Kubernetes scenario-based questions

I. Core Kubernetes Concepts (50 Questions)

These scenarios focus on the fundamental building blocks of Kubernetes.

A. Pods & Containers (10 Questions)

1. **CrashLoopBackOff:** A critical Pod is repeatedly restarting with `CrashLoopBackOff`. Describe your step-by-step debugging process.
2. **ImagePullBackOff:** A new Pod is stuck in `ImagePullBackOff`. What are the common causes, and how would you resolve them?
3. **Pending Pod:** A Pod remains in the `Pending` state. What are the potential reasons, and how would you investigate?
4. **Multi-Container Pod:** You have an application that requires a sidecar container for logging/monitoring. How would you design a Pod for this, and what are the benefits?
5. **Init Containers:** Your application needs to perform a pre-start check (e.g., waiting for a database to be ready) before the main container starts. How would you implement this?
6. **Resource Limits & Requests:** An application is consuming too much CPU on a node, impacting other Pods. How would you mitigate this using Pod specifications? What's the difference between requests and limits?
7. **Pod Eviction:** A Pod was evicted from a node. What are the common reasons for eviction, and how would you prevent it?
8. **Ephemeral Storage:** Your Pod needs temporary scratch space during its lifecycle. How would you configure ephemeral storage, and what are its limitations?
9. **Pod Liveness/Readiness Probes:** An application deploys successfully but traffic isn't routed to it, or it serves errors. How would you use probes to ensure application health and proper traffic routing?
10. **Tightly Coupled Services:** You have two containers that are tightly coupled and must run on the same node and share network/storage. How would you deploy them?

B. Deployments, ReplicaSets, DaemonSets, StatefulSets (15 Questions)

1. **Rolling Update Failure:** You initiated a rolling update, and the new version is failing. How would you roll back to the previous stable version with minimal downtime?
2. **Zero-Downtime Deployment:** Design a strategy for deploying a new version of a stateless web application with zero downtime.
3. **Canary Deployment:** How would you implement a canary deployment strategy for a new feature, directing a small percentage of traffic to the new version before a full rollout?

4. **Blue/Green Deployment:** Explain how you would perform a blue/green deployment for an application running in Kubernetes. What are the pros and cons compared to rolling updates?
5. **Scaling a Stateless App:** Your web application is experiencing high load during peak hours. How would you automatically scale it up and down?
6. **Deploying a Logging Agent:** You need to deploy a logging agent that runs on *every* node in your cluster. Which workload object would you use, and why?
7. **Persistent Database Deployment:** You need to deploy a PostgreSQL database in Kubernetes. Which workload object is most suitable, and what specific challenges does it address for stateful applications?
8. **Database Scaling (StatefulSet):** Your database deployed with a StatefulSet is experiencing high read load. How would you scale it? (Focus on Read Replicas concepts within Kubernetes)
9. **Ordered Deployment/Scaling:** You have a distributed application where components must start and scale in a specific order. How can StatefulSets help with this?
10. **Handling Node Failures with Deployments:** A node fails. What happens to the Pods managed by a Deployment, and how does Kubernetes recover them?
11. **Pod Disruption Budgets (PDB):** You need to perform node maintenance, but want to ensure a minimum number of replicas for a critical application. How would you configure this?
12. **Deployment History & Revisions:** You've made multiple changes to a Deployment. How can you view the history of changes and revert to a specific revision?
13. **DaemonSet vs. Init Container for Node-level tasks:** When would you choose a DaemonSet over an Init Container (or vice-versa) for tasks that need to run on all nodes?
14. **Customizing Rollout Strategy:** You want more fine-grained control over your rolling updates (e.g., custom delays, pre-check hooks). How might you achieve this beyond basic Deployment settings? (Hint: Operators or specialized tools).
15. **Auto-healing DaemonSet:** A DaemonSet Pod on a specific node keeps failing. How would you troubleshoot this specific instance?

C. Services & Networking (15 Questions)

1. **Exposing a Web Application:** You have a Deployment for a web application. How would you expose it to external users, providing a stable IP address?
2. **Internal Communication:** Your frontend Pods need to communicate with backend API Pods within the same cluster. How would you enable this securely and reliably?
3. **LoadBalancer vs. Ingress:** Explain the difference between a `Service` of type `LoadBalancer` and an `Ingress`. When would you choose one over the other?
4. **Ingress Routing:** You have multiple microservices, and you want to route traffic to different services based on URL paths or hostnames. How would you configure Ingress for this?
5. **Headless Service:** You're deploying a custom database cluster. How would you use a Headless Service, and why is it important for stateful applications?

6. **Network Policy:** You need to restrict communication between Pods in different namespaces or between specific application tiers. How would you implement this?
7. **DNS Resolution Issues:** Pods in your cluster are unable to resolve internal service names. How would you troubleshoot this?
8. **ExternalName Service:** You want to point a Kubernetes service to an external database not running in the cluster. How would you do this?
9. **NodePort vs. ClusterIP:** Explain the differences between `NodePort` and `ClusterIP` service types. When would you use a `NodePort`?
10. **Egress Control:** How would you restrict outbound traffic from your Pods to only allow access to specific external domains?
11. **Service Mesh Introduction:** When would you consider introducing a Service Mesh (e.g., Istio, Linkerd) into your Kubernetes cluster, and what benefits would it provide for networking?
12. **Troubleshooting Service Connectivity:** A service is deployed, but Pods cannot reach it. What steps would you take to diagnose connectivity problems?
13. **IP Address Management:** How does Kubernetes assign IP addresses to Pods, and what is the underlying network model (flat network)?
14. **Container Network Interface (CNI):** Briefly explain the role of a CNI plugin in Kubernetes networking. How does it enable Pod-to-Pod communication?
15. **Path-based vs. Host-based Ingress:** Elaborate on the differences and use cases for path-based versus host-based routing with Ingress.

D. Storage (10 Questions)

1. **Persistent Volume (PV) & Persistent Volume Claim (PVC):** Your application needs persistent storage that survives Pod restarts. Explain how PVs and PVCs work, and how your application would consume storage.
2. **Dynamic Provisioning:** You want storage to be automatically provisioned when a PVC is created. How is this achieved in Kubernetes?
3. **Storage Class:** Explain the purpose of a StorageClass. How would you define one for a specific type of storage (e.g., SSD vs. HDD)?
4. **Volume Snapshots:** Your stateful application requires regular backups. How can Kubernetes volume snapshots help with this?
5. **ReadWriteMany vs. ReadWriteOnce:** Explain the different access modes for Persistent Volumes and when you would use `ReadWriteMany`.
6. **StatefulSet Persistent Storage:** How does a StatefulSet ensure stable, persistent storage for each of its replicas, even after rescheduling?
7. **Ephemeral Volumes:** Your Pod needs a temporary, isolated filesystem for its current execution. How would you use `emptyDir` or `hostPath` volumes? What are the considerations/warnings for `hostPath`?
8. **Expanding Persistent Volumes:** You have a PVC that is running out of space. How can you expand it without downtime?
9. **Troubleshooting PVC Pending:** A PVC remains in the `Pending` state. What are common reasons, and how would you debug this?

10. **Storage for Log Aggregation:** Design a strategy for storing large volumes of application logs generated by Pods, ensuring durability and retrievability.

II. Advanced Kubernetes Concepts & Operations (70 Questions)

These delve into more complex aspects, including security, scaling, monitoring, and advanced deployment patterns.

A. Security & Access Control (15 Questions)

1. **RBAC (Role-Based Access Control):** A new team needs read-only access to resources in a specific namespace. How would you configure RBAC for them?
2. **Service Accounts:** Explain the purpose of Service Accounts. How do Pods use them to interact with the Kubernetes API?
3. **Secrets Management:** Your application needs to access a database password. How would you securely store and inject this secret into your Pods? What are the limitations of native Kubernetes Secrets?
4. **Pod Security Standards/Admission Controllers:** How would you enforce security best practices for Pods (e.g., preventing privileged containers, restricting root access)?
5. **Network Policies for Egress:** You need to allow Pods in a specific namespace to communicate only with certain external IP addresses or DNS names. How would you implement this using Network Policies?
6. **Image Security:** How would you ensure that only trusted container images are deployed into your cluster? (e.g., Image signing, Admission Controllers).
7. **Vulnerability Scanning:** Describe your strategy for scanning container images and running Pods for known vulnerabilities.
8. **Auditing Kubernetes API:** How would you monitor and audit API calls within your Kubernetes cluster for security and compliance purposes?
9. **Securing the Kubernetes API Server:** What are key considerations for securing the API server endpoint?
10. **Encrypting ETCD:** Why is it crucial to encrypt data at rest in etcd, and how is this typically achieved?
11. **Third-Party Secret Managers:** When would you consider using external secret management solutions (e.g., HashiCorp Vault, AWS Secrets Manager) instead of native Kubernetes Secrets? How would you integrate them?
12. **Kubernetes Dashboard Security:** What are the security implications of exposing the Kubernetes Dashboard, and how would you secure it?
13. **Privileged Containers:** Explain why running privileged containers is a security risk and how you would prevent it in a production environment.
14. **CIS Benchmarks for Kubernetes:** Are you familiar with security benchmarks for Kubernetes? How would you apply them?
15. **Supply Chain Security:** How do you ensure the integrity and security of your application from source code to deployment in Kubernetes?

B. Scaling & Performance (15 Questions)

1. **Horizontal Pod Autoscaler (HPA):** Your application experiences fluctuating load. Configure HPA to automatically scale your Pods based on CPU utilization and a custom metric.
2. **Vertical Pod Autoscaler (VPA):** When would you consider using VPA, and what are its pros and cons compared to HPA?
3. **Cluster Autoscaler:** Your HPA is scaling Pods, but new nodes aren't being added. What component is missing, and how does it work?
4. **Karpenter (or similar node autoscaler):** Discuss the advantages of Karpenter over Cluster Autoscaler in certain scenarios.
5. **Resource Overcommitment:** How would you approach resource requests and limits to maximize cluster utilization while preventing resource starvation for critical applications?
6. **Pod Affinity/Anti-affinity:** You need to ensure certain Pods run on different nodes (anti-affinity) or prefer to run on the same node (affinity). How would you configure this?
7. **Node Selectors & Taints/Tolerations:** You have specialized nodes (e.g., with GPUs) and want to ensure only specific workloads run on them. How would you achieve this?
8. **Troubleshooting Performance Bottlenecks:** Your application is running slowly despite appearing healthy. How would you diagnose performance bottlenecks within Kubernetes?
9. **Throttling & OOMKilled:** A Pod is constantly being throttled or OOMKilled. How would you diagnose and resolve these issues?
10. **Eviction Thresholds:** How can you configure node eviction thresholds to manage resource pressure proactively?
11. **Service Load Balancing Algorithms:** Does Kubernetes allow configuring different load balancing algorithms for Services? If so, how?
12. **Network Latency Troubleshooting:** Users are complaining about slow application response times. How would you trace network latency within your Kubernetes cluster?
13. **Connection Draining on Pod Termination:** How do you ensure that a terminating Pod doesn't abruptly drop active connections, causing errors for users?
14. **Custom Metrics for HPA:** Your application's scaling needs are based on message queue length, not CPU. How would you use custom metrics with HPA?
15. **Scaling StatefulSets:** What specific challenges do you face when scaling StatefulSets, and how do you address them?

C. Monitoring, Logging & Alerting (15 Questions)

1. **Centralized Logging:** Design a centralized logging solution for your Kubernetes cluster, collecting logs from all Pods and nodes.
2. **Metrics Collection:** How would you collect application and cluster-level metrics in Kubernetes? Which open-source tools are commonly used?
3. **Alerting Strategy:** Design an alerting strategy for critical Kubernetes cluster events and application-specific issues.

4. **Troubleshooting Application Logs:** Your application is throwing errors, but the Pod keeps restarting. How would you retrieve and analyze its logs, even from previous instances?
5. **Debugging Node-level Issues:** A node is showing high CPU usage, but no specific Pod is identified as the culprit. How would you investigate node-level resource consumption?
6. **Prometheus & Grafana:** Explain the architecture of a Prometheus-Grafana stack for Kubernetes monitoring. What role does `kube-state-metrics` play?
7. **Distributed Tracing:** When would you implement distributed tracing in a Kubernetes microservices environment, and what tools would you consider?
8. **Cost Monitoring:** How would you monitor and attribute costs within your Kubernetes cluster to different teams or applications?
9. **Health Checks vs. Monitoring:** Explain the difference between Kubernetes liveness/readiness probes (health checks) and external monitoring solutions.
10. **Log Retention:** Your company has a compliance requirement for log retention. How would you ensure logs are stored for a specific period in your logging solution?
11. **Custom Metrics in CloudWatch:** If using EKS, how would you export custom application metrics from Kubernetes to AWS CloudWatch?
12. **Alert Fatigue:** How do you avoid alert fatigue when setting up monitoring and alerting for a large Kubernetes cluster?
13. **Service Level Indicators (SLIs) & Service Level Objectives (SLOs):** How can you define and monitor SLIs and SLOs for applications running in Kubernetes?
14. **Event Monitoring:** How do you monitor Kubernetes events (e.g., Pod scheduling, image pulls, node failures) for operational insights?
15. **Debugging Network Latency with Monitoring Tools:** How can you use metrics from your monitoring system to pinpoint network latency issues between services?

D. CI/CD & GitOps (15 Questions)

1. **CI/CD Pipeline Design:** Design a CI/CD pipeline for deploying a new microservice to Kubernetes, from code commit to production deployment.
2. **Image Registry Integration:** How would you integrate a private container image registry (e.g., Docker Hub, ECR, GCR) into your CI/CD pipeline and Kubernetes cluster?
3. **Helm Charts:** How would you package and deploy a complex application with multiple Kubernetes resources using Helm? What are the benefits?
4. **Automated Rollbacks in CI/CD:** How would you integrate automated rollback capabilities into your CI/CD pipeline in case of a failed deployment?
5. **GitOps Principles:** Explain GitOps and how you would implement it for managing your Kubernetes cluster configurations and application deployments.
6. **Argo CD / Flux CD:** Compare and contrast Argo CD and Flux CD. When would you choose one over the other for a GitOps setup?
7. **Kustomize:** When would you use Kustomize instead of (or in conjunction with) Helm for managing Kubernetes configurations?
8. **Testing in CI/CD:** How would you incorporate various types of tests (unit, integration, end-to-end) into your Kubernetes CI/CD pipeline?

9. **Environment Promotion:** How would you manage promoting application versions through different environments (dev, staging, prod) in a CI/CD pipeline for Kubernetes?
10. **Secrets in CI/CD:** How do you handle sensitive information (e.g., cloud credentials) within your CI/CD pipeline when interacting with Kubernetes?
11. **Managing Database Migrations:** Your application requires database schema changes with new deployments. How would you automate and manage these migrations safely within a CI/CD pipeline for a Kubernetes-deployed database?
12. **Container Image Tagging:** Describe a robust strategy for tagging container images in your CI/CD pipeline to ensure traceability and reproducibility.
13. **Service Account for CI/CD:** How would you configure a Service Account with minimal permissions for your CI/CD tool to deploy to Kubernetes?
14. **Immutable Infrastructure:** How does Kubernetes, combined with CI/CD, enable the concept of immutable infrastructure?
15. **Rollout Strategies with CI/CD:** Discuss how you would implement different rollout strategies (e.g., rolling update, canary, blue/green) within your CI/CD pipeline using Kubernetes capabilities or external tools.

E. Cluster Management & Operations (10 Questions)

1. **Node Maintenance:** You need to perform maintenance on a Kubernetes node (e.g., OS updates). How would you safely drain and cordon the node?
2. **Cluster Upgrades:** Describe your process for upgrading a Kubernetes cluster (e.g., from Kubernetes 1.28 to 1.29) with minimal downtime.
3. **Backup & Restore:** How would you back up the Kubernetes etcd data and restore a cluster from a backup?
4. **Multi-Cluster Management:** When would you consider managing multiple Kubernetes clusters? What tools or strategies would you use?
5. **Disaster Recovery Plan:** Design a disaster recovery plan for your Kubernetes cluster and the applications running on it.
6. **Custom Resource Definitions (CRDs) & Operators:** When would you create a Custom Resource Definition (CRD) and a Custom Controller (Operator) in Kubernetes? Provide an example.
7. **Tear Down and Rebuild:** How would you tear down and rebuild a Kubernetes cluster from scratch using Infrastructure as Code?
8. **Managing Certificate Rotations:** How do you handle certificate rotations for internal and external components within your Kubernetes cluster?
9. **Troubleshooting Control Plane Issues:** The API server is unresponsive. How would you diagnose and recover the Kubernetes control plane?
10. **Admission Controllers:** Explain the purpose of admission controllers and provide an example of how you might use a validating admission controller.

III. Troubleshooting Scenarios (40 Questions)

These focus specifically on diagnosing and resolving problems.

A. Pod & Container Troubleshooting (10 Questions)

1. A Pod is in `Error` state. What are your first steps to debug?
2. A Pod frequently restarts and you see `OOMKilled` in the logs. How do you address this?
3. Your application within a Pod is unresponsive, but the Pod itself is still `Running`. How do you investigate?
4. A Pod is consuming excessive network bandwidth. How do you identify the cause and mitigate it?
5. Logs from a specific Pod are not appearing in your centralized logging system. How would you troubleshoot this?
6. You see "Readiness probe failed" messages in your Pod events. What does this mean, and how do you fix it?
7. A Pod needs to access a file that should be present at a certain path, but it's missing. How would you debug volume mounting issues?
8. Your application needs a specific environment variable, but it's not being set in the container. How would you debug ConfigMap/Secret injection issues?
9. A Pod is stuck in `ContainerCreating`. What are the common causes and how would you resolve them?
10. You need to get a shell into a running Pod to debug an application issue. How would you do this securely?

B. Deployment & Rollout Troubleshooting (10 Questions)

1. A `Deployment` update is stuck, and new Pods are not coming up. How do you identify the root cause?
2. You performed a rolling update, and now users are reporting 500 errors. How would you quickly roll back and diagnose the issue?
3. A new Deployment version is causing high CPU usage across the cluster. How would you identify this and revert?
4. Your HPA is not scaling Pods despite high CPU. What could be the issue, and how would you troubleshoot HPA?
5. A `StatefulSet` Pod fails to restart on a new node after a node failure. What specific considerations would you have for debugging?
6. A `DaemonSet` Pod is not running on one specific node. How do you investigate why it's not scheduled?
7. Your `Deployment` is constantly in a "progressing" state but never completes. What could be the reasons?
8. You want to temporarily pause a `Deployment` rollout to investigate an issue without reverting. How do you do this?
9. A new version of your application introduced a bug that only manifests after a long time. How would you use a canary deployment to catch this earlier?

10. You need to get detailed events about a `Deployment`'s rollout. What commands would you use?

C. Networking Troubleshooting (10 Questions)

1. Pods cannot communicate with each other within the same namespace. How do you start troubleshooting network connectivity?
2. Your external load balancer (exposed via a `LoadBalancer` Service) is not routing traffic to your Pods. How do you debug the service-to-Pod connectivity?
3. Users cannot access your application via Ingress. How do you troubleshoot Ingress controller issues and routing rules?
4. DNS resolution for external services (e.g., `google.com`) is failing from within your Pods. How do you debug external DNS issues?
5. A specific `NetworkPolicy` is preventing legitimate traffic. How do you debug and identify the problematic policy?
6. You suspect a CNI plugin issue is causing network problems. How would you investigate and gather information?
7. Traffic from a specific external IP address is not reaching your application, but other IPs are working. How do you troubleshoot external access control?
8. You've changed a Service's selector, and now traffic is not reaching the correct Pods. How do you revert or fix this?
9. How would you capture network traffic (e.g., using `tcpdump`) from within a Pod for detailed analysis?
10. Your `Service` is showing a pending external IP. What could be the common causes for this in a cloud environment?

D. Storage Troubleshooting (10 Questions)

1. A `PVC` is stuck in `Pending`. How do you determine why it's not binding to a `PV`?
2. An application reports "disk full" errors, but its `PVC` shows available space. How would you troubleshoot this discrepancy?
3. Data written to a `PersistentVolume` is not persisting across Pod restarts. What could be the reason?
4. You've expanded a `PVC`, but the Pod still sees the old size. What steps are needed to apply the size change?
5. An application using `hostPath` is failing on a new node. What potential problems can `hostPath` introduce, and how would you fix it?
6. Your storage class is incorrectly configured, leading to provisioning failures. How would you debug this?
7. A `Pod` cannot mount a `Volume`. How do you investigate `VolumeMount` errors?
8. You are getting "permission denied" errors when writing to a mounted volume from within a container. How do you troubleshoot file permissions?
9. A `StorageClass` refers to a non-existent provisioner. How would you correct this?

10. You need to recover data from a deleted `PersistentVolume`. What is your strategy, assuming you have backups?

IV. Architecture & Design Scenarios (40 Questions)

These require you to design solutions using Kubernetes best practices.

A. Application Design & Deployment (15 Questions)

1. **Microservices Architecture:** Design a Kubernetes architecture for a new microservices application with a frontend, several backend services, and a database.
2. **Stateless Web Application:** Design a highly available and scalable deployment for a stateless web application.
3. **Stateful Application (e.g., Kafka/Elasticsearch):** Design a Kubernetes architecture for a distributed, stateful application like Kafka or Elasticsearch. What specific Kubernetes features would you leverage?
4. **Batch Processing Workload:** You have a batch processing job that runs periodically and needs to process large datasets. How would you deploy this using Kubernetes?
5. **Event-Driven Architecture:** Design an event-driven application using Kubernetes, incorporating message queues or streaming platforms.
6. **Multi-Tenant Application:** Design a multi-tenant application on a single Kubernetes cluster, ensuring isolation between tenants.
7. **Legacy Application Containerization:** You need to containerize and deploy a legacy monolithic application to Kubernetes. What challenges would you anticipate, and how would you address them?
8. **Choosing a Database:** When would you run a database inside Kubernetes versus using a managed database service outside the cluster? Justify your choice.
9. **External Dependencies:** Your application depends on external services (e.g., external APIs, SaaS products). How would you manage these dependencies from within Kubernetes?
10. **Application Observability:** Design an observability strategy (logging, metrics, tracing) for a new application deployed on Kubernetes.
11. **Cost-Optimized Architecture:** Design a cost-optimized Kubernetes architecture for a non-critical development environment.
12. **High-Security Application:** Design a Kubernetes architecture for an application handling sensitive financial data, focusing on security aspects.
13. **GitOps for Application Deployment:** How would you design a GitOps workflow for managing application deployments across multiple environments?
14. **Helm Chart Design:** You need to create a Helm chart for a complex application. What considerations would you have for templating, dependencies, and values?
15. **Cross-Region/Multi-Cloud Deployment:** Discuss the challenges and strategies for deploying a single application across multiple Kubernetes clusters in different regions or clouds.

B. Cluster Architecture & Management (15 Questions)

1. **Kubernetes Cluster on a Cloud Provider (EKS/GKE/AKS):** Design a production-ready Kubernetes cluster on your preferred cloud provider, including networking, worker node configuration, and control plane considerations.
2. **On-Premises Kubernetes Cluster:** If deploying Kubernetes on-premises, what are the key differences and challenges compared to cloud-managed services?
3. **HA Control Plane:** Design a highly available Kubernetes control plane. What components need to be redundant?
4. **Multi-AZ Deployment:** Design a Kubernetes cluster that spans multiple Availability Zones for high availability.
5. **Network Topology for Kubernetes:** Design a VPC/network topology suitable for a Kubernetes cluster, including public/private subnets, NAT gateways, and connectivity options.
6. **Resource Management Strategy:** Develop a resource management strategy for a large Kubernetes cluster with multiple teams and applications.
7. **Cost Management Strategy:** Outline a comprehensive strategy for managing and optimizing costs in a Kubernetes environment.
8. **Security Hardening:** What steps would you take to harden the security of a new Kubernetes cluster from scratch?
9. **Governance & Compliance:** How would you enforce governance and compliance policies (e.g., resource quotas, security policies) across a large Kubernetes cluster?
10. **Customizing kube-apiserver/kubelet:** When and why would you need to customize the configuration of core Kubernetes components like the API server or Kubelet?
11. **Upgrade Strategy:** Design an upgrade strategy for the Kubernetes control plane and worker nodes, minimizing downtime.
12. **Monitoring Stack Selection:** You need to choose a monitoring stack for a new Kubernetes cluster. Compare and contrast two popular options (e.g., Prometheus/Grafana vs. cloud-native options).
13. **Logging Stack Selection:** You need to choose a logging stack. Compare and contrast two popular options (e.g., Fluentd/Elasticsearch/Kibana vs. cloud-native options).
14. **GitOps for Cluster Configuration:** How would you manage the configuration of your Kubernetes cluster itself using GitOps principles?
15. **DR for Cluster:** Design a disaster recovery plan for the Kubernetes cluster control plane itself.

C. Kubernetes Ecosystem & Tooling (10 Questions)

1. **Service Mesh Use Cases:** Beyond basic traffic management, describe advanced use cases for a service mesh in Kubernetes.
2. **Kubernetes Operators for Custom Resources:** You have a custom application that needs complex lifecycle management. How would you leverage Kubernetes Operators?
3. **Cloud-Native Observability Tools:** Compare open-source observability tools (Prometheus, Grafana, Jaeger) with cloud-native managed services (e.g., CloudWatch, Stackdriver).

4. **Policy Enforcement:** How would you enforce policies (e.g., disallowing privileged containers, enforcing specific labels) across your Kubernetes cluster using policy engines?
5. **Chaos Engineering:** Explain the concept of Chaos Engineering in a Kubernetes environment and how you would implement it.
6. **Serverless on Kubernetes:** How can you run serverless functions (e.g., Knative) on Kubernetes, and what are the benefits?
7. **Kubernetes and Machine Learning Workloads:** How is Kubernetes suitable for managing ML workloads, and what specialized tools are available (e.g., Kubeflow)?
8. **External Secrets Operators:** Elaborate on the need for and implementation of an External Secrets Operator.
9. **Cost Optimization Tools:** Name and describe Kubernetes-native tools or practices for cost optimization.
10. **Comparing Kubernetes with other Orchestrators:** When would you choose Kubernetes over other container orchestrators (e.g., Docker Swarm, Nomad)?

V. Behavioral & Experiential Questions (10 Questions)

These questions aim to understand your practical experience and problem-solving approach in a team setting.

1. Describe a challenging Kubernetes issue you faced in a production environment. How did you diagnose it, and what was the resolution?
2. Tell me about a time you had to optimize resource usage in a Kubernetes cluster. What metrics did you look at, and what changes did you make?
3. How do you stay up-to-date with the rapidly evolving Kubernetes ecosystem?
4. Describe a time you had to implement a new Kubernetes feature or tool. What was your process?
5. How do you approach documenting your Kubernetes deployments and configurations for other team members?
6. Tell me about a time you had to convince a team or management to adopt Kubernetes or a new Kubernetes pattern. What challenges did you face?
7. How do you handle incidents and on-call responsibilities for a Kubernetes cluster?
8. What is your philosophy on automation in a Kubernetes environment? Provide an example where you significantly improved something through automation.
9. Describe a time you received critical feedback on a Kubernetes design or implementation. How did you respond?
10. What are some of the biggest challenges you foresee when managing Kubernetes at scale (e.g., 100+ nodes, 1000+ Pods)?

NAVNEET YADAV