DevOps Internship Preparation: Step-by-Step Practice Guide

This document provides hands-on practice scenarios for aspiring DevOps interns using local setups or free-tier cloud resources. Each section includes goals, tools involved, and step-by-step instructions.

Sample Job Description:

1. Assist in setting up and managing:
CI/CD pipelines. Support infrastructure as code (IaC) using tools like Terraform or Ansible. Help monitor system performance and troubleshoot deployment issues. Learn and support automation scripts for build and release processes. Collaborate with developers and operations teams to streamline workflows.

2. Specify Required Skills and Qualifications:
Must-Have Skills: Basic knowledge of Linux/Unix commands and shell scripting. Understanding of Git and version control. Familiarity with CI/CD tools like GitHub Actions, Jenkins, GitLab CI. Exposure to cloud platforms (e.g., AWS, Azure, or GCP).

3. Nice-to-Have Skills:
Experience with Docker and containerization. Understanding of networking basics and system architecture. Knowledge of infrastructure as code (Terraform, CloudFormation). Basic familiarity with monitoring/logging tools like Prometheus, Grafana, or ELK stack.

---

1. Linux & Shell Scripting

Goal:

Learn basic system management and automation using bash scripting.

Practice:

Task 1: Disk Usage Monitor

```
#!/bin/bash
THRESHOLD=80
USAGE=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ "$USAGE" -gt "$THRESHOLD" ]; then
  echo "Disk usage above ${THRESHOLD}% at $(date)" >> /var/log/disk_alert.log
fi
```

Task 2: Scheduled System Report

- Use `crontab -e` to schedule the above script to run every 15 minutes.

---

2. Git Version Control

Goal:

Understand version control workflows and collaboration.

Practice:

Task 1: Git Basics

- Create a GitHub repository.
- Clone it locally: `git clone <repo-url>`
- Create a file, commit, push.

Task 2: Branch and Merge

- Create a new branch: `git checkout -b feature-branch`
- Modify files and commit.
- Merge with main: `git checkout main` → `git merge feature-branch`
- Simulate a merge conflict and resolve it.

---

3. CI/CD Tools (GitHub Actions / Jenkins)

Goal:

Build basic continuous integration pipelines.

Practice:

Task 1: GitHub Actions

1. Create `.github/workflows/python-test.yml`

```
name: Python CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: 3.10
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest
```

Task 2: Jenkins (if available)

- Create a Freestyle project
- Pull from GitHub, run a script, archive results.

---

4. AWS Free-Tier Cloud

Goal:

Familiarize with basic AWS services: EC2, S3.

Practice:

Task 1: Host Static Website on S3

- Create S3 bucket
- Enable static site hosting
- Upload `index.html`
- Set bucket policy for public read

Task 2: EC2 Provisioning

- Launch EC2 (Amazon Linux, t2.micro)
- SSH into instance
- Install Apache: `sudo yum install httpd -y && sudo systemctl start httpd`

---

5. Terraform (IaC)

Goal:

Provision infrastructure declaratively.

Practice:

Task 1: Basic EC2 Instance

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "my_ec2" {
  ami           = "ami-0c02fb55956c7d316" # Amazon Linux 2
  instance_type = "t2.micro"

  tags = {
    Name = "devops-test-instance"
  }
}

output "ec2_public_ip" {
  value = aws_instance.my_ec2.public_ip
}
```

- Commands: `terraform init`, `apply`, `destroy`

---

6. Docker

Goal:

Containerize and run a basic application.

Practice:

Task 1: Hello World App

```
# app.py
print("Hello from Docker!")
```

Dockerfile

```
FROM python:3.10
COPY app.py /app.py
CMD ["python", "/app.py"]
```

- Build: `docker build -t hello-docker .`
- Run: `docker run hello-docker`

---

7. Monitoring (Optional Docker Demo)

Goal:

Understand monitoring basics using Prometheus + Grafana.

Practice:

- Use Docker Compose from: https://github.com/stefanprodan/dockprom
- Run with: `docker-compose up -d`

---

8. Capstone Integration: CI/CD + Docker + Terraform

Goal:

Automate end-to-end deployment.

Practice:

1. Code a simple web app in Python or Node.js.
2. Write a Dockerfile.
3. Build and push Docker image (if Docker Hub is available).
4. Provision EC2 with Terraform.
5. SSH into EC2 from GitHub Actions using deploy keys.
6. Pull image or code, run container.

Interview Questions (70 Total)

A. Linux & Shell

1. What are some basic Linux commands every DevOps engineer should know?
2. How do you check CPU and memory usage in Linux?
3. What is a cron job and how is it configured?
4. Write a bash script to send an alert if disk usage exceeds 90%.
5. How would you troubleshoot a failing script scheduled with cron?

B. Git & Version Control

6. What is the difference between Git and GitHub?
7. How do you resolve a merge conflict?
8. What is the purpose of .gitignore?
9. Explain Git branching strategy for a small team project.
10. How would you revert a specific commit?

C. CI/CD Tools

11. What is CI/CD, and why is it important?
12. Explain a simple GitHub Actions workflow.
13. What's the difference between on: push and on: pull_request?
14. How do you store secrets in GitHub Actions?
15. How do you configure Jenkins to pull code from GitHub?

D. Cloud Basics

16. What is the purpose of an EC2 instance?
17. How do you secure an EC2 instance?
18. What is an S3 bucket?
19. What are IAM roles and policies?
20. How do you SSH into an EC2 instance?

E. Terraform

21. What is Infrastructure as Code?
22. Explain terraform init, plan, and apply.

23.What are Terraform providers and modules?

24.How do you handle secrets in Terraform?25.What happens if you manually delete a resource provisioned by Terraform?

F. Docker

26.What is Docker and how is it different from a VM?

27.How do you build a Docker image?

28.What is the use of a Dockerfile?

29.How do you persist data in a Docker container?

30.What is Docker Hub?

G. Monitoring & Logging

31.What is the role of Prometheus?

32.How does Grafana work with Prometheus?

33.What metrics would you monitor for a web server?

34.Explain alerting thresholds.

35.What is the ELK stack?

H. CI/CD Pipeline

36.How would you deploy a Python app to AWS using GitHub Actions?

37.What are deploy keys in GitHub?

38.How can you automate EC2 provisioning and deployment in a pipeline?

39.What would you do if a deployment step fails?

40.How do you roll back a bad deployment?

I. DevOps Practices

41.What does "shift left" mean?

42.How do DevOps principles improve development?

43.Why is automation important in DevOps?

44.What’s the difference between DevOps and SRE?

45.How do you ensure security in DevOps?

J. Behavioral / Scenario-Based (30 total)

46.You deployed a script to production and it broke something — what do you do?

47.Your GitHub Actions workflow is taking too long — how would you optimize it?

48.A junior teammate is stuck with Terraform errors — how would you help?

49.Describe a time you automated a task and what the result was.50.How would you explain a project to a non-technical stakeholder?

51.EC2 instance provisioning failed in terraform apply. What do you do?

52.Your Jenkins job fails randomly. How do you troubleshoot?

53.Docker container is running but the app is unreachable. Steps?

54.Secrets are leaking into logs — what's your action?

55.Monitoring shows CPU spikes every 15 mins — how to debug?

56.Team pushes code without tests — what’s your response?

57.GitHub Actions fails only on pull requests. Why might that happen?

58.You need to setup alerting for disk usage on multiple EC2s. How?

59.You see high latency in a deployed app. What do you check?

60.How do you structure a repo for a Terraform + Docker + App project?

61.Your teammate committed sensitive data. What do you do?

62.Docker image size is too large — how do you reduce it?

63.Git history is messy. Do you squash commits? Why?

64.You have to setup CI/CD for a new repo — what’s your plan?

65.How do you design a basic disaster recovery plan for a web app?

66.Monitoring system is down — how do you proceed?

67.EC2 was stopped accidentally — how do you protect against that?

68.You need to run the same CI workflow on multiple repos. Best practice?

69.You want to train others in Terraform — how would you begin?

70.How do you maintain IAM role security across environments?

Bonus: Tools and Technologies

9. Kubernetes (minikube/kind)

Goal:

Learn Kubernetes basics: Pods, Deployments, Services.

Practice:

Task 1: Install and Start Cluster

- Install minikube or kind
- Start minikube: `minikube start`
- Verify: `kubectl get nodes`

Task 2: Deploy Hello World Pod

```
# hello-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
spec:
  containers:
    - name: hello
      image: busybox
      command: ['sh', '-c', 'echo Hello from Kubernetes && sleep 3600']Apply: kubectl apply -f hello-pod.yaml
```

- View: `kubectl logs hello-pod`

Task 3: Create Deployment + Service

```
# hello-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
        - name: hello
          image: hashicorp/http-echo
          args:
            - "-text=Hello from Kubernetes"
---
apiVersion: v1
kind: Service
metadata:
  name: hello-service
spec:
  type: NodePort
  selector:
    app: hello
  ports:
    - port: 80
      targetPort: 5678
      nodePort: 30007
```

- **Apply:** `kubectl apply -f hello-deployment.yaml`
- **Access:** `minikube service hello-service` or open `http://localhost:30007`

---

10. Helm

Goal:

Package, install, and manage Kubernetes applications.

Practice:

Task 1: Install Helm

- Follow: https://helm.sh/docs/intro/install/

Task 2: Use Existing Chart

- Add chart repo: `helm repo add bitnami https://charts.bitnami.com/bitnami`
- Search charts: `helm search repo nginx`
- Install chart:

  ```
  helm install my-nginx bitnami/nginx
  ```

  ```
  kubectl get svc --namespace default
  ```

Task 3: Create Your Own Chart:

```
helm create hello-chart
```

```
cd hello-chart/templates
```

```
# Edit deployment.yaml and service.yaml to match your app
```

- Install: `helm install hello-app ./hello-chart`
- Upgrade: `helm upgrade hello-app ./hello-chart`
- Uninstall: `helm uninstall hello-app`

---

11. GitOps with Argo CD

Goal:

Automate Kubernetes deployment via Git repository.

Practice:

Task 1: Install Argo CD on minikube/kind:

```
kubectl create namespace argocd
```

```
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/
install.yaml
```

- Port-forward UI:
  `kubectl port-forward svc/argocd-server -n argocd 8080:443`
- Login: `admin` / initial password from:
  `kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`

Task 2: Deploy App from Git

- Push `hello-deployment.yaml` (from Kubernetes section) to GitHub
- Create Argo App:

```
argocd app create hello-app \
 --repo https://github.com/your-username/your-repo.git \
 --path manifests \
 --dest-server https://kubernetes.default.svc \
 --dest-namespace default
```

- Sync:
  `argocd app sync hello-app`
- View:
  `argocd app get hello-app`

Bonus: Mock Interview Drill

Give a 20-minute challenge:

> Create a GitHub repo, add a Python script, setup CI/CD using GitHub Actions, containerize it with Docker, and deploy to EC2 using Terraform.

This guide helps build real skills that closely match internship expectations in a step-by-step manner.

✅ Free & Community-Driven Platforms:

1. **Pramp**
    - Offers free peer-to-peer mock interviews.
    - Good for behavioral + system design + technical problem solving.
2. **[Exercism.io + Live Discord/Slack Groups]**
    - Pair with DevOps learning paths and find partners for mock interviews in community channels.
3. **Reddit Communities:**
    - r/devops
    - r/cscareerquestions
    - r/learnprogramming
    - You can post asking for mock interview partners.
4. **Meetup.com / LinkedIn Events:**
    - Search for “DevOps mock interview” or “interview prep” events.
5. **Google Meet + GitHub Collaboration:**
    - Have your interns pair up and simulate interviews from the 70 questions listed in your doc.
    - Rotate roles (interviewer/interviewee) weekly.

✅ Paid / Structured Platforms:

1. **Interviewing.io**
    - Anonymous technical interviews with experienced engineers.
    - Focus is more on SWE/SRE but works well for DevOps too.
2. **Tryexponent.com**
    - Has dedicated DevOps interview prep.
    - Includes mock interviews, scenario questions, and coaching.
3. **Scaler / Preplaced / Springboard**
    - Indian ed-tech platforms that offer mentorship and mock interviews for DevOps roles.

Prepared by: Harishankar Dubey
Purpose: Internship preparation for aspiring DevOps candidates