

# Accenture Interview Round 1 & 2 — DevOps Technical Deep Dive

LinkedIn: [Amit Singh](#)

Medium: [Amit Singh – Medium](#)

## Q. How are your day-to-day activities as a DevOps Engineer?

### Pointers

- Monitoring infrastructure
- Writing/maintaining IaC
- Improving pipelines
- Handling releases
- Supporting dev teams
- Troubleshooting incidents

```
[ Daily ]
|
+--> Monitor clusters
+--> CI/CD maintenance
+--> Infra as Code
+--> Incident support
```

### Scenario style

“On a typical day, I monitor production clusters, manage and improve our Jenkins pipelines, work on Terraform modules for resource provisioning, and collaborate with developers for application releases. I also handle incidents like pod failures or networking issues.”

🧠 *Easy to remember:* **Monitor, Improve, Support, Troubleshoot**

## Q. What is a NAT Gateway?

### Pointers

- Allows private subnets to access the internet
- Without exposing resources to incoming traffic
- Managed by AWS



## ✅ Scenarios

“We had EC2 instances in private subnets that needed to pull Docker images from public repos. We placed a NAT Gateway in a public subnet, routing private subnet traffic through it so those instances could reach the internet safely.”

“In my last project, the private subnets for EC2 instances needed to pull OS updates from the internet. I used a NAT gateway in the public subnet, routing traffic through it, while blocking inbound access to those instances.”

## Diagram

Internet-> IGW → NAT Gateway → Private Subnet EC2

## Advanced Q&A

**Q:** Can a NAT Gateway receive inbound traffic?

**A:** No — it only handles outbound requests from private subnets.

## Best practices

- Always place a NAT gateway in a public subnet
- Remember, *private* subnet routes point to the NAT

# Q. Prerequisites to upgrade a Kubernetes cluster

## Pointers

- Backup etcd
- Drain nodes
- Check deprecated APIs
- Update kubectl
- Test in staging

## ✅ Scenario style

“We planned a Kubernetes upgrade from v1.24 to v1.27. First, we backed up etcd, verified compatibility of Ingress controllers, and checked deprecated APIs. Then we drained and upgraded worker nodes in a rolling manner.”

```
[ etcd snapshot ] --> [ kubeadm upgrade plan ]
|
[ drain nodes ]
|
[ rolling node upgrade ]
```

🧠 *Easy to remember: Backup, Drain, Check, Test*

## Q. What is Pod Disruption Budget (PDB)?

### Pointers

- Defines minimum available pods during voluntary disruptions
- Prevents too many pods from going down during drain/updates

### ✅ Scenario style

1. “When upgrading a deployment, I set PDB to minAvailable: 2 to ensure at least 2 pods are always running so user traffic isn’t impacted.”
2. “We needed to upgrade the node pool for a production app. We set a PDB to minAvailable: 2 to keep at least 2 pods online even during the drain.”

### PDB

+++> minAvailable: 2

+++> maxUnavailable: 1

### Advanced Q&A

**Q:** Does PDB protect from node failures?

**A:** No, PDB only controls voluntary disruptions (e.g., drain, upgrades).

### Best practices

- always set PDB for critical deployments
- test the effect by cordoning a node

🧠 *Easy to remember: Minimum running pods during changes*

## Q. Shell script for factorial of a number

```
#!/bin/bash
echo "Enter a number:"
read num
fact=1
for (( i=1; i<=num; i++ ))
do
    fact=$((fact * i))
done
echo "Factorial of $num is $fact"
```



## Q. Tell me about VPC structure in your project

### Pointers

- VPC CIDR
- Public & private subnets
- NAT gateway
- Internet gateway
- Route tables
- Security groups

### ✓ Scenario style

“We designed a VPC with /16 CIDR, split into multiple private subnets for application servers, with a NAT gateway for outbound internet, public subnets for the ALB, and separate security groups for database layers.”

“Our VPC has a /16 CIDR block, split into public subnets for the ALB and private subnets for EC2/EKS nodes. NAT gateways handle outbound traffic for private subnets. RDS is placed in private subnet.”

```
VPC
|
+--> Public Subnet (ALB, NAT GW)
|
+--> Private Subnet (EC2, EKS, RDS)
```

### Advanced Q&A

**Q:** Why put RDS in a private subnet?

**A:** For security — no direct internet access.

### Best practices

- separate public/private

- follow least privilege on security groups

🧠 *Easy to remember: CIDR, Subnets, NAT, IGW, Routing*

## Q. How is your CI/CD pipeline set up? What security tools are integrated?

### Pointers

- Jenkins/GitLab
- Docker builds
- SonarQube (code scan)
- Trivy/Anchore (image scan)
- HashiCorp Vault (secrets)

**Code → Build → Scan (Trivy) → Test → Deploy to K8s**

### ✅ Scenario style

“Our pipeline is on GitLab CI, running Docker builds, security scanning with Trivy, static code analysis with SonarQube, and uses Vault to inject secrets. This ensures secure, consistent, automated releases.”

### Advanced Q&A

**Q:** How do you manage secrets?

**A:** Vault or SSM Parameter Store, never hard-coded.

### Best practices

- integrate image scanning
- automate secrets rotation

🧠 *Easy to remember: Build, Scan, Store secrets, Deploy*

## Q. How do you manage them?

### Pointers

- Version control (Git)
- IaC for infra
- Role-based access

- Automated tests

```
GitOps
|
Infra as Code
|
RBAC
```

#### ✓ Scenario style

“We manage pipelines through version-controlled YAML, infrastructure with Terraform, and RBAC controls in Kubernetes to delegate least privilege.”

🧠 *Easy to remember:* **Version, Automate, Secure**

## Q. Write a rough pipeline script for microservices architecture

```
[ microservice 1 ] --> pipeline
[ microservice 2 ] --> pipeline
```

```
stages:
  - build
  - test
  - deploy

build:
  script:
    - docker build -t myapp:${CI_COMMIT_SHA} .
    - docker push myapp:${CI_COMMIT_SHA}

test:
  script:
    - docker run myapp:${CI_COMMIT_SHA} pytest

deploy:
  script:
    - kubectl apply -f deployment.yaml
```



## Advanced Q&A

**Q:** How do you handle dependencies?

**A:** Use semantic versioning + separate pipelines to avoid coupling.

## Best practices

- keep microservices decoupled tag images uniquely

🧠 *Remember:* **Build → Test → Deploy**

## Q. What is multi-stage Docker build?

### Pointers

- Separate build & run stages
- Reduce image size
- Improves security

### ✅ Scenario style

“In a microservice build, we use a Golang builder image, compile binaries, and then copy them to a scratch image in a second stage. That keeps the production image minimal.”

## Advanced Q&A

**Q:** Why use multi-stage?

**A:** Reduce attack surface and image size.

## Best practices

- keep only production dependencies
- scan final images

🧠 *Easy to remember:* **Build once, copy artifacts, keep clean**

## Q. What are manifest files?

### ✅ Pointers

- YAML files for Kubernetes resources
- Define pods, deployments, services, etc.

### ✅ Scenario style

“We manage Kubernetes manifests for deployments and services in a GitOps workflow to apply them consistently across environments.”

“We store Deployment and Service YAMLs in Git repos. We apply them with kubectl or FluxCD.”

## Advanced Q&A

**Q:** How to manage multiple environments?

**A:** Use Kustomize or Helm.

## Best practices

- version manifests
- keep separate folders for dev/prod

🧠 *Easy to remember:* **K8s blueprints**

## Q. What is Ansible Vault?

### Pointers

- Encrypt sensitive data (passwords, secrets)
- Stored in playbooks securely



[ ansible-playbook ] --> [ encrypted vault file ]

### ✓ Scenario style

“We use Ansible Vault to encrypt DB passwords in our inventory, and decrypt only during runtime with a vault password file.”

### Advanced Q&A

**Q:** What if you lose the vault password?

**A:** You cannot decrypt — store vault password securely.

### Best practices

- rotate vault passwords
- never commit vault keys to Git

🧠 *Easy to remember:* **Encrypted secrets for playbooks**

## Q. How to make a K8s cluster highly available?

### Pointers

- Multiple control plane nodes
- Etcd cluster quorum
- Load balancer in front of control plane
- Spread worker nodes across AZs

### ✓ Scenario style

“We deployed 3 control plane nodes with an external HA load balancer and spread worker nodes in 3 AZs to achieve high availability.”

```
LB
|
+--> master1
+--> master2
+--> master3
```

### Advanced Q&A

**Q:** How do you handle etcd failure?

**A:** Ensure odd number of etcd members and frequent snapshots.

### Best practices

- HA LB in front of control planes
- spread AZs

🧠 *Easy to remember:* **Multi-master + Load Balancer + Spread**

## Q. Monitoring tools & common pod errors

### Pointers

- Prometheus, Grafana
- Alertmanager
- Common errors: CrashLoopBackOff, ImagePullBackOff

### ✅ Scenario style

“We use Prometheus + Grafana for metrics and Alertmanager for notifications. The most common pod issue I handled was CrashLoopBackOff due to wrong configmaps or missing secrets.”

### Advanced Q&A

**Q:** What's CrashLoopBackOff?

**A:** Container keeps crashing repeatedly, often due to bad configs.



### Best practices

- set up alerts
- test alert receivers regularly

🧠 *Easy to remember:* **Prometheus + Grafana + Alerts**

## Q. Terraform script for VPC architecture

```
provider "aws" {  
  region = "ap-south-1"  
}  
  
resource "aws_vpc" "my_vpc" {  
  cidr_block = "10.0.0.0/16"  
}  
  
resource "aws_subnet" "public_subnet" {  
  vpc_id      = aws_vpc.my_vpc.id  
  cidr_block   = "10.0.1.0/24"  
  availability_zone = "ap-south-1a"  
  map_public_ip_on_launch = true  
}  
  
resource "aws_internet_gateway" "igw" {  
  vpc_id = aws_vpc.my_vpc.id  
}
```

 Copy  Edit



## Advanced Q&A

**Q:** How to handle state files?

**A:** Use remote backend with S3 + DynamoDB locking.

## Best practices

- use terraform fmt
- version lock your providers

 **Remember: VPC → Subnets → IGW**

## Q. How many objects can an S3 bucket store?

### ✓ Answer

- Unlimited objects
- Practically limited by storage quotas

“S3 scales virtually unlimited. One bucket can store billions of objects.”

## Advanced Q&A

**Q:** Any hard limits?

**A:** Only practical ones (like request rates), no hard object limit.

## Best practices

- enable bucket versioning

- use lifecycle rules

🧠 *Easy to remember:* **Unlimited**

## Q. What are IAM roles and policies?

### Pointers

- Roles: identities with permissions
- Policies: permission documents in JSON
- Used to grant granular access

### ✅ Scenario style

“I assigned an IAM role to an EC2 instance with a policy to only allow S3 access for backup storage.”

### Advanced Q&A

**Q:** Difference between policy and role?

**A:** Role = identity; Policy = permission rules.

### Best practices

- follow least privilege
- audit IAM regularly

🧠 *Easy to remember:* **Role = identity, Policy = rules**

## Q. What are artifacts?

### Pointers

- Build outputs
- Stored in artifact repositories like Artifactory or S3

```
Build
|
Artifact
|
Artifactory
```

### ✅ Scenario style

“Our CI pipeline pushes JAR artifacts to Artifactory after a successful Maven build.”

“Our Maven builds produce JARs stored in Artifactory as versioned artifacts.”

### Advanced Q&A

**Q:** Why store them?

**A:** To enable rollback or re-deploys.

### Best practices

- automate artifact cleanup
- version artifacts clearly

🧠 *Easy to remember:* **Build results**

## Q. SATS and DATS?

### Common terms

- SATS = System Acceptance Testing
- DATS = Data Acceptance Testing

SATS → DATS

### ✅ Scenario style

“After deploying, we perform SATS to validate application behavior, and DATS to verify data correctness with staging data.”


### Advanced Q&A

**Q:** Are they manual or automated?

**A:** Usually manual with automated test cases integrated.

### Best practices

- always document acceptance criteria
- automate where possible

 *Easy to remember: System + Data Testing*

## Q. How do you find errors in pipelines?

### Pointers

- Logs
- CI/CD dashboard
- Alerts
- Test reports

### ✅ Scenario style

“Whenever the pipeline fails, I check Jenkins logs, review the failing stage, and correlate with Git commit changes.”

### Advanced Q&A

**Q:** How do you get notified?

**A:** Slack or email from pipeline notifications.

### Best practices

- break pipeline into small stages
- add retry steps

 *Easy to remember: Logs + Stage + Commit*

## Q. What are Ansible Roles?

### Pointers

- Reusable set of tasks
- Structured like modules
- Makes playbooks cleaner

### ✅ Scenario style

“We created roles for installing Nginx, managing users, and deploying apps, to keep our playbooks DRY and modular.”

### Advanced Q&A

**Q:** Benefits of roles?

**A:** Reusable, maintainable, clean structure.

### Best practices

- structure roles with defaults, tasks, handlers
- version control them

🧠 *Easy to remember:* **Reusable Ansible blocks**

**Thanks Everyone!**

Connect with me: [Amit Singh](#)