DevOps Engineer interview questions

careertips.com.au: common questions and what they're really testing

What to expect

DevOps interviews test whether you can actually build and run the systems on your resume, not just talk about them. Expect a mix of hands-on technical questions, scenario-based troubleshooting, and behavioural questions about working across development and operations teams.

- **Process:** Questions asking you to walk through how you'd design or build a pipeline, deployment process, or piece of infrastructure from scratch.
- **Technical troubleshooting:** Diagnostic questions about how you'd investigate a specific failure, using logs, metrics and system knowledge.
- **Scenario / judgement:** Time-pressured or ambiguous situations, such as a production incident, where the interviewer wants to see your decision-making under pressure.
- **Behavioural:** Past-experience questions about cross-team collaboration, influencing developers or handling disagreement over tooling or process.
- **Security awareness:** Questions checking whether security is built into your pipeline and infrastructure work rather than treated as a separate concern.

Most DevOps interviews start with a recruiter or technical screen covering your tooling background, followed by a technical round with a senior engineer that includes live troubleshooting or a whiteboard system design. A behavioural round with the hiring manager or wider team usually follows, sometimes combined with a take-home or pairing exercise on a real pipeline problem.

1. Walk me through how you'd design a CI/CD pipeline for a new microservice, from code commit to production deployment.

Why they ask

This checks whether you understand the full pipeline lifecycle, not just individual tools, and whether you build in testing and rollback safeguards.

How to structure your answer

Walk through it in order: source control trigger, build and test stages, artifact storage, deployment stages (staging then production), and rollback or approval gates. Name the tools you'd use at each stage and explain why.

Example answer

“I'd start with a GitLab pipeline triggered on merge to main. The build stage compiles and runs unit tests, then a Docker image gets built and pushed to a registry. From there it deploys automatically to a staging environment on Kubernetes, where integration tests run against real dependencies. Production deployment would need a manual approval gate, using a rolling update strategy in Kubernetes so we can roll back quickly if Prometheus alerts show error rates climbing after release.”

2. A deployment has just failed in production. How would you go about troubleshooting it?

Why they ask

This is a core, recurring part of the job, so interviewers want a clear diagnostic process rather than guesswork.

How to structure your answer

Give a diagnostic sequence: confirm the scope of impact, check recent changes, work through logs and metrics systematically, isolate the cause, then explain how you'd fix and prevent recurrence.

Example answer

“First I'd check whether the failure is affecting all users or a subset, and look at what changed in the last deployment, since that's the most common cause. I'd pull logs from the ELK Stack and check Prometheus dashboards for resource spikes or error rate changes around the deployment time. If it looks like a config issue, I'd check the Terraform state and recent commits for anything that changed environment variables or resource limits. Once I've isolated the cause, I'd roll back if the fix isn't immediate, then write up what happened so we can add a check for it in the pipeline.”

3. Tell me about a time you had to convince a development team to change how they worked, for example adopting a new tool or process.

Why they ask

DevOps engineers spend a lot of time influencing teams they don't manage, so this tests communication and persuasion, not just technical skill.

How to structure your answer

Use STAR: describe the situation and task, the specific action you took to bring the team along, and the measurable result.

Example answer

“A development team I worked with was deploying manually because they didn't trust the automated pipeline after an earlier bad experience. I sat down with them, walked through exactly what the pipeline checked at each stage, and ran a few of their own recent deployments through it in a test environment so they could see it catch issues before production. After a couple of weeks running both processes in parallel, they moved over fully. Deployment frequency went up and the manual errors that used to happen on Friday afternoons stopped.”

4. It's outside business hours and a Kubernetes pod keeps crashing under load. What do you do?

Why they ask

This tests judgement under pressure and whether you have a calm, structured approach to incidents rather than panicking or guessing.

How to structure your answer

Describe your immediate priority (stabilise the system), then your investigation approach, then how you'd communicate and follow up afterwards.

Example answer

“My first move is to stop the bleeding, so I'd check if scaling up replicas or increasing resource limits buys immediate stability while I investigate. I'd pull pod logs and check Prometheus for memory or CPU patterns leading up to the crash. If it's a memory leak under load, I'd look at recent code changes and consider a temporary rollback to the last stable version. I'd keep the team informed as I go, and once it's stable, follow up the next day with a proper root cause writeup so we fix it rather than just patching around it.”

5. How do you use Terraform to manage infrastructure consistently across development, staging and production environments?

Why they ask

This checks practical, hands-on knowledge of infrastructure as code rather than surface familiarity with the term.

How to structure your answer

Explain your technical approach directly: how you structure code, manage state, and handle differences between environments.

Example answer

“I structure Terraform code into reusable modules with environment-specific variable files, so the underlying infrastructure definition stays the same and only the variables change between environments. I keep state files in remote storage with locking to avoid conflicts when more than one person applies changes. Any change goes through a plan step reviewed in a pull request before it's applied, so we catch unintended changes to production before they happen.”

6. How do you build security into a CI/CD pipeline rather than treating it as a separate step?

Why they ask

With cyber security listed as a core skill for this role, interviewers want to know you think about security proactively, not as an afterthought.

How to structure your answer

Lay out the specific checks and practices you'd add at each pipeline stage, and how you'd balance security with release speed.

Example answer

“I'd add dependency and container image scanning as an automatic pipeline stage, so vulnerable packages get flagged before a build progresses. Secrets management would go through a dedicated vault rather than environment variables in code, and I'd restrict who can approve production deployments. I'd also make sure infrastructure changes through Terraform go through the same review process as application code, since misconfigured infrastructure is just as likely to cause a security incident as bad application code.”