INTERNSHIP PROJECTS (2 WEEKS)

Ministry of MSME, Govt. of India

PROJECT GUIDELINES

- **Choose any 1 or 2 projects from the given list.**
- **You are free to improvise — take the given project as a base and modify it as you like.**
- **You can use any tools, technologies, or steps you're comfortable with — there are no restrictions.**
- **Focus and work sincerely so that you have complete clarity and can explain the project confidently in interviews.**
- **Go through the Top 50 Interview Questions for your domain (attached at the end).**
- **Update your project status regularly using the provided Google Form while working on the project**
- **YOU CAN CHOOSE ANY DATASET RELAVENT TO THE PROJECT.**

.

After project completion, prepare a 1–2 page report in PDF format, containing:

- Introduction
- Abstract
- Tools Used
- Steps Involved in Building the Project
- Conclusion

◆ Note: Report must not exceed 2 pages.

DEAR INTERNS,
Status updation link will be shared every 3/4 days once in the group to update the status of your projects

Final Project Submission Date:

08 September 2025: Submission of your final project GitHub repository link with all deliverables and the project report.
Final submission links will be shared later.

❗ READ ALL THE GUIDELINES CAREFULLY ❗

LIST OF PROJECTS

1. **Self-Healing Infrastructure with Prometheus, Alertmanager & Ansible**
   - Objective: Automatically detect service failures and recover them using alerts and automation.
   - Tools: Prometheus, Alertmanager, Ansible, Shell Scripting, Ubuntu VM/Docker
   - Mini Guide:
     - Deploy a sample service (e.g., NGINX).
     - Monitor uptime with Prometheus.
     - Set thresholds (e.g., service down or CPU > 90%).
     - On alert trigger, Alertmanager executes an Ansible playbook to restart the service or system.
   - Deliverables:
     - Prometheus config
     - Alertmanager webhook setup
     - Ansible playbook
     - Demo logs or screenshots of auto-healing in action

2. **Multi-Cloud Auto Deployment using Terraform (AWS + GCP Free Tier)**
   - Objective: Provision resources in both AWS and GCP simultaneously and validate auto-deployment with a single command.
   - Tools: Terraform, AWS Free Tier, GCP Free Tier, NGINX, DNSMasq (local)
   - Mini Guide:
     - Configure Terraform providers for both AWS & GCP.
     - Deploy web servers and configure health checks.
     - Use local DNS or HTTP checks to simulate routing based on availability.
   - Deliverables:
     - Terraform scripts
     - Infrastructure diagram
     - Screenshots from both cloud dashboards
     - Deployment report with validation steps

3. **Complete Observability System (Metrics, Logs & Traces)**
   - Objective: Build an integrated monitoring system that includes performance metrics, centralized logs, and request tracing.
   - Tools: Prometheus, Grafana, Loki (logs), Jaeger (tracing), Docker Compose
   - Mini Guide:
     - Containerize a sample Python/Node app with custom logs.
     - Connect Prometheus to scrape metrics.
     - Integrate Loki for logging and Jaeger for tracing HTTP requests.
     - Display everything on Grafana dashboards.
   - Deliverables:
     - docker-compose.yml
     - Dashboards JSON
     - Log samples, trace visualizations
     - Report explaining insights observed

4. **CI/CD Pipeline with GitHub Actions & Docker (No Cloud Needed)**
- Objective: Set up a full CI/CD pipeline that builds a Docker image, runs tests, and deploys locally.
- Tools: GitHub Actions, Docker, Docker Hub (free), Minikube or local VM
- Mini Guide:
  - Write a Dockerfile and docker-compose.yml.
  - Configure GitHub Actions to run tests, build the image, and push to Docker Hub.
  - Use Minikube or a local VM to pull and run the image.
- Deliverables:
  - GitHub repo with workflows
  - Docker image link
  - CI/CD workflow results
  - Screenshots of the deployed app

5. **FinOps Dashboard for Cloud Cost Visibility (Free Tier Usage Tracker)**
- Objective: Monitor and display cloud resource usage to detect free-tier limit breaches before billing starts.
- Tools: AWS Cost Explorer API or GCP Billing API (Free), Python, SQLite, Grafana
- Mini Guide:
  - Use APIs to fetch daily resource usage and estimated costs.
  - Store data in SQLite and visualize trends in Grafana.
  - Add basic rules to flag "at-risk" services that may incur cost.
- Deliverables:
  - Python scripts to fetch and parse API data
  - Dashboard screenshots
  - Alert/flag logic documentation
  - Weekly usage report

6. **Scalable Static Website with S3 + Cloudflare + GitHub Actions**
- Objective: Host and auto-deploy a static website using S3 (free tier) with global CDN and HTTPS via Cloudflare, triggered through GitHub commits.
- Tools: AWS S3 (Free), Cloudflare (Free), GitHub Actions, HTML/CSS, Bash
- Mini Guide:
  - Create a static HTML site and push to GitHub.
  - Set up GitHub Actions to sync content to an S3 bucket.
  - Connect S3 to a custom domain via Cloudflare and enable SSL.
  - Add cache settings and versioning.
- Deliverables:
  - GitHub Actions CI/CD workflow
  - Cloudflare + S3 integration steps
  - Live website link with HTTPS
  - Screenshot + deployment report

7. **Kubernetes-Based Canary Deployment with K3s and Istio**

- Objective: Simulate modern canary deployments with traffic splitting between stable and new app versions.
- Tools: K3s (lightweight Kubernetes), Istio (Free), Docker, Helm, Node.js/Python app
- Mini Guide:
  - Deploy a basic app with 2 versions to K3s.
  - Set up Istio to route 80% to stable, 20% to canary.
  - Monitor and switch traffic based on performance.
  - Roll back or promote versions accordingly.
- Deliverables:
  - YAML files for K3s deployment
  - Istio gateway & routing config
  - Traffic logs or metrics dashboard
  - Canary deployment strategy document

8. **Open-Source Incident Management System**

- Objective: Build a basic incident management portal to log, track, and resolve infra or app issues with role-based access.
- Tools: Python (Flask/Django), SQLite, Bootstrap, Docker, Git
- Mini Guide:
  - Create REST APIs to create, update, assign, and resolve incidents.
  - Add email notifications using SMTP.
  - Use Docker to containerize the service.
  - Store issues in SQLite and version control in Git.
- Deliverables:
  - Source code (hosted on GitHub)
  - Docker image
  - Demo video/screenshots
  - Sample issues logged + resolved

9. **Local DevOps Sandbox for Monitoring & Alerting (All-in-One VM)**

- Objective: Create a fully working sandbox environment for DevOps training with all monitoring tools pre-installed.
- Tools: VirtualBox, Vagrant or shell scripts, Prometheus, Grafana, Node Exporter, Alertmanager
- Mini Guide:
  - Automate provisioning of a VM with pre-installed monitoring stack.
  - Pre-configure alerts for CPU, disk, and service health.
  - Provide README so others can use this for learning.
- Deliverables:
  - VM setup script or Vagrantfile
  - Default dashboards & alert configs
  - GitHub repo with installation guide
  - Screenshots of working sandbox

10. **GitOps Workflow using ArgoCD on Kubernetes**

- Objective: Implement GitOps by syncing Kubernetes deployment states directly from a Git repository.
- Tools: K3s/Minikube, ArgoCD (Free), GitHub, Docker
- Mini Guide:
  - Deploy ArgoCD on Kubernetes.
  - Push app deployment manifests to a Git repo.
  - Configure ArgoCD to auto-sync and deploy from Git.
  - Update versions via Git commits and observe changes.
- Deliverables:
  - Git repo with manifest files
  - ArgoCD screenshots showing sync in action
  - Video/notes explaining GitOps flow
  - Resume line: "Implemented GitOps pipeline using ArgoCD and Kubernetes"

❗ TOP 50 INTERVIEW QUESTIONS FOR DEVOPS ENGINEERS ❗

1.What is DevOps, and why is it important?
2.What are the key benefits of using DevOps in software development?
3.Can you explain the concept of Continuous Integration (CI) and Continuous Deployment (CD)?
4.What is Infrastructure as Code (IaC), and how is it useful in DevOps?
5.What are some popular CI/CD tools you have worked with?
6.Explain the difference between GitOps and traditional DevOps.
7.How do you ensure high availability and scalability in a cloud environment?
8.What is containerization? How is it different from virtualization?
9.What is Docker, and how does it work?
10.Explain the use of Docker Compose and Docker Swarm.
11.How do Kubernetes and Docker work together?
12.What is the purpose of Kubernetes, and what problems does it solve?
13.How do you deploy and manage applications on Kubernetes?
14.What is Helm, and how does it help in managing Kubernetes applications?
15.What is a microservices architecture, and how does DevOps support it?
16.Can you explain the concept of blue-green deployment and how to implement it?
17.What are rolling updates, and how are they different from blue-green deployment?
18.What is an automated testing framework, and how does it fit into a DevOps pipeline?
19.How would you handle logging and monitoring in a DevOps environment?

20.What is the purpose of Prometheus and Grafana in DevOps?
21.Can you explain the difference between a stateful and stateless application?
22.What is a DevOps pipeline, and how do you set it up?
23.What are some tools for configuration management, and what are their advantages?
24.How do you manage secrets (like API keys, passwords) in a DevOps environment?
25.What is Terraform, and how does it help with infrastructure automation?
26.How does Jenkins help in automating the build and deployment process?
27.What are the advantages of using Ansible over other configuration management tools like Chef or Puppet?
28.Explain the concept of "Infrastructure as Code" and name some tools that implement it.
29.What is the difference between continuous integration and continuous delivery?
30.Can you explain the concept of "serverless" computing, and when would you use it?

31.How do you implement security within a CI/CD pipeline?
32.What is the purpose of version control systems (VCS), and why are they important in DevOps?
33.What is a rollback, and how do you handle rollbacks in production environments? 34.How do you
ensure data consistency in a microservices environment? 35.Can you explain the difference
between Docker containers and virtual machines? 36.What is the role of a load balancer in a
DevOps setup? 37.Explain the purpose of an API gateway in a microservices architecture. 38.What
is the difference between a canary release and feature flagging? 39.What is the concept of
"immutable infrastructure" in DevOps? 40.How do you handle system failures and downtime in a
DevOps environment?

41.Can you explain the concept of "DevSecOps" and how it relates to DevOps?
42.How would you monitor the performance of a distributed application?
43.What is a self-healing infrastructure, and how can you implement it?
44.What is Git, and why is it commonly used in DevOps environments?
45.How do you handle application scaling in a cloud environment?
46.What is the difference between a monolithic and microservices application, and how does DevOps impact each?
47.Can you explain what "chaos engineering" is and why it's important in DevOps?
48.How do you perform load testing in a continuous delivery pipeline?
49.What is the role of a DevOps engineer in managing cloud infrastructure?
50.How do you ensure the security and compliance of applications in a DevOps environment?