access the Kubernetes API.

91. How do you troubleshoot a Kubernetes Node?

Answer:

Check node status:

```
kubectl get nodes
```

Describe node:

```
kubectl describe node <node-name>
```

Inspect kubelet logs:

```
journalctl -u kubelet
```

92. What is Kubernetes Kustomize?

Answer: Kustomize allows customization of Kubernetes YAML files without modifying the originals.

Example:

```
kubectl kustomize <path-to-folder>
```

93. What is Kubernetes Operator Pattern?

Answer: Operators extend Kubernetes to manage custom resources and their lifecycles programmatically.

Examples:

- Prometheus Operator.
- Kafka Operator.

94. What is Kubernetes Ephemeral Containers?

Answer: Ephemeral containers are temporary containers used for debugging.

Example:

```
kubectl debug <pod-name> --image=busybox
```

95. How do you secure Kubernetes Clusters?

Answer:

1. Enable RBAC.
2. Use NetworkPolicies.
3. Encrypt secrets.
4. Restrict API access using firewalls.

5. Audit logs and monitor anomalies.

96. How do you handle a Kubernetes CrashLoopBackOff error?

Answer:

Check pod logs:

```
kubectl logs <pod-name>
```

1. Debug configuration (e.g., environment variables, secrets).
2. Test connectivity to dependencies.

97. What is Kubernetes Mutating Admission Webhook?

Answer: A Mutating Admission Webhook modifies Kubernetes API requests before they are persisted. It is useful for injecting default configurations or sidecar containers.

Example Use Case:

- Injecting logging or monitoring sidecars into pods.

98. What are Kubernetes EndpointSlices?

Answer: EndpointSlices are a scalable and efficient way to track network endpoints in Kubernetes. They replace the older Endpoints resource.

Key Benefits:

- Improved scalability.
- Reduced network traffic for large services.

99. What is Kubernetes Resource Quota?

Answer: Resource Quotas manage resource allocation in a namespace, limiting CPU, memory, and object counts.

Example YAML:

```
apiVersion: v1

kind: ResourceQuota

metadata:

  name: resource-quota-example

  namespace: dev

Spec:

  hard:

    pods: "10"

    requests.cpu: "4"

    requests.memory: "16Gi"

    limits.cpu: "8"

    limits.memory: "32Gi"
```

100. What is Kubernetes Cluster Autoscaler?

Answer: Cluster Autoscaler dynamically adjusts the number of cluster nodes based on pod requirements.

Key Features:

- **Scales up if pods cannot be scheduled due to insufficient resources.**
- **Scales down nodes if they are underutilized.**

Usage with cloud providers:

- AWS Auto Scaling Groups.
- Google GKE Node Pools.
- Azure VM Scale Sets.

101. What are Kubernetes Aggregated APIs?

Answer: Aggregated APIs extend Kubernetes functionality by adding custom APIs using the API server proxy.

Example Use Case:

- Add APIs for custom resources like monitoring tools or databases.

102. How does Kubernetes handle container runtime deprecation for Docker?

Answer: Starting with Kubernetes 1.20, Dockershim was deprecated in favor of CRI-compliant runtimes like containerd and CRI-O.

Migration Steps:

1. Install a CRI-compliant runtime (e.g., containerd).
2. Update kubelet to use the new runtime.
3. Test workloads for compatibility.

103. What is Kubernetes API Server Audit Logging?

Answer: Audit logging records all API requests to the Kubernetes API server. It is used for security and troubleshooting.

Enable Audit Logs:

Update API server flags:

```
--audit-log-path=/var/log/kubernetes/audit.log
```

```
--audit-policy-file=/etc/kubernetes/audit-policy.yaml
```

Define an audit policy:

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: RequestResponse
  resources:
  - group: ""
    resources: ["pods"]
```

104. What is the difference between Helm and Kustomize?

Answer:

- Helm: A package manager for deploying Kubernetes applications using charts.
- Kustomize: A tool to customize Kubernetes YAML files without templating.

When to Use:

- Use Helm for reusable application templates.
- Use Kustomize for environment-specific configuration.

105. What are Kubernetes PriorityClasses?

Answer: PriorityClasses define the importance of pods during scheduling and preemption.

Example YAML:

```
apiVersion: scheduling.k8s.io/v1

kind: PriorityClass

metadata:

  name: high-priority

value: 1000

globalDefault: false

description: "High-priority class for critical workloads."
```

106. How do you implement Pod Anti-Affinity in Kubernetes?

Answer: Pod Anti-Affinity ensures that certain pods are not scheduled on the same node.

Example YAML:

```
affinity:

  podAntiAffinity:

    requiredDuringSchedulingIgnoredDuringExecution:

    - labelSelector:

        matchExpressions:

        - key: app

          operator: In

          values:
              - my-app
```

```
        topologyKey: "kubernetes.io/hostname"
```

107. What is Kubernetes Service Topology?

Answer: Service Topology routes traffic to endpoints based on node topology (e.g., region, zone, hostname).

Enable Service Topology: Add this to the API server:

```
--feature-gates=ServiceTopology=true
```

108. How do you implement Rolling Back Kubernetes Deployments?

Answer: Rollback to the previous version of a deployment:

```
kubectl rollout undo deployment/<deployment-name>
```

Check deployment revision history:

```
kubectl rollout history deployment/<deployment-name>
```

109. What are Kubernetes StatefulSet Headless Services?

Answer: Headless Services allow direct pod-to-pod communication by disabling cluster IP allocation.

Example YAML:

```
apiVersion: v1

kind: Service

metadata:
  name: headless-service



Spec:

  clusterIP: None

  selector:

    app: my-app

  ports:

  - port: 80

    targetPort: 8080
```

110. How do you debug Kubernetes Networking Issues?

Answer:

Check Pod Network Connectivity:

```
kubectl exec -it <pod-name> -- ping <target-pod-ip>
```

Test DNS Resolution:

```
kubectl exec -it <pod-name> -- nslookup <service-name>
```

Inspect Network Policies:

```
kubectl describe networkpolicy
```

1. **Use tools like cURL or traceroute inside pods.**

111. How do you handle Kubernetes Version Compatibility?

Answer:

1. Use kubeadm for version upgrades.
2. Ensure all components (API server, kubelet, kubectl) are within one minor version difference.
3. Test upgrades in a staging environment.

112. What is Kubernetes ArgoCD?

Answer: ArgoCD is a declarative GitOps tool for Kubernetes. It continuously syncs cluster state with Git repositories.

Features:

- Application version control.
- Automated rollbacks.
- Multi-cluster management.

113. What are Kubernetes Helm Charts?

Answer: Helm Charts are collections of templates and values used to package Kubernetes applications.

Helm Commands:

Install a chart:

```
helm install <release-name> <chart-name>
```

Upgrade a release:

```
helm upgrade <release-name> <chart-name>
```

114. How do you implement NetworkPolicies in Kubernetes?

Answer: NetworkPolicies control traffic flow to/from pods.

Example YAML:

```yaml
apiVersion: networking.k8s.io/v1


kind: NetworkPolicy

metadata:

  name: allow-web

  namespace: default

spec:

  podSelector:

    matchLabels:

      app: web

  policyTypes:

  - Ingress

  ingress:

  - from:

    - podSelector:

        matchLabels:

          app: frontend

    ports:

    - protocol: TCP
    port: 80
```

115. What is Kubernetes kubeadm?

Answer: kubeadm is a tool to bootstrap Kubernetes clusters.

Key Commands:

Initialize a cluster:

```
kubeadm init
```

Join a node to a cluster:

```
kubeadm join <master-ip>:<port> --token <token>
```

116. What are Kubernetes ServiceAccount Tokens?

Answer: ServiceAccount tokens are credentials automatically mounted into pods, enabling them to interact with the Kubernetes API.

Key Points:

- Each namespace has a default ServiceAccount.
- Tokens are mounted at `/var/run/secrets/kubernetes.io/serviceaccount`.

Example: Create a ServiceAccount and associate it with a pod:

```
apiVersion: v1

kind: ServiceAccount

metadata:

  name: my-service-account
```

```
apiVersion: v1



kind: Pod

metadata:

  name: my-pod

spec:

  serviceAccountName: my-service-account

  containers:

  - name: app

    image: nginx
```

117. What is Kubernetes Custom Metrics API?

Answer: The Custom Metrics API allows Kubernetes to autoscale workloads based on custom application metrics.

Use Case: Scale based on queue length, requests per second, etc.

Steps:

1. **Install Prometheus Adapter.**
2. **Expose custom metrics using Prometheus.**
3. **Configure HPA to use these metrics.**

118. What is Kubernetes Ingress TLS Termination?

Answer: Ingress can terminate SSL/TLS traffic, offloading the SSL work from applications.

Example YAML:

```
apiVersion: networking.k8s.io/v1



kind: Ingress
metadata:
  name: tls-example
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  tls:
  - hosts:
    - example.com
      secretName: tls-secret
  Rules:

  - host: example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          Service:



            name: my-service
            port:
              number: 80
```

119. How do you back up and restore etcd in Kubernetes?

Answer: etcd is the backbone of Kubernetes cluster state. Regular backups are essential.

Backup Command:

```
ETCDCTL_API=3 etcdctl snapshot save backup.db \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key
```

Restore Command:

```
ETCDCTL_API=3 etcdctl snapshot restore backup.db \
  --data-dir=/var/lib/etcd-new
```

120. What is Kubernetes Horizontal Pod Autoscaler with External Metrics?

Answer: External metrics allow scaling based on metrics outside the cluster, such as cloud service metrics.

Steps:

1. Install an external metrics adapter (e.g., AWS CloudWatch).
2. Configure HPA to use external metrics:

```
metrics:

- type: External

  external:

    metricName: queue_messages

    targetValue: 100
```

121. What are Kubernetes Pod Security Standards?

Answer: Kubernetes has three security profiles for namespaces:

1. Privileged: No restrictions.
2. Baseline: Minimal restrictions for common use cases.
3. Restricted: Strong security enforcement.

Enable by setting namespace labels:

```
kubectl label namespace default
pod-security.kubernetes.io/enforce=baseline
```

122. What are Kubernetes Runtime Hooks?

Answer: Runtime hooks (alpha feature) allow executing custom logic during pod lifecycle events (e.g., post-start).

Example Use Case:

- Notify a monitoring system after pod creation.

123. What is Kubernetes Multi-Tenancy?

Answer: Multi-tenancy isolates resources and workloads for different users or teams.

Best Practices:

- **Use namespaces to separate workloads.**
- **Enforce quotas and limits per namespace.**
- **Use RBAC for access control.**

124. How do you secure Kubernetes with Network Policies?

Answer: Network Policies define allowed ingress and egress traffic between pods.

Example YAML:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
Metadata:
  name: restrict-traffic
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
  - Egress
```

```
Egress:
- to:
  - podSelector:
      matchLabels:
        app: database
  ports:
  - protocol: TCP
    port: 3306
```

125. How do you monitor Kubernetes Clusters?

Answer:

1. **Metrics Monitoring:**
   - **Use Prometheus + Grafana.**
   - **Install the Kubernetes Metrics Server.**
2. **Log Monitoring:**
   - **Use Fluentd, ELK Stack, or Loki.**
3. **Health Monitoring:**
   - **Use Kubernetes Dashboard or Lens.**
4. **Third-Party Tools:**
   - **Datadog, New Relic, or Dynatrace.**

126. What is Kubernetes Kube-proxy, and what are its modes?

Answer: Kube-proxy maintains network rules for Pods and Services.

Modes:

1. iptables: Uses iptables rules for load balancing.
2. ipvs: Uses IP Virtual Server for higher performance.
3. Userspace: Legacy mode, not commonly used.

127. What is the difference between StatefulSets and DaemonSets?

Answer:

- StatefulSet: Manages stateful applications with unique pod identities (e.g., databases).
- DaemonSet: Ensures one pod per node for system services (e.g., logging).

128. How do you handle Kubernetes Node Pressure?

Answer:

Check node conditions:

```
kubectl describe node <node-name>
```

1. Evict low-priority pods using priority classes.
2. Scale up nodes with the Cluster Autoscaler.

129. How do you create Kubernetes Custom Resources?

Answer:

Define a CustomResourceDefinition (CRD):

```
apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

metadata:

  name: widgets.example.com
```

```
Spec:



  group: example.com

  names:

    kind: Widget

    listKind: WidgetList

    plural: widgets

    singular: widget

  scope: Namespaced

  versions:

  - name: v1

    served: true

    storage: true
```

Create instances of the custom resource:

```
apiVersion: example.com/v1

kind: Widget

metadata:

  name: my-widget

spec:
  size: large
```

130. What are Kubernetes Pod Termination Grace Periods?

Answer: When a pod is deleted, Kubernetes allows time for cleanup before termination. The default is 30 seconds.

Set Grace Period:

```
kubectl delete pod <pod-name> --grace-period=10
```

131. What is the difference between Kubernetes Jobs and CronJobs?

Answer:

- Jobs: Run tasks once or until completion. Used for short-lived workloads.
- CronJobs: Schedule Jobs at specific times or intervals using cron syntax.

Example Job YAML:

```
apiVersion: batch/v1

kind: Job

Metadata:

  name: example-job

spec:

  template:

    spec:

      containers:
```

```
      - name: my-task
        image: busybox
        command: ["sh", "-c", "echo Hello, Kubernetes!"]
      restartPolicy: OnFailure
```

Example CronJob YAML:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: example-cronjob
spec:
  schedule: "*/5 * * * *"
  jobTemplate:
    Spec:
      Template:
        spec:
          containers:
          - name: cron-task
            image: busybox
            command: ["sh", "-c", "echo Running at $(date)"]

          restartPolicy: OnFailure
```

132. What is Kubernetes Pod Affinity?

Answer: Pod Affinity ensures that certain pods are scheduled close to others based on defined rules.

Example YAML:

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app
          operator: In
          Values:
          - frontend
      topologyKey: "kubernetes.io/hostname"
```

133. What is Kubernetes Helm Templating?

Answer: Helm templates enable dynamic Kubernetes manifest generation using variables.

Example `values.yaml`:

```yaml
replicaCount: 3
image:
  repository: nginx
  tag: latest
```

Example `deployment.yaml`:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
spec:
  replicas: {{ .Values.replicaCount }}
  template:
    Spec:
      containers:
      - name: app
        image: "{{ .Values.image.repository }}:{{
.Values.image.tag }}"
```

Render the template:

```
helm template my-release ./my-chart
```

134. How do you configure Kubernetes Logging?

Answer:

1. Basic Logging:
   - Use `kubectl logs` for pod logs.

Stream logs:

```
kubectl logs -f <pod-name>
```

2. Centralized Logging:
   - Use Fluentd to ship logs to Elasticsearch or Loki.
   - Visualize logs in Kibana or Grafana.

135. What is Kubernetes Eviction Policy?

Answer: Eviction policies manage how pods are removed from nodes under resource pressure.

Types of Evictions:

1. Soft Evictions: Triggered by thresholds (e.g., memory usage).
2. Hard Evictions: Force eviction when limits are exceeded.

Configure thresholds in kubelet:

```
evictionHard:

  memory.available: "100Mi"

  nodefs.available: "10%"
```

136. What is Kubernetes `kubectl debug`?

Answer: `kubectl debug` is a command for debugging running pods or nodes by attaching ephemeral containers or creating debug pods.

Examples:

Debug a pod with an ephemeral container:

```
kubectl debug pod/my-pod --image=busybox
```

Debug a node:

```
kubectl debug node/my-node --image=busybox
```

137. How do you secure Kubernetes with Pod Security Standards?

Answer:

1. Use Pod Security Admission:

Set `enforce` labels on namespaces:

```
kubectl label namespace default
pod-security.kubernetes.io/enforce=restricted
```

2. Use PodSecurityPolicies (Deprecated) or OPA Gatekeeper for custom rules.
3. Enforce:
   - Non-root user.
   - Limited capabilities (`CAP_NET_RAW`).
   - Read-only file systems.

138. What are Kubernetes Static Pods?

Answer: Static pods are managed directly by the kubelet without being bound to the API server. They are defined in a configuration directory on the node.

Example: Create a static pod configuration file at `/etc/kubernetes/manifests/nginx.yaml`:

```
apiVersion: v1

kind: Pod

metadata:

  name: nginx

spec:

  containers:

  - name: nginx

    image: nginx
```

139. What is Kubernetes kube-state-metrics?

Answer: kube-state-metrics generates cluster-level metrics for objects like pods, nodes, and deployments, useful for monitoring with Prometheus.

Install kube-state-metrics:

```
helm install kube-state-metrics
prometheus-community/kube-state-metrics
```

140. How do you use Kubernetes Secrets with Environment Variables?

Answer:

Create a secret:

```
kubectl create secret generic db-credentials
--from-literal=username=admin --from-literal=password=secret
```

Use it in a pod:

```
env:

- name: DB_USER

  valueFrom:

    secretKeyRef:

      name: db-credentials

      key: username

- name: DB_PASSWORD

  valueFrom:

    secretKeyRef:

      name: db-credentials

      key: password
```

141. What is the Kubernetes `kubectl apply --server-side` command?

Answer: This command applies changes to Kubernetes resources while leveraging server-side apply, which enables conflict detection and better management of shared fields.

Example:

```
kubectl apply --server-side -f deployment.yaml
```

142. What is Kubernetes Container Probes' `initialDelaySeconds`?

Answer: `initialDelaySeconds` specifies the delay before the probe starts checking the container after it has started.

Example:

```
livenessProbe:

  httpGet:

    path: /healthz

    port: 8080

  initialDelaySeconds: 5

  periodSeconds: 10
```

143. How do you ensure High Availability in Kubernetes?

Answer:

1. Control Plane:
   - Use multiple master nodes.
   - Use etcd clustering.
   - Front the API server with a load balancer.
2. Worker Nodes:
   - Distribute workloads across nodes.
   - Use PodDisruptionBudgets.
3. Self-Healing:
   - Rely on health checks and auto-scaling.

144. How do you upgrade a Kubernetes Cluster?

Answer:

1. Backup etcd.

Upgrade control plane:

```
kubeadm upgrade apply <version>
```

2. Upgrade kubelets and kubectl on nodes.
3. Validate workloads.

145. How does Kubernetes handle Load Balancing?

Answer:

1. Internal Load Balancing:
   ○ Handled by `kube-proxy` via Services (ClusterIP or NodePort).
2. External Load Balancing:
   ○ Achieved using Service type `LoadBalancer` or Ingress controllers.

146. What are Kubernetes Volumes and their Types?

Answer: Kubernetes Volumes provide storage for containers in pods. Types include:

1. EmptyDir: Temporary storage, deleted when the pod stops.
2. HostPath: Maps a directory on the host to the pod.
3. PersistentVolume (PV): Abstracts physical storage.
4. ConfigMap and Secret: Inject configuration or sensitive data.
5. Cloud-specific: AWS EBS, Azure Disk, GCE Persistent Disk.
6. CSI (Container Storage Interface): For third-party storage solutions.

Example PV YAML:

```
apiVersion: v1

kind: PersistentVolume
```

```
metadata:

  name: pv-example

spec:

  capacity:

    storage: 5Gi

  accessModes:

    - ReadWriteOnce

  hostPath:

    path: /data/pv
```

147. How do you enforce resource limits in Kubernetes?

Answer:

Use Resource Quotas to limit CPU, memory, or storage per namespace:

```
apiVersion: v1

kind: ResourceQuota

metadata:

  name: resource-limits

Spec:

  Hard:
    requests.cpu: "4"

    requests.memory: "16Gi"

    limits.cpu: "8"

    limits.memory: "32Gi"
```

Configure LimitRanges to set default and max resource limits for pods:

```
apiVersion: v1

kind: LimitRange

metadata:

  name: limit-range

spec:

  limits:

  - default:

      cpu: "500m"

      memory: "256Mi"

    max:

      cpu: "1"

      memory: "512Mi"

    type: Container
```

148. What is the Kubernetes API Aggregation Layer?

Answer: The API Aggregation Layer extends Kubernetes by adding custom APIs without modifying the core API server.

Use Case: Create additional APIs like custom monitoring endpoints.

Steps:

1. **Deploy an extension API server.**
2. **Register it using an APIService resource.**

149. What is Kubernetes ClusterIP vs. NodePort vs. LoadBalancer?

Answer:

1. **ClusterIP: Default service type, accessible only within the cluster.**
2. **NodePort: Exposes the service on a static port across all nodes.**
3. **LoadBalancer: Integrates with cloud provider's external load balancer for external traffic.**

Example NodePort YAML:

```
apiVersion: v1

kind: Service

metadata:

  name: nodeport-example

spec:

  type: NodePort

  Ports:
```

```
  - port: 80
    targetPort: 8080
    nodePort: 30001
  selector:
    app: my-app
```

150. What is Kubernetes PreStop Hook?

Answer: A PreStop Hook runs a custom command before a container stops, giving time to clean up resources.

Example YAML:

```
lifecycle:
  preStop:
    exec:
      command: ["/bin/sh", "-c", "echo Stopping... && sleep 10"]
```

151. How does Kubernetes handle multi-cluster management?

Answer:

1. **Centralized Tools:**
   - **Rancher**
   - **ArgoCD**
   - **Anthos**
2. **Kubernetes Federation:**

- Synchronize resources across multiple clusters.

3. Custom Solutions:
    - Use GitOps for consistent configuration.

152. How do you debug a Kubernetes CrashLoopBackOff Error?

Answer:

Check pod events:

```
kubectl describe pod <pod-name>
```

View logs:

```
kubectl logs <pod-name>
```

1. Test dependencies:
    - Verify secret/config maps.
    - Confirm external service connectivity.

Restart pod for troubleshooting:

```
kubectl delete pod <pod-name>
```

153. What is the purpose of Kubernetes ReplicaSets?

Answer: ReplicaSets ensure a specific number of pod replicas are running at all times.

Example YAML:

```
apiVersion: apps/v1

kind: ReplicaSet

metadata:

  name: frontend

Spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: app
        image: nginx
```

154. How do you roll back a failed Kubernetes Deployment?

Answer:

Check the deployment history:

```
kubectl rollout history deployment <deployment-name>
```

Roll back to the previous version:

```
kubectl rollout undo deployment <deployment-name>
```

Roll back to a specific revision:

```
kubectl rollout undo deployment <deployment-name>
--to-revision=<revision-number>
```

155. What are Kubernetes StorageClasses?

Answer: StorageClasses define the provisioner and parameters for dynamically provisioning PersistentVolumes.

Example YAML:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
  fsType: ext4
reclaimPolicy: Retain
```

156. How do you handle Kubernetes Service discovery?

Answer:

1. **Use ClusterIP Services to provide internal service access.**

Use DNS-based service discovery:

```
<service-name>.<namespace>.svc.cluster.local
```

2. **Use Ingress or LoadBalancer for external discovery.**

157. What is Kubernetes Horizontal Pod Autoscaler?

Answer: HPA adjusts the number of pods in a deployment based on resource usage like CPU or custom metrics.

Example YAML:

```
apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

  name: hpa-example

spec:

  scaleTargetRef:

    apiVersion: apps/v1

    kind: Deployment

    name: example-deployment

  minReplicas: 2

  maxReplicas: 10

  metrics:

  - type: Resource

    Resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

158. What is Kubernetes DaemonSet Update Strategy?

Answer: Defines how DaemonSet updates are applied to pods.

Strategies:

1. RollingUpdate (default): Update pods incrementally.
2. OnDelete: Updates pods only after manual deletion.

Example YAML:

```
updateStrategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
```

159. How do you secure Kubernetes Ingress?

Answer:

1. Use TLS for encrypted communication.
2. Restrict access with NetworkPolicies.
3. Enable rate-limiting via Ingress annotations.

Example TLS Ingress YAML:

```
apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: secure-ingress

spec:

  tls:

  - hosts:

    - example.com

      secretName: tls-secret

  rules:

  - host: example.com

    http:

      paths:

      - path: /

        pathType: Prefix

        backend:

          service:

            name: my-service

            Port:
              number: 80
```

160. What is Kubernetes Admission Controller?

Answer: Admission Controllers intercept API requests to validate or modify objects before they are persisted.

Examples:

1. ValidatingWebhook: Reject requests if rules are not met.
2. MutatingWebhook: Modify requests (e.g., add sidecars).

161. What are Kubernetes Init Containers?

Answer: Init containers run before the main application containers in a pod. They are used for initialization tasks like setting up configurations or waiting for a dependency to be ready.

Example YAML:

```
apiVersion: v1

kind: Pod

metadata:

  name: init-container-example

spec:

  initContainers:

  - name: init-myservice

    image: busybox
    command: ["sh", "-c", "echo Initializing... && sleep 10"]
  containers:
  - name: app-container
    image: nginx
```

162. How does Kubernetes handle Secrets Encryption?

Answer: By default, Kubernetes Secrets are base64-encoded. To improve security, you can enable encryption at rest.

Steps:

Configure the encryption provider:

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
- resources:
  - secrets
  providers:
  - aescbc:
      keys:
      - name: key1
```

```
        secret: <base64-encoded-key>
  - identity: {}
```

Update the API server:

```
--encryption-provider-config=/path/to/encryption-config.yaml
```

1. Rotate keys periodically.

163. What is Kubernetes OPA Gatekeeper?

Answer: OPA (Open Policy Agent) Gatekeeper enforces policies on Kubernetes resources using admission controllers.

Example Policy: Restrict pods from running as root:

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: disallow-root
spec:
  match:
    kinds:
    - apiGroups: [""]
      kinds: ["Pod"]
  Parameters:
    key: "runAsUser"
    regex: "^(?!0$).*"
```

164. How do you implement Kubernetes Canary Deployments?

Answer:

1. Deploy a new version (canary) alongside the existing version.
2. Route a small percentage of traffic to the canary version.
3. Gradually increase traffic if the canary version is stable.

Example with Service:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: canary
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: my-app
        version: canary
```

Adjust traffic using weights in Ingress or ServiceMesh (e.g., Istio).

165. What is Kubernetes NodeLocal DNS Cache?

Answer: NodeLocal DNS Cache improves DNS performance by caching queries on each node.

Enable NodeLocal DNS Cache:

Deploy the `nodelocaldns` addon:

```
kubectl apply -f
https://k8s.io/examples/admin/dns/nodelocaldns.yaml
```

1. Update `--cluster-dns` in kubelet to point to the local cache IP.

166. How do you troubleshoot Kubernetes High API Server Latency?

Answer:

Check etcd performance:

```
etcdctl endpoint status --write-out=table
```

Inspect API server metrics:

```
kubectl top pods -n kube-system
```

1. Use audit logs to identify slow or heavy requests.
2. Optimize resource usage:
   - Increase API server replicas.
   - Use `PriorityClasses` for critical workloads.

167. What is Kubernetes Service Mesh?

Answer: A Service Mesh manages service-to-service communication in a Kubernetes cluster, offering features like:

- Traffic routing.
- Observability (metrics, logs, tracing).
- Security (mTLS).

Popular Service Meshes:

- Istio
- Linkerd
- Consul

168. What is Kubernetes Vertical Pod Autoscaler (VPA)?

Answer: VPA automatically adjusts pod resource requests and limits based on usage.

Use Case: Workloads with unpredictable or varying resource needs.

Steps:

Install VPA:

```
kubectl apply -f https://github.com/kubernetes/autoscaler/releases/latest/download/vertical-pod-autoscaler.yaml
```

Annotate deployments to use VPA:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: vpa-example
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-app
  updatePolicy:
    updateMode: "Auto"
```

169. How do you handle Kubernetes Namespaces efficiently?

Answer:

1. **Use namespaces to isolate environments (e.g., dev, staging, production).**

Apply resource quotas to limit resource usage:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
  namespace: dev
spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    limits.cpu: "8"
```

Implement RBAC for fine-grained access control:

```
apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

  name: dev-rolebinding

  namespace: dev

subjects:

- kind: User

  name: dev-user

roleRef:

  kind: Role

  name: dev-role

  apiGroup: rbac.authorization.k8s.io
```

170. What is Kubernetes RBAC (Role-Based Access Control)?

Answer: RBAC controls access to Kubernetes resources based on roles and bindings.

Key Components:

1. **Role/ClusterRole: Define permissions.**
2. **RoleBinding/ClusterRoleBinding: Assign roles to users or groups.**

Example:

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: Role

metadata:

  namespace: dev

  name: pod-reader

rules:

- apiGroups: [""]

  resources: ["pods"]

  verbs: ["get", "list", "watch"]

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

  name: read-pods

  namespace: dev

subjects:

- kind: User

  name: dev-user

  apiGroup: rbac.authorization.k8s.io

roleRef:
  kind: Role

  name: pod-reader

  apiGroup: rbac.authorization.k8s.io
```

171. What are Kubernetes Dynamic Admission Controllers?

Answer: Dynamic Admission Controllers intercept API requests to enforce rules or modify objects before they are persisted.

Types:

1. **MutatingAdmissionWebhook: Modifies requests (e.g., injects sidecars).**
2. **ValidatingAdmissionWebhook: Validates requests (e.g., disallows pods running as root).**

Example Webhook YAML:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deny-root
webhooks:
- name: deny-root.example.com
  clientConfig:
    service:
      name: webhook-service
      namespace: default
      path: /validate
  rules:
  - apiGroups: [""]
```

```
    apiVersions: ["v1"]
    operations: ["CREATE"]
    resources: ["pods"]
  failurePolicy: Fail
```

172. What is Kubernetes Kubeadm?

Answer: Kubeadm is a tool that simplifies the process of setting up a Kubernetes cluster by automating the installation and configuration of cluster components.

Key Features:

- Initializes a Kubernetes control plane node.
- Joins nodes to an existing cluster.
- Manages cluster upgrades.

Example Commands:

Initialize the control plane:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Join a node to the cluster:

```
kubeadm join <master-ip>:<master-port> --token <token>
--discovery-token-ca-cert-hash sha256:<hash>
```

173. How do you perform a Rolling Update in Kubernetes?

Answer: A rolling update allows you to update the version of an application without downtime by incrementally replacing old pods with new ones.

Steps:

1. Update the container image in the Deployment manifest.

Apply the changes:

```
kubectl apply -f deployment.yaml
```

Monitor the rollout status:

```
kubectl rollout status deployment/<deployment-name>
```

Example: To update the image of a deployment:

```
kubectl set image deployment/<deployment-name> <container-name>=<new-image>:<tag>
```

174. What is Kubernetes Helm?

Answer: Helm is a package manager for Kubernetes that enables the deployment and management of applications through reusable packages called charts.

Key Features:

- Simplifies application deployment.
- Manages application versions.
- Facilitates application upgrades and rollbacks.

Basic Commands:

Install a chart:

```
helm install <release-name> <chart-name>
```

List releases:

```
helm list
```

Upgrade a release:

```
helm upgrade <release-name> <chart-name>
```

175. How do you secure Kubernetes Secrets?

Answer: Kubernetes Secrets store sensitive information, such as passwords and API keys.

Best Practices:

- Enable encryption at rest for secrets.
- Restrict access using RBAC policies.
- Avoid storing secrets in environment variables; use volume mounts instead.
- Regularly audit and rotate secrets.

Example: Create a secret:

```
kubectl create secret generic db-credentials
--from-literal=username=admin --from-literal=password=secret
```

Use the secret in a pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
  - name: test-container
    image: nginx
    env:
    - name: DB_USERNAME
      valueFrom:
        secretKeyRef:
          name: db-credentials
          key: username
    - name: DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-credentials
          key: password
```

176. What is Kubernetes Kubectl?

Answer: kubectl is the command-line interface (CLI) tool for interacting with the Kubernetes API server.

Common Commands:

Get cluster information:

```
kubectl cluster-info
```

List all pods in a namespace:

```
kubectl get pods -n <namespace>
```

Describe a resource:

```
kubectl describe <resource-type> <resource-name>
```

Apply a configuration file:

```
kubectl apply -f <file.yaml>
```

177. How do you monitor Kubernetes Clusters?

Answer: Monitoring is essential for maintaining the health and performance of a Kubernetes cluster.

Tools:

- Prometheus: Collects and stores metrics.
- Grafana: Visualizes metrics through dashboards.
- Kube-state-metrics: Exposes Kubernetes API server metrics.
- cAdvisor: Provides container resource usage and performance metrics.

Implementation Steps:

1. Deploy Prometheus to collect metrics.
2. Install Grafana and configure it to use Prometheus as a data source.
3. Set up dashboards in Grafana to visualize cluster and application metrics.

178. What is Kubernetes Ingress?

Answer: Ingress is an API object that manages external access to services within a Kubernetes cluster, typically HTTP and HTTPS.

Features:

- Defines rules for routing traffic to services.
- Supports virtual hosting (routing based on hostnames).
- Can terminate SSL/TLS.

Example Ingress Resource:

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

179. What are Kubernetes Finalizers?

Answer: Finalizers are metadata annotations that ensure specific cleanup actions are performed before a Kubernetes resource is deleted.

Use Case:

- Ensuring PersistentVolumes are detached before deleting a pod.
- Running custom cleanup scripts.

Example: Adding a finalizer to a resource:

```
metadata:
  finalizers:
  - kubernetes.io/pv-protection
```

180. How does Kubernetes handle Cluster Networking?

Answer: Kubernetes uses a flat networking model, ensuring that:

1. Every pod gets its own unique IP.
2. Pods can communicate without NAT (Network Address Translation).
3. Services provide stable endpoints for communication.

Networking Tools:

- CNI Plugins: Calico, Flannel, Weave Net.
- CoreDNS: Handles internal DNS for service discovery.

181. What is Kubernetes CoreDNS?

Answer: CoreDNS is the default DNS server in Kubernetes, providing service discovery by resolving names like my-service.my-namespace.svc.cluster.local.

Example Query:

```
kubectl exec -it <pod-name> -- nslookup my-service
```

182. What are Kubernetes Resource Requests and Limits?

Answer: Resource requests and limits control how much CPU and memory a pod can use.

Key Points:

- Requests: Minimum guaranteed resources for a container.

- Limits: Maximum resources a container can use.

Example YAML:

```
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

183. What is the purpose of Kubernetes Custom Resource Definitions (CRDs)?

Answer: CRDs enable users to define and manage custom resources in Kubernetes, extending its functionality.

Use Case:

- Managing custom workloads like databases or monitoring tools.

Example YAML:

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: widgets.example.com
spec:
  group: example.com
  names:
    kind: Widget
    plural: widgets
    singular: widget
  scope: Namespaced
  versions:
  - name: v1
    served: true
    storage: true
```

184. What are Kubernetes DaemonSet Use Cases?

Answer: DaemonSets ensure that a pod runs on every (or a subset of) nodes in a cluster.

Use Cases:

- Collecting logs (e.g., Fluentd, Logstash).
- Node monitoring (e.g., Prometheus Node Exporter).
- Running system-level applications (e.g., CNI plugins).

185. How do you manage Kubernetes Config Drift?

Answer:

1. Use GitOps tools like ArgoCD or Flux to manage and synchronize configurations.
2. Regularly use kubectl diff or CI pipelines to compare desired and actual states.
3. Automate remediation with continuous reconciliation tools.

186. What is Kubernetes Cluster Autoscaler?

Answer: Cluster Autoscaler automatically adjusts the number of nodes in a cluster based on pod requirements.

Key Features:

- Scales up when pods cannot be scheduled due to resource constraints.
- Scales down underutilized nodes without evicting critical pods.

Configuration:

```
kubectl apply -f cluster-autoscaler.yaml
```

187. What is the role of Kubernetes API Server?

Answer: The API Server is the central component of the Kubernetes control plane, responsible for:

- Exposing the Kubernetes API.
- Validating and processing requests.
- Acting as a communication hub for all cluster components.

188. How do you troubleshoot Kubernetes Node NotReady State?

Answer:

Check node status:
```
kubectl describe node <node-name>
```
Inspect kubelet logs:
```
journalctl -u kubelet
```

1. Verify network connectivity: Ensure the node can communicate with the control plane.

Check disk pressure or resource usage:

```
kubectl top nodes
```

189. What are Kubernetes Taints and Tolerations?

Answer:

- Taints: Prevent pods from being scheduled on specific nodes unless they have matching tolerations.
- Tolerations: Allow pods to bypass node taints.

Example: Taint a node:

```
kubectl taint nodes node1 key=value:NoSchedule
```

Add a toleration to a pod:

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

190. What is the Kubernetes Audit Logging?

Answer: Audit logging records all API server requests for security and troubleshooting purposes.

Enable Audit Logs:

Update the API server:

```
--audit-log-path=/var/log/kubernetes/audit.log
--audit-policy-file=/etc/kubernetes/audit-policy.yaml
```

Define an audit policy:

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: RequestResponse
  resources:
  - group: ""
    resources: ["pods"]
```

191. How do you debug Kubernetes Networking Issues?

Answer:

Test pod connectivity:

```
kubectl exec -it <pod-name> -- ping <target-ip>
```

Verify DNS resolution:

```
kubectl exec -it <pod-name> -- nslookup <service-name>
```

Inspect NetworkPolicies:

```
kubectl describe networkpolicy
```

192. What is Kubernetes LoadBalancer Service?

Answer: A LoadBalancer service integrates with a cloud provider's load balancer to expose services externally.

Example YAML:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalancer
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: my-app
```

193. What are Kubernetes Admission Webhooks?

Answer: Admission webhooks intercept and validate or modify API requests before they are persisted.

Types:

1. Mutating Webhooks: Modify requests.
2. Validating Webhooks: Validate requests.

194. What is Kubernetes Sidecar Pattern?

Answer: A sidecar is a helper container that runs alongside the main application container in the same pod.

Use Cases:

- Log aggregation.
- Monitoring.
- Service mesh proxies.

195. How do you configure Kubernetes Ingress Annotations?

Answer: Ingress annotations customize behavior like SSL redirection and rate limiting.

196. What are Kubernetes Network Policies?

Answer: Network Policies control the flow of traffic to and from pods based on rules.

Key Features:

- Define allowed ingress and egress traffic.
- Use labels to target pods.

Example NetworkPolicy: Allow traffic to a specific pod from a given namespace:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-frontend
spec:
  podSelector:
    matchLabels:
      app: backend
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: frontend-namespace
    Ports:
```

```
  - protocol: TCP
    port: 80
```

197. What is Kubernetes RollingUpdate Strategy?

Answer: RollingUpdate strategy incrementally replaces old pods with new pods during updates to minimize downtime.

Configuration Example:

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
```

Commands:

Monitor rollout:

```
kubectl rollout status deployment <deployment-name>
```

Pause rollout:

```
kubectl rollout pause deployment <deployment-name>
```

Resume rollout:

```
kubectl rollout resume deployment <deployment-name>
```

198. What is Kubernetes Ephemeral Containers?

Answer: Ephemeral containers are temporary containers added to a running pod for debugging purposes without restarting it.

Command Example:

```
kubectl debug pod/<pod-name> --image=busybox
```

Use Case:

- Debugging issues in long-running pods without disrupting their state.

199. What are Kubernetes Pod Conditions?

Answer: Pod conditions provide details about the state of a pod, including readiness and scheduling status.

Common Conditions:

1. PodScheduled: Pod is scheduled on a node.
2. Ready: Pod is ready to handle requests.
3. ContainersReady: All containers in the pod are ready.

Check Pod Conditions:

```
kubectl describe pod <pod-name>
```

200. How do you use Kubernetes `kubectl cp`?

Answer: The `kubectl cp` command copies files between a pod and a local system.

Examples:

Copy file from local to pod:

```
kubectl cp ./localfile.txt <pod-name>:/path/inside/container
```

- **Copy file from pod to local:**
  ```
  kubectl cp <pod-name>:/path/inside/container
  ./localfile.txt
  ```