

DevOps engineer (fresher)

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Full-time | CTC: [CTC for the full-time role]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

[Company name] needs a fresher DevOps engineer in the [Team] team to help keep our builds, releases and servers running. You will start by reading pipeline logs, fixing small pipeline and container problems and watching our dashboards, with a senior approving every change that touches production. Over time you will take on more of the release routine and learn one cloud platform. The role suits a graduate who likes to find out why a system stopped and to write the fix down.

What you will do

- Check the CI pipeline after each push in GitHub Actions or Jenkins, read the log of any failed step and either fix it or tell the developer what broke
- Make small changes to pipeline files, such as a new test step, a cache or an environment variable for a non-secret setting, and raise them as pull requests for a senior to approve; keep tokens and passwords in the CI tool's secret store, never in the file or the repository
- Build Docker images from a Dockerfile, run the containers locally and in the test environment, and fix common image, port and variable problems
- Follow the release checklist: deploy to staging, check that the service is healthy afterwards and record what you saw, with a senior running or approving every production release
- Check Linux servers for full disks, low memory and stopped services using commands such as df, du, top and systemctl, and clear only what the runbook says is safe
- Write short shell scripts for repeat jobs, such as log clean-up, backup checks and environment set-up, and keep them in Git with a note on how to run them
- Watch the monitoring dashboards and alerts for disk, memory, failed health checks and error responses, read the logs behind an alert and tell the on-call engineer what you found and when it began
- Handle access and environment requests, such as a new developer account or a test environment, through the approved ticket process and record who approved each one
- Sit in on incidents, run the checks a senior asks for, write the timeline of events afterwards and turn the fix into a runbook entry for next time
- Learn one cloud (AWS, Azure or GCP) by building practice resources in a sandbox account, such as a virtual machine, a storage bucket and network rules, and write up what you built

What we are looking for

- Linux command line: moving around the file system, file permissions, reading logs with tail and grep, and starting and stopping services
- Git: branches, pull requests and merge conflicts, and a clear idea of how a commit becomes a running service
- CI/CD basics: build, test and deploy stages, with one pipeline you have written or edited in a tool such as GitHub Actions, Jenkins or GitLab CI/CD, shown in a repository or described if the work is private

- Docker: writing a simple Dockerfile, building an image and running a container with ports and environment variables
- Networking basics: IP address, port, DNS, and what HTTP and HTTPS do
- Shell or Python scripting for a repeat task, such as renaming files or checking a service
- A project you can show or walk through: an application you containerised and deployed through a pipeline to a server, a virtual machine on your own computer or a cloud practice account, with the files in a repository or the steps described if the work is private
- The habit of asking before changing a shared server or a production setting

Good to have

- A practice deployment on AWS, Azure, GCP or a virtual machine on your own computer, such as a web server with storage and firewall rules
- Basic Nginx: serving a site and acting as a reverse proxy
- Dashboards in Grafana or in the cloud provider's own monitoring tool
- Awareness of what Kubernetes or Terraform is for, from a course or a practice project
- Python scripting for small automation jobs, such as calling an API or a cloud SDK

Who can apply

Final-year students and graduates of B.E./B.Tech in computer science, IT, electronics or related streams fit well, as do BCA, B.Sc (computer science or IT) and MCA candidates. Candidates from other streams are welcome. Linux practice and one project run on a server or virtual machine matter more than the degree stream.

How to apply

Apply through this job post by [\[Apply by\]](#). Shortlisted candidates will hear from us about the next steps.

DevOps engineer intern

Company: [Company name]

Location: [Location] | Work mode: [Work mode]

Employment type: Internship | Duration: [Internship duration] | Stipend: [Stipend for the internship]

Joining: [Joining date] | Apply by: [Apply by]

About [Company name]

[About your company]

About the role

[Company name] offers a DevOps internship for students who want to see how software is built, packaged and released. You will work with a mentor on a sample pipeline, a container and a test environment, and learn from real failures. Interns work in test environments only; production stays with the senior engineers. You will finish with a runbook and a short demo.

What you will do

- Containerise a small sample application with a Dockerfile and run it on your laptop and in the test environment
- Build a simple pipeline in GitHub Actions or Jenkins that builds, tests and reports on a sample repository
- Break a pipeline on purpose in a practice repository, read the log, and note what each failure looked like and how you fixed it
- Write a shell script that checks disk space, memory and a service's status and writes the result to a log file
- Set up a dashboard and one alert for a test service and note what a normal day looks like
- Write a runbook for one repeat problem, then follow it yourself on a fresh test server to check that every step works

What we are looking for

- A student or recent graduate of computer science, IT, electronics or a related course
- Has used the Linux command line, on a laptop install, a virtual machine, Windows Subsystem for Linux or a college lab machine, and is comfortable in the terminal
- Knows Git basics and can show a project in a repository, or walk through it if the work is private
- Has run at least one application in Docker, even from a tutorial, and can say what each line of the Dockerfile does
- Ready to ask before changing anything on a shared server or environment

Who can apply

Final-year students and graduates of B.E./B.Tech in computer science, IT, electronics or related streams fit well, as do BCA, B.Sc (computer science or IT) and MCA candidates. Candidates from other streams are welcome. Linux practice and one project run on a server or virtual machine matter more than the degree stream.

How to apply

Apply through this job post by [Apply by]. Shortlisted candidates will hear from us about the next steps.

Screening questions to add to your job post

1. **A pipeline that passed yesterday fails today and nobody changed the code. What could have changed?** Look for a new version of a dependency or runner image, an expired token or secret, a changed environment variable, a flaky test or an outside service that is down.
2. **A server's disk is full and the application has stopped. What do you check?** Look for `df` to find the full filesystem, `du` to find large folders, and old logs or unused Docker images as common causes, plus asking before deleting anything.
3. **What is the difference between a container and a virtual machine?** Look for a container sharing the host's operating system kernel and starting quickly, while a virtual machine carries its own operating system. One clear sentence is enough.
4. **A web page will not load for a user. Which checks would you make, from the browser to the server?** Look for a check of the address, DNS, the server's IP address and port, the HTTP status code and the server logs. Accept any sensible order. Nginx knowledge is good to have, so do not score it here.
5. **On Linux, what does the permission string `-rwxr-x---` mean?** Look for a regular file whose owner can read, write and execute it, whose group can read and execute it, and which gives others no permissions. A clear explanation matters more than remembering permission numbers.
6. **A teammate asks you for access to the production database. What do you do?** Look for checking the request ticket and the database owner's approval, then referring the request to a senior to grant only the access needed. The fresher must follow the approved access process and must not grant access on their own authority.
7. **Tell us about something in your own project that broke after you deployed it. How did you find the cause?** Look for logs and status checks and a specific fix, such as a wrong port, a missing environment variable or a full disk.