internStudio

DEVOPS INTERNSHIP

TASK GUIDE

Automate the path to production.
Build, ship and operate reliably.

SIX ADVANCED PROJECTS — CHOOSE ANY TWO

01 **CI/CD Pipeline for a Multi-Service Application**

02 **Infrastructure as Code Provisioning System**

03 **Containerized Microservices Orchestration**

04 **Automated Monitoring & Incident Alerting Pipeline**

05 **GitOps-Based Deployment Workflow**

06 **Centralized Log Aggregation & Observability Stack**

INTERNSHIP	REQUIRED TASKS	FOCUS
DevOps	Complete any 2 projects	CI/CD · Infrastructure as Code · Containers · Observability

INTERNSHIP INSTRUCTIONS

Your Internship Tasks

Select and complete **any 2 projects** from the task list. Each should demonstrate real automation of build, deployment or operations — not a single manual deploy you screenshot once.

Plan	Automate	Build	Deploy	Monitor	Iterate

EVERY PROJECT SHOULD INCLUDE

- [ ] Fully automated pipeline or provisioning — no manual steps
- [ ] Version-controlled configuration, not console clicks
- [ ] Reasonable failure handling (rollback, retries, alerts)
- [ ] Secrets handled securely, never hard-coded
- [ ] Logs or metrics you can actually inspect
- [ ] Documentation of the architecture and flow
- [ ] Free-tier or low-cost resources wherever possible
- [ ] Evidence the pipeline actually ran successfully

BEFORE YOU SUBMIT

- [ ] Complete any 2 projects from the list
- [ ] Push all configuration to GitHub
- [ ] Include screenshots or logs proving it ran
- [ ] Add a README with setup and run steps
- [ ] Document the architecture
- [ ] Submit details via the submission form
- [ ] Share your work on LinkedIn as required

> **Suggested technologies:** GitHub Actions or Jenkins · Docker · Terraform · Kubernetes · Prometheus/Grafana. Free-tier cloud services wherever possible.

Submit your completed work here

Once both projects are running, documented and pushed to GitHub, submit your project details through the official submission form.

https://forms.gle/vxL5JE8cvRXXfDBv8

> **A note on quality.** A pipeline you ran once by hand is not automation. It should run reliably, again, without you present, and tell you clearly when it fails.

01 CI/CD Pipeline for a Multi-Service Application

Build a complete CI/CD pipeline that builds, tests and deploys a multi-service application automatically.

REQUIREMENTS

- Trigger automatically on push/PR
- Build and tag container images
- Deploy automatically to a target environment
- Notify on pipeline failure
- Run automated tests as a required gate
- Push images to a registry
- Support rollback to a previous version
- Document each pipeline stage

Focus: Continuous integration/deployment, automated gating and rollback strategy.

02 Infrastructure as Code Provisioning System

Provision real cloud infrastructure entirely from code, for at least two distinct environments.

REQUIREMENTS

- Define infrastructure declaratively (Terraform or similar)
- Use variables/modules instead of duplicated code
- Support safe teardown of all provisioned resources
- Document the resource graph
- Support multiple environments (dev/staging) from shared config
- Store state securely, not committed to Git
- Apply least-privilege access to provisioned resources
- Demonstrate a full provision-then-destroy cycle

Focus: Infrastructure as code, environment management and safe, repeatable provisioning.

03 Containerized Microservices Orchestration

Deploy a multi-service application on Kubernetes with proper service discovery and configuration.

REQUIREMENTS

- Containerize each service independently
- Configure service discovery between services
- Configure resource limits and health checks
- Demonstrate a rolling update with zero downtime
- Deploy services to Kubernetes with manifests or Helm
- Use ConfigMaps/Secrets for configuration
- Demonstrate scaling a service up and down
- Document the cluster architecture

Focus: Container orchestration, service discovery and zero-downtime deployment practice.

04 Automated Monitoring & Incident Alerting Pipeline

Build a monitoring stack that tracks a real application's health and alerts automatically on problems.

REQUIREMENTS

- Instrument an application to expose metrics
- Build dashboards for key health indicators
- Route alerts to a real notification channel
- Avoid alert fatigue (sensible thresholds, not noise)
- Collect and store metrics (Prometheus or similar)
- Define alert rules for meaningful thresholds
- Demonstrate an alert firing on an induced failure
- Document what each alert means and how to respond

Focus: Metrics instrumentation, alerting design and on-call-style incident response.

05 GitOps-Based Deployment Workflow

Build a deployment workflow where the Git repository is the single source of truth for what is running.

REQUIREMENTS

- Store desired deployment state declaratively in Git
- Trigger deployments automatically on a merge
- Support rollback via a Git revert
- Document the promotion flow between environments
- Use a GitOps controller (ArgoCD/Flux or equivalent) to sync it
- Support drift detection between actual and desired state
- Separate configuration per environment
- Demonstrate a full change-to-deployed cycle

Focus: GitOps principles, drift detection and Git-driven deployment promotion.

06 Centralized Log Aggregation & Observability Stack

Build a centralized logging stack that aggregates logs from multiple services and makes them searchable.

REQUIREMENTS

- Emit structured logs from at least two services
- Support searching and filtering across services
- Build a dashboard of common queries
- Demonstrate diagnosing a simulated issue using only logs
- Aggregate logs centrally (ELK/EFK or equivalent)
- Correlate logs to a request or trace ID
- Set retention limits appropriate for the environment
- Document the log pipeline architecture

Focus: Structured logging, centralized aggregation and log-driven debugging.

FINAL STEP

Ready to Submit?

Verify everything below before you send your work in.

SUBMISSION

- [ ] GitHub repository
- [ ] Configuration files included
- [ ] Evidence of a successful run
- [ ] Submission form completed
- [ ] LinkedIn post where required

DOCUMENTATION

- [ ] Architecture documented clearly
- [ ] README with setup and run steps
- [ ] Screenshots or logs as evidence

AUTOMATION

- [ ] No manual steps required to reproduce
- [ ] Configuration is version-controlled
- [ ] Pipeline runs reliably, repeatably

RELIABILITY

- [ ] Failure handling in place (rollback/retry/alert)
- [ ] Secrets never committed to GitHub
- [ ] Logs or metrics are inspectable

Important. Complete any 2 projects. Your work must demonstrate real automation, not a manual process you documented afterward.

Submission Form

Submit your repository, live link and screenshots here.

https://forms.gle/vxL5JE8cvRXXfDBv8

internStudio

LEARN · BUILD · GROW

www.internstudio.in

team.internstudio@gmail.com