

QUESTION 1: “TELL ME ABOUT YOURSELF”

2–3 minute

“I’m <yourname>. I have a little over three years of experience working as a DevOps Engineer, mostly on AWS-based microservices platforms.

In my last role at <company>, I worked closely with development teams to automate how applications were built, deployed, and operated in production. My work wasn’t just setting things up once — it was ongoing ownership of CI/CD pipelines, infrastructure, and Kubernetes environments.

On the CI/CD side, I worked mainly with Jenkins and Argo CD. Jenkins handled the build and validation part — things like building the application, creating Docker images, scanning them, and pushing them to ECR. Argo CD handled deployments using GitOps, so production changes were controlled and traceable.

I also spent a lot of time on Kubernetes, especially AWS EKS. That included deploying applications, handling autoscaling, upgrading node groups, and supporting production issues. I regularly monitored clusters using Prometheus and Grafana and responded to alerts when something went wrong.

Overall, my experience is very hands-on and production-focused. I’ve seen how systems behave under load, how deployments can fail, and what it takes to keep platforms stable over time.”

QUESTION 2: “WHAT WERE YOUR ROLES AND RESPONSIBILITIES?”

2–3 minute

“In my last role, my responsibilities were spread across infrastructure, CI/CD, and Kubernetes operations.

On the infrastructure side, I worked with Terraform to provision and manage AWS resources like EKS, VPCs, IAM roles, EC2, and supporting components. Infrastructure changes were always done through Git, so we had version control and proper review before applying anything.

For CI/CD, I was responsible for maintaining Jenkins pipelines. That included troubleshooting build failures, fixing Docker-related issues, integrating security scans, and making sure builds were reliable. I also worked on Argo CD to manage deployments so that production changes were predictable and easy to roll back.

On the Kubernetes side, I handled deployments, scaling, upgrades, and day-to-day operational issues. This included managing replica counts, tuning resource requests and limits, handling node upgrades, and ensuring workloads were healthy after changes.

Beyond tools, I also worked closely with developers — helping them debug deployment failures, explaining why something broke in Kubernetes, and improving how applications were packaged and deployed.”

QUESTION 3: “WHAT DID YOUR DAY-TO-DAY ACTIVITIES LOOK LIKE?”

2–3 minute

“On a typical day, I usually started by checking the health of our systems. That meant looking at Jenkins pipelines, Argo CD sync status, and Grafana dashboards to see if anything unusual happened overnight.

If there were pipeline failures, I’d investigate whether it was a code issue, a dependency problem, or something related to Docker or the build environment. Many times, it was small things like version mismatches or missing environment variables.

On the Kubernetes side, I regularly checked pod health, node utilization, and autoscaling behavior. If alerts came in — like high memory usage or pod restarts — I’d dig into logs and metrics to understand the root cause.

I also spent time supporting deployments. That could mean helping developers with rollout issues, validating that new releases were stable, or rolling back changes when something didn’t behave as expected.

Overall, my day-to-day work was a mix of monitoring, troubleshooting, improving automation, and making sure production stayed stable.”

QUESTION 4: “EXPLAIN YOUR PROJECT END-TO-END”

2–3 minute

“The project I worked on was a cloud-native DevOps platform built to support multiple microservices running on AWS EKS. The main goal was to automate everything — from infrastructure provisioning to application deployment — while keeping production stable and observable.

Developers pushed their code to GitHub. Jenkins handled the CI part — building the application, creating Docker images, scanning them, and pushing them to Amazon ECR. Deployment was handled separately using Argo CD, following a GitOps approach. Any change to Kubernetes manifests in Git would automatically sync to the EKS cluster.

Kubernetes handled scaling and deployment strategies like rolling updates, which helped us avoid downtime. For monitoring, we used Prometheus and Grafana to track application and cluster metrics.

My role in this project was hands-on. I worked on Terraform modules, Jenkins pipelines, Argo CD setup, Kubernetes deployments, and production support. I also handled issues like failed deployments, scaling problems, and resource constraints.

It wasn’t just about building the platform — it was about running it reliably in production.”

QUESTION 5: “WHAT HAPPENS FROM CODE COMMIT TO PRODUCTION?”

2–3 minute

“When a developer merged code into the main branch, a Jenkins pipeline was triggered automatically using webhooks.

The pipeline pulled the code, ran the build process, and created a Docker image. Before pushing the image, we ran basic validations and vulnerability scans. If anything failed, the pipeline stopped there, so broken or insecure images never reached production.

Once the image was pushed to ECR, deployment was handled through GitOps. We updated the image tag in Kubernetes manifests stored in Git. Argo CD continuously monitored that repository and synced the changes to the EKS cluster.

Kubernetes then handled the rollout using rolling updates, replacing old pods gradually. After deployment, I monitored metrics and logs to make sure the application behaved normally. If anything went wrong, rollback was straightforward through Argo CD.”

QUESTION 6: “DESCRIBE A PRODUCTION ISSUE YOU HANDLED”

2–3 minute

“One production issue I clearly remember was when a service started crashing after a new release. The pods were going into CrashLoopBackOff.

I checked the pod logs and descriptions and noticed the containers were being OOM-killed. That told me it was a memory issue rather than an application crash.

I looked at Prometheus metrics and confirmed memory usage had increased significantly after the new release. To stabilize the service quickly, I updated the resource requests and limits through our GitOps workflow and redeployed the application using Argo CD.

Once the service stabilized, I worked on prevention. I added memory usage alerts in Grafana so we’d catch similar issues earlier in the future. This helped avoid repeated incidents.”

QUESTION 7: “WHY SHOULD WE HIRE YOU?”

2–3 minute

“You should hire me because I’ve already worked in environments where stability, reliability, and automation really matter.

I’ve handled the full DevOps lifecycle — building CI/CD pipelines, automating infrastructure, deploying applications on Kubernetes, and supporting production systems when things break. I understand that production is unpredictable, and I’m comfortable handling failures calmly and fixing them without panic.

I focus a lot on reducing manual work through automation and making systems observable so issues are caught early. I also care about safe deployments — having rollbacks, monitoring, and controlled changes in place.

Overall, I don’t just push changes and move on. I take ownership of the platform and make sure it remains stable, secure, and easy to operate over time.”