Kubernetes Concepts Summary

## 1. Difference between Deployment and StatefulSet

\*\*Deployment\*\*:

- Manages stateless applications.

- Pods are interchangeable; no stable identity.

- Scaling and rolling updates are straightforward.

- No guarantees about pod ordering or persistence.

\*\*StatefulSet\*\*:

- Manages stateful applications.

- Each pod has a stable identity (e.g., web-0, web-1).

- Maintains persistent storage via PVCs tied to pod identity.

- Ensures ordered deployment, scaling, and termination.

## 2. When to use StatefulSet instead of Deployment

- You need stable network identities (e.g., databases like Cassandra, Kafka).

- Each pod requires a persistent volume that should not be reused.

- You need ordered startup/shutdown (e.g., leader election, quorum-based systems).

- Applications maintain internal state across restarts.

## 3. Can you attach a volume to a Deployment? How is it different from StatefulSet?

- Yes, you can attach volumes to a Deployment using PVCs or ephemeral volumes.

- In a Deployment, volumes are shared or dynamically provisioned, but not tied to pod identity.

- In a StatefulSet, each pod gets a unique PVC (e.g., data-web-0, data-web-1) that persists across rescheduling.

## 4. What could cause a StatefulSet pod to fail when rescheduled to a different availability zone?

- Zonal PV binding: The PVC is bound to a PV in a specific AZ and cannot be mounted in another.

- Storage class limitations: Some storage classes (like EBS in AWS) are AZ-specific.

- Node affinity: Pod may be scheduled to a node that cannot access the volume.

- Network or IAM issues: Cross-AZ access may be restricted.

## 5. How do PV/PVC behave across zones in EKS or Kubernetes in general?

- EBS volumes in AWS are AZ-bound. A pod rescheduled to another AZ cannot mount the same volume.

- Use multi-AZ aware storage classes (e.g., EFS, FSx).

- Use volume replication or backup/restore strategies.

- Consider zonal affinity to keep pods in the same AZ as their volumes.

## 6. What is a DaemonSet and when would you use it?

- A DaemonSet ensures one pod per node.

- Use cases include node-level monitoring, log collection, security agents, CSI drivers, or network plugins.

## 7. If you want two pods per node (instead of one), what alternatives to DaemonSet can you use?

- Use a Deployment with node affinity and pod anti-affinity to place multiple pods per node.

- Use Jobs or CronJobs with node selectors.

- Use custom controllers or init scripts if you need tight control.

## 8. What is a Pod Disruption Budget (PDB) and how is it useful?

- A PDB defines the minimum number of pods that must be available during voluntary disruptions.

- Prevents service downtime during maintenance.

- Example: minAvailable: 2 ensures at least 2 pods are running before allowing disruption.

## 9. How do you handle certificate rotation in on-prem Kubernetes clusters?

- Use cert-manager with internal CA or Vault.

- Automate rotation via cron jobs or custom scripts.

- Monitor expiry with tools like Kube-Bench or Prometheus alerts.

- Rotate Kubelet certificates, API server certs, Etcd certs, Ingress TLS certs.

## 10. Challenges with scheduling pods in a multi-node, multi-AZ setup

- Volume locality: PVs may not be accessible across AZs.

- Network latency: Cross-AZ communication can impact performance.

- Affinity/anti-affinity rules must be carefully designed.

- Resource fragmentation and failure domains must be considered.

## 11. How does the Kubernetes scheduler decide where to place pods?

- Scheduler considers resource availability, node selectors, taints/tolerations.

- Affinity/anti-affinity rules, pod topology spread constraints.

- Custom scheduling policies via plugins or extenders.

## 12. What happens when a StatefulSet pod cannot mount its volume after moving to another node?

- Pod enters CrashLoopBackOff or Pending state.

- PVC is bound to a PV in a different AZ or node.

- Resolution: Reschedule pod to original AZ, use multi-AZ storage (e.g., EFS), recreate PVC/PV if data loss is acceptable.

Advanced DevOps and Cloud Interview Questions & Answers

# Terraform Questions

## 1. Common Challenges with Terraform

* State file conflicts in team environments.
* Drift detection between actual infrastructure and Terraform state.
* Secret management (e.g., credentials in `.tf` files).
* Module versioning and reuse across environments.
* Complex dependency graphs causing slow or unpredictable applies.
* Provider limitations or breaking changes in updates.

## 2. State File Management

* Use remote backends like S3 with DynamoDB for state locking.
* Enable versioning on the backend (e.g., S3) for rollback.
* Avoid manual edits; use `terraform state` commands for manipulation.
* Use workspaces for environment separation (e.g., dev/stage/prod).

## 3. Drift Detection and Resolution

* Run `terraform plan` regularly to detect drift.
* Use tools like Terraform Cloud, Atlantis, or Driftctl.
* Investigate and reconcile differences manually or via automation.
* Enforce immutable infrastructure practices to reduce drift.

## 4. Managing Secrets Securely

* Use Terraform Cloud/Enterprise with Vault integration.
* Store secrets in AWS Secrets Manager, SSM Parameter Store, or Vault.
* Avoid hardcoding secrets in `.tf` files or variables.
* Use environment variables or encrypted files with CI/CD.

## 5. Why Use a Remote Backend

* Enables collaboration with state locking.
* Provides centralized state management.
* Supports versioning and recovery.
* Facilitates automation in CI/CD pipelines.

# AWS & Networking Questions

## 6. Ways to Deploy Nginx on AWS

* EC2 instance with Nginx installed.
* ECS (Fargate or EC2) with Nginx container.
* EKS with Nginx as a pod or ingress controller.
* Elastic Beanstalk with Nginx configuration.
* Lambda + API Gateway (for lightweight reverse proxy use cases).
* Lightsail with Nginx pre-configured.

## 7. Pre-requisites for VPC Peering

* Non-overlapping CIDR blocks.
* Both VPCs must be in the same region (or support inter-region peering).
* Proper route tables and security groups.
* IAM permissions for peering creation.

## 8. Problems with Overlapping CIDRs

* Routing conflicts and packet drops.
* Security group misconfigurations.
* No direct peering possible without NAT or tunneling.
* DNS resolution issues.

## 9. Communication Between Overlapping CIDRs

* Use NAT gateways or NAT instances.
* Implement VPN tunnels with custom routing.
* Use Transit Gateway with route domain isolation.
* Deploy proxy/jump servers to relay traffic.

## 10. What is a Transit Gateway?

* A centralized hub for connecting multiple VPCs and on-prem networks.
* Supports transitive routing, unlike VPC peering.
* Simplifies network architecture and scaling.
* Enables inter-region communication.

## 11. Jump Server in Overlapping Networks

* Acts as a bastion host to access resources in overlapping CIDRs.
* Can be configured with SSH tunneling or proxying.
* Helps segregate access and audit traffic.

## 12. Transitive Routing Between VPCs A, B, and C

* Not supported via VPC peering directly.
* Use Transit Gateway or third-party appliances.
* Example: A ↔ TG ↔ B ↔ TG ↔ C, with proper route tables.

# CI/CD & GitHub Actions

## 13. CI/CD Tools Used

* GitHub Actions, Jenkins, GitLab CI, ArgoCD, Spinnaker.
* Integrated with SonarQube, Trivy, Docker, Helm, Terraform.

## 14. Integrating SonarQube, Docker, Trivy

* SonarQube: Static code analysis in build stage.
* Docker: Build and push images in CI.
* Trivy: Scan Docker images for vulnerabilities before deployment.
* Use GitHub Actions or Jenkins pipelines with stages for each tool.

## 15. Trigger GitHub Actions in Another Repo

* Use `repository\_dispatch` event.
* Requires a personal access token with repo access.
* Example:  
   curl -X POST \  
   -H "Authorization: token $TOKEN" \  
   -H "Accept: application/vnd.github.v3+json" \  
   https://api.github.com/repos/owner/repo/dispatches \  
   -d '{"event\_type":"trigger-ci"}'

## 16. Purpose of `repository\_dispatch`

* Allows external triggers for workflows.
* Useful for cross-repo orchestration.
* Supports custom payloads for dynamic workflows.

## 17. Trigger Pipeline in Repo A from Repo B

* Use `repository\_dispatch` from Repo B to Repo A.
* Configure Repo A to listen for the event and run the workflow.
* Secure with tokens and permissions.