

PROGRAM SYLLABUS

DevOps Engineer Program

Join as the DevOps engineer and take **i27Helpdesk**, a real product with 6 apps, from source code to production.

LIVE, INSTRUCTOR-LED

END-TO-END DEVSECOPS

CKA / CKAD PATH

13

Modules, then the real project

6

Apps in one real product

4

Environments: dev, test, stage, prod

Live

Classes, with daily recordings

WHAT YOU WORK WITH

Foundations

Linux, Git, GitHub

Build

Maven, npm, Python

Quality & security

SonarQube, Trivy

CI/CD

GitHub Actions

Containers

Docker, Kubernetes

Observability

Prometheus, Grafana

PROGRAM OVERVIEW

You join the team. Your job: take i27Helpdesk live.

This program works like joining a company as its DevOps engineer. You learn each tool because the product needs it, then use them all together: **CI/CD for every app, quality and security gates, container images, Kubernetes deployments, and monitoring**, releasing through every environment up to production.

THE PRODUCT
i27Helpdesk
A real product that i27Academy uses, built to be replicated in training. You take it from source code to production.

6
apps

3
languages

4
environments

TECH STACK

Next.js
UI

Node.js
Services

Java
Services

Python
Services

MySQL
Database

RELEASE PATH

dev → test → stage → ✓ approval → prod

Every environment runs on managed Kubernetes, released through GitHub Actions.



Who this program is for

- Freshers starting a career in DevOps
- Professionals from support, admin, testing, or networking
- Developers moving into DevOps and platform roles
- Engineers who want real, end-to-end project experience



Before you start

- A basic understanding of how software is built
- Willingness to learn hands-on, every class
- No prior DevOps experience required
- We start from Linux and Git and build up step by step

Your learning journey

13 modules · the real project · career readiness

STAGE 01

Foundations

- 01 DevOps & cloud foundations
- 02 Linux
- 03 Git & GitHub

STAGE 02

Build & DevSecOps

- 04 Build tools
- 05 SonarQube
- 06 Trivy
- 07 Docker
- 08 Artifact Registry
- 09 GitHub Actions

STAGE 03

Kubernetes & observability

- 10 Kubernetes core
- 11 Kubernetes advanced
- 12 Operations
- 13 Prometheus & Grafana

STAGE 04

Real project & career

- P i27Helpdesk, end to end
- P Break & fix labs
- C Career readiness

Built live, yours to keep: source repos for all 6 apps, GitHub Actions workflows, Dockerfiles, Kubernetes manifests, and architecture diagrams are created in class and shared with you.

STAGE 01

Foundations

The ground every DevOps engineer stands on: how products are delivered, the servers they run on, and the code they start from.

01 DevOps and cloud foundations



Why it matters · Before any tool, understand how a product is built, delivered, and run.

- What DevOps is and the problems it solves
 - Where DevOps fits in the software lifecycle
 - Cloud computing and the major clouds
 - Account setup on AWS and Google Cloud
 - How a project starts, and the teams involved
 - Continuous integration, delivery, and deployment
 - What infrastructure means for an application
 - Environments: dev, test, stage, and prod
- o *and more*

Applied to i27Helpdesk · Meet the product: its architecture, its environments, and the release plan you will deliver.

02 Linux



Why it matters · Every server, container, and pipeline runner is Linux.

- The Linux file system and navigation
 - Comparing files and link files
 - Redirection and piping
 - Wildcards and regular expressions
 - locate and find
 - Archives and compression (tar, gzip, bzip2)
 - SSH keys and remote access
 - Working with files and directories
 - wc, sort, and uniq
 - Running multiple commands in one line
 - Command aliasing
 - grep, cut, paste, and tr
 - File permissions and ownership
 - Working with editors
- o *and more*

Applied to i27Helpdesk · Work on the servers and runners the product's apps build and run on, and read their logs.

03 Git and GitHub



Why it matters · All 6 apps live in Git, and every pipeline starts with a commit.

- Version control, and Git vs GitHub
 - Initialising and connecting a project
 - Rebase, reset, and undoing commits
 - Going back in history and checkouts
 - Pull requests and code reviews
 - Local and remote repositories
 - Branching and merging
 - Resolving merge conflicts
 - Tags and releases in the real world
 - GitHub organisation and administration
- o *and more*

Applied to i27Helpdesk · Every app lives in one GitHub organisation, and every change starts with a branch and a pull request.

STAGE 02

Build & DevSecOps

Turn source code into checked, secured, versioned releases, automatically, for every app.

04 Build tools: Maven, npm, and Python



Why it matters · i27Helpdesk has Java, Node.js, and Python apps, and each one builds differently.

- What a build tool does, and why DevOps needs it
 - Maven repositories and project structure
 - Maven dependency management
 - npm: package.json, installs, and lockfiles
 - Python: requirements and virtual environments
 - Installing Maven and configuring Java on Linux
 - Maven lifecycles and goals
 - Using private or company dependencies
 - npm ci vs npm install, and build scripts
 - Installing dependencies with pip
- o *and more*

Applied to i27Helpdesk · Build the Java services with Maven, the Next.js and Node.js apps with npm, and the Python services with pip.

05 Code quality with SonarQube



Why it matters · Bad code should be stopped before it is merged, not found in production.

- Why code quality matters, and SonarQube use cases
 - Installing SonarQube on Linux
 - Quality profiles and quality gates
 - Reading issues, coverage, and duplication
 - Users and admin activities
 - SonarQube architecture
 - Configuring projects for scanning
 - Scanning Java, Node.js, and Python code
 - Failing a pipeline on a quality gate
- o *and more*

Applied to i27Helpdesk · Every app is scanned in its pipeline, and a failed quality gate stops the release.

06 Image security with Trivy



Why it matters · A container image can ship known vulnerabilities. Catch them before release.

- Why container images need scanning
 - Installing and running Trivy
 - Reading reports and severity levels
 - Running scans inside the pipeline
 - Vulnerabilities and CVEs, in practice
 - Scanning container images
 - Blocking releases on critical findings
- o *and more*

Applied to i27Helpdesk · Every app's image is scanned before it is pushed, so nothing vulnerable reaches an environment.

07 Docker



Why it matters · Every one of the 6 apps ships as a container image.

- Monolith vs microservices, and why containers
- Docker installation options
- Container lifecycle and port binding
- Persistence, and storage drivers
- Images, layers, and tags in the real world
- Committing changes to an image
- Multi-stage builds
- Docker architecture and the Docker engine
- VM vs container
- Bridge, user-defined, and host networking
- Volumes, bind mounts, and tmpfs mounts
- Inspecting, pulling, and pushing images
- Writing Dockerfiles and instructions
- Deploying a private registry

○ *and more*

Applied to i27Helpdesk · Write production Dockerfiles for all 6 apps, using multi-stage builds to keep images small and secure.

08 Artifact Registry



Why it matters · Every release needs a versioned image, stored where the cluster can pull it.

- What an artifact repository is, and why teams use one
- Authenticating Docker to the registry
- Tagging images with commits and versions
- Creating and organising repositories
- Pushing and pulling images
- Access control for pipelines and clusters

○ *and more*

Applied to i27Helpdesk · Every app publishes commit-tagged images that the Kubernetes cluster pulls on deploy.

09 CI/CD with GitHub Actions



Why it matters · This is how every change to i27Helpdesk gets built, checked, and released.

- Workflows, jobs, steps, and actions
- Inputs for choosing an environment
- Secrets and variables
- Self-hosted runners
- Environments and approval gates
- Reusable workflows across apps
- Troubleshooting failed runs
- Triggers: push, path filters, and manual runs
- Job dependencies and outputs
- GitHub-hosted runners
- OIDC: keyless access to the cloud
- Caching and matrix builds
- Deploying to Kubernetes and verifying rollouts

○ *and more*

Applied to i27Helpdesk · One pipeline per app: build, scan, push, and deploy, promoted from dev to prod with approvals.

STAGE 03 · CKA / CKAD LEVEL

Kubernetes & observability

A Kubernetes deep dive to CKA and CKAD level, then running and watching the product in production.

10

Kubernetes core



Why it matters · i27Helpdesk runs on Kubernetes in every environment, from dev to prod.

■ INTRODUCTION

- History of Kubernetes
- The need for an orchestration tool
- Introduction to Kubernetes

■ BOOTSTRAP A CLUSTER ON CLOUD

- Getting infrastructure ready on the cloud
- Bootstrapping master and worker nodes with kubeadm
- Managed Kubernetes clusters

■ ARCHITECTURE

- Kubernetes components
- Master and worker node architecture
- Kube API Server
- etcd
- Kube Controller Manager
- Kube Scheduler
- Kubelet
- Kube Proxy

■ PODS AND DEPLOYMENTS

- Pods, and container vs pod
- Namespaces in real projects
- Understanding YAML
- Creating pods: imperative and declarative
- Labels, selectors, and annotations
- Why Deployments, and the limits of pods
- ReplicaSets vs Replication Controllers
- Deployments deep dive
- Scaling with RC, RS, and Deployments
- Deployment strategies used in production
- Blue/green and canary deployments
- Rolling updates and rollbacks
- Jobs and CronJobs
- DaemonSets

■ SERVICES AND INGRESS

- What Services are, and why they matter
- Services deep dive
- ClusterIP, NodePort, and LoadBalancer
- Headless Services
- Ingress controllers, and how to create them

and more

Applied to i27Helpdesk · Deploy all 6 apps to a managed cluster, with a namespace per environment, Services, and Ingress.

11

Kubernetes advanced



Why it matters · Production workloads have to be scheduled, configured, connected, and kept healthy.

● HEALTH AND AUTOSCALING

- Liveness and readiness probes
- Real use cases of probes
- Horizontal Pod Autoscaler
- Automated scaling with HPA and Metrics Server
- Logging, monitoring, and debugging apps

● RESOURCES, QOS, AND QUOTAS

- Resource requests and limits for CPU and memory
- Quality of Service (QoS) classes
- Namespace resource quotas

● ADVANCED SCHEDULING

- How scheduling works in Kubernetes
- Manually scheduling pods on nodes
- Node affinity and anti-affinity
- Node taints and pod tolerations
- Init containers
- DaemonSets
- Static pods
- Working with resource limits

● NETWORKING

- Kubernetes networking overview
- CNI in Kubernetes
- DNS in Kubernetes
- Ingress networking
- API deprecation

● STORAGE

- Why Kubernetes storage matters
- Ephemeral vs persistent storage
- Volumes: concepts and types
- Persistent Volumes and Persistent Volume Claims
- Dynamic volume provisioning
- Storage classes and provisioners
- Volume access modes
- Managing storage resources

● CONFIGMAPS, SECRETS, AND ENVIRONMENT VARIABLES

- ConfigMaps and their real-world use cases
- Creating and using ConfigMaps
- Mounting ConfigMaps as volumes
- What a Secret is
- ConfigMaps vs Secrets
- Creating and using Secrets
- Setting environment variables in pods
- Environment variables from ConfigMaps and Secrets

and more

Applied to i27Helpdesk · Give every app health probes, resource limits, and its configuration and secrets, the way production clusters need.

12

Kubernetes security and operations



Why it matters · Clusters must be secured, upgraded, and fixed while the product keeps running.

● ROLE-BASED ACCESS CONTROL (RBAC)

- What RBAC is, and its use cases
- Authentication mechanisms
- TLS certificates for cluster components
- Kubeconfig and its structure
- Managing and switching between clusters
- Roles and RoleBindings
- ClusterRoles and ClusterRoleBindings
- Service accounts

● NETWORK POLICIES

- What a network policy is
- Components of a network policy
- Creating and applying network policies
- Real-world use cases

● MULTI-CONTAINER PODS

- Multi-container pods and use cases
- Init containers
- Static pods

● CLUSTER MANAGEMENT

- High availability in Kubernetes
- Working with OS upgrades
- Draining a node safely for maintenance
- Upgrading a Kubernetes cluster
- Backing up and restoring etcd

● MANAGED KUBERNETES

- Creating a highly available managed cluster
- Cluster types for different requirements
- Dynamic volume allocation with storage classes
- Zero-downtime cluster upgrades
- Private container registries

● TROUBLESHOOTING

- Checking cluster and node logs
- Troubleshooting a broken cluster
- Troubleshooting broken applications
- Common issues when deploying applications

● HELM AND TOOLING

- Helm introduction
- Installing Helm
- Creating Helm charts
- Lens IDE

and more

Applied to i27Helpdesk · Lock down access with RBAC and network policies, and upgrade the cluster with zero downtime while the product keeps running.

13

Observability with Prometheus and Grafana



Why it matters · Once it is live, you need to know when something goes wrong, before users do.

- Monitoring concepts, and why observability matters
- Collecting metrics from Kubernetes workloads
- Building Grafana dashboards
- Using metrics to troubleshoot issues
- Prometheus architecture
- Querying metrics
- Alerting rules and notifications
- and more

Applied to i27Helpdesk · Dashboards and alerts for the product's apps, used to spot and fix problems in production.

STAGE 04 · BUILT LIVE IN CLASS

i27Helpdesk, end to end

You join the team as its DevOps engineer and take **i27Helpdesk, a real 6-app product**, from source code to a monitored production release. Every phase is built live in class, and everything you create is shared with you.

01

Onboarding

- The company scenario, and your role on the team
- Product architecture: how the 6 apps work together
- The environments, and the release plan

✓ A clear plan for taking the product live

02

Run it locally, end to end

- Run the Next.js UI and the Node.js, Java, and Python services on localhost
- Connect every app to MySQL
- Configuration and environment variables

✓ The whole product working on your machine

03

Containerise all 6 apps

- Production Dockerfiles for every app
- Multi-stage builds for small, secure images

✓ A production image for every app

04

CI for every app

- A GitHub Actions pipeline per app
- Triggered only by that app's changes
- Builds and tests with Maven, npm, and Python tooling

✓ Every commit builds and tests itself

05

Quality, security, and images

- SonarQube quality gates in every pipeline
- Trivy scans on every image
- Commit-tagged images stored in Artifact Registry

✓ Only checked, scanned images get published

06

Keyless access to the cloud

- OIDC from GitHub Actions to Google Cloud, with no stored keys
- GitHub-hosted and self-hosted runners
- Secrets and variables, handled safely

✓ Pipelines reach the cloud securely

07

Run it on Kubernetes

- A managed cluster, with a namespace per environment
- Deployments with readiness and liveness probes
- Secrets, Services, and NGINX Ingress
- DNS: one public domain per environment, with HTTPS certificates (cert-manager, Let's Encrypt)
- Internal services stay private inside the cluster

✓ The product live on Kubernetes, behind one public domain

08

Promote to production

- Release through dev, test, stage, and prod
- Approvals before protected environments
- Verified rollouts on every deploy

✓ A gated, repeatable release to prod

09

Operate, break, and fix

- Prometheus metrics, Grafana dashboards, and alerts
- Handle the failures we cause on purpose
- Troubleshoot like you would on call

✓ A product you can run, and explain in interviews

LIVE IN PROD

i27Helpdesk, delivered end to end by you

BREAK & FIX LABS

Things will break. That is the point.

INCIDENT

Broken pipeline

A build fails, a quality gate blocks the merge, or an image scan finds a critical vulnerability.

YOU FIX IT

Read the logs, find the cause, and get the pipeline green again.

INCIDENT

Bad release

A new version will not start, pods keep restarting, or the rollout never completes.

YOU FIX IT

Debug the deployment, roll back safely, and ship the fix.

INCIDENT

Database issues

An app cannot reach MySQL, credentials are wrong, or the schema does not match.

YOU FIX IT

Trace connections, secrets, and configuration until every app is healthy.

...and more, from access and permission errors to failed rollouts.

CAREER READINESS

Walk into interviews prepared



Roles and responsibilities

What a DevOps engineer owns on a real team.



Day to day on the job

The everyday work: releases, incidents, and requests.



Resume building

Built around the i27Helpdesk project you delivered.



Interview preparation

Explain the project end to end, decision by decision.



Bonus recorded courses

Included with the program, at no extra cost.

Jenkins end to end

Ansible

Python core & advanced

i27Academy

Ready to join? Talk to us.

Ask about the next batch and how to enroll.

CALL & WHATSAPP +91 93810 62032

EMAIL support@i27academy.com

WEBSITE i27academy.com

YOUTUBE youtube.com/@i27academy