Manual Tester to DevOps

Complete Career Transition Guide

Mid-Level Professional (3-6 Years Experience)

Includes: LinkedIn & Naukri Job Tips | Resume Guide | Study Notes | Interview Q&A | Mock Interview

SECTION 1: LinkedIn & Naukri Job Search Strategy

LinkedIn Search Tips

Use these keywords in your LinkedIn job search to find DevOps roles:

Job Title	Why It Suits You
DevOps Engineer	Most common title - start here
CI/CD Engineer	Focused on pipelines and automation
Build & Release Engineer	Great for testers moving to DevOps
Platform Engineer	Modern cloud-native DevOps role
SRE - Site Reliability Engineer	Senior-level, good target for 3-6yr exp
DevOps Automation Engineer	Leverages your testing automation background
Infrastructure Engineer	Cloud + infra work
Cloud DevOps Engineer	AWS/Azure/GCP focused

LinkedIn Profile Optimization

- Headline: Use 'DevOps Engineer | CI/CD | Docker | Jenkins | Ex-Manual Tester | QA Automation' - this shows transition and skills together
- About Section: Write 3-4 lines about your QA background + DevOps learning journey. Mention tools you know: Selenium, Jenkins, Git, Docker, Linux
- Add DevOps tools to Skills section: Docker, Kubernetes, Jenkins, Git, Linux, Ansible, Terraform, AWS, Python/Shell scripting
- Certifications: Add AWS Cloud Practitioner or DevOps certifications (even if in

progress - mark as 'Learning')

- Follow: HashiCorp, Docker, CNCF, Atlassian - this makes you visible in DevOps circles
- Post 1-2 times/week: Share what you learned, lab screenshots, small projects - it dramatically increases recruiter visibility

Naukri Search Tips

Naukri.com specific strategies for Indian job market:

Strategy	Action
Profile Score	Get it above 80% - Naukri pushes high-score profiles to recruiters
Resume Update	Update every 7-10 days even without changes - keeps your profile fresh
Key Skills Tags	Add: DevOps, Docker, Jenkins, CI/CD, Linux, Shell Script, Git, Kubernetes
Job Alert	Set alerts for: 'DevOps 3 years', 'CI/CD Engineer', 'Build Engineer'
Premium	Naukri FastForward helps your profile appear on top - worth 1 month investment
Expected CTC	Set 20-30% higher than current - you're transitioning to a high-demand role
Location	Add 'Open to Relocation' to increase matches - DevOps jobs are pan-India/remote

Top Companies Hiring DevOps (India) - Search These on Naukri/LinkedIn

- Product companies: Flipkart, Swiggy, Zomato, PhonePe, Razorpay, CRED, Zepto
- IT Services: Infosys, TCS, Wipro, Cognizant (large scale, stable roles)
- MNCs: Amazon/AWS, Microsoft, Google, IBM, Cisco (premium roles, great learning)
- Startups: Y-Combinator India startups, funded Series A/B companies on AngelList
- Remote-first: Search 'DevOps Remote India' on LinkedIn for global salaries

SECTION 2: Resume & LinkedIn Profile Tips

How to Position Your QA Experience as DevOps Value

Your Manual Testing background is NOT a disadvantage - smart DevOps hiring managers VALUE it. Here's how to frame it:

Your QA Experience	DevOps Framing for Resume
Wrote and executed test cases	Experience with SDLC processes, quality gates, release validation
Worked with developers & release teams	Cross-functional DevOps collaboration mindset
Used JIRA for bug tracking	Experience with DevOps ticketing and workflow tools
Regression testing before releases	Exposure to release pipelines and deployment validation
API testing with Postman	Direct DevOps-relevant skill - service health monitoring
Performance/load testing	SRE mindset - availability and reliability testing experience
Worked with CI/CD pipelines	Direct DevOps experience - highlight this strongly

Resume Structure for Career Transition

Summary (Top of Resume - Most Important)

Write this type of summary:

> *"QA Engineer with 4+ years of experience in manual and API testing, transitioning to DevOps with hands-on expertise in Jenkins, Docker, Git, and Linux. Built CI/CD pipelines for personal projects, strong understanding of SDLC and release processes. AWS Certified Cloud Practitioner."*

Skills Section - Include All of These

- DevOps Tools: Jenkins, GitHub Actions, GitLab CI, CircleCI
- Containers: Docker, Docker Compose, Kubernetes (basics)
- Cloud: AWS (EC2, S3, IAM, EKS), Azure basics
- IaC: Terraform (basics), Ansible
- Scripting: Shell/Bash, Python (for automation)
- Monitoring: Prometheus, Grafana, ELK Stack
- Version Control: Git, GitHub, GitLab, Bitbucket
- Testing Background: Selenium, Postman, JIRA, TestNG - list but deprioritize

Projects Section (Critical for Career Changers)

Add 2-3 personal/side projects. These show DevOps initiative:

- Project 1: Set up a CI/CD pipeline on GitHub Actions that builds, tests, and

deploys a simple web app to AWS EC2

- Project 2: Containerized an application using Docker and deployed on Kubernetes (minikube locally)
- Project 3: Set up monitoring using Prometheus + Grafana for a sample application

Host all projects on GitHub. Add links to your resume and LinkedIn!

SECTION 3: DevOps Study Notes & Roadmap

Phase-wise Learning Plan (3 Months)

Month 1: Core Foundations

Topic	What to Learn
Linux & Shell Scripting	Learn: file system, permissions, processes, cron jobs, bash scripting
Git & Version Control	Learn: branching, merging, rebasing, pull requests, git workflows
Networking Basics	Learn: TCP/IP, DNS, HTTP/S, load balancers, firewalls, ports
CI/CD Concepts	Understand: pipeline stages, build → test → deploy, artifacts, triggers

Month 2: DevOps Tools

Tool	Key Concepts
Docker	Images, containers, Dockerfile, docker-compose, registry, networking
Jenkins / GitHub Actions	Pipeline as code, Jenkinsfile, stages, agents, webhooks
Kubernetes Basics	Pods, deployments, services, namespaces, kubectl commands
Terraform (Basics)	Provider, resource, plan, apply, state management, modules
Ansible (Basics)	Playbooks, roles, inventory, ad-hoc commands, vault

Month 3: Cloud & Monitoring

Topic	Focus Areas
AWS Core Services	EC2, S3, IAM, VPC, RDS, EKS, CloudWatch, Route53
Monitoring Stack	Prometheus metrics, Grafana dashboards, alerting rules
ELK Stack	Elasticsearch, Logstash, Kibana for log management

Security Basics	DevSecOps: SAST/DAST, secrets management, least privilege IAM

Key Concepts Cheat Sheet

CI/CD Pipeline Stages

- Source: Code commit triggers pipeline (via webhook or polling)
- Build: Compile code, run unit tests, create artifact (JAR/Docker image)
- Test: Integration tests, security scans, code quality checks (SonarQube)
- Staging Deploy: Deploy to staging/UAT environment
- Acceptance Test: Run smoke tests, automated E2E tests
- Production Deploy: Blue-green, canary, or rolling deployment strategy

Docker Key Commands

```
docker build -t myapp:v1 .  |  docker run -d -p 8080:80 myapp:v1  |
docker ps  |  docker logs <id>  |  docker exec -it <id> bash  |
docker-compose up -d
```

Git Workflow

- GitFlow: main → develop → feature/fix branches → PR/MR → code review → merge
- Trunk-Based: Short-lived feature branches, merge to main daily, feature flags for incomplete work
- Key commands: git clone, git checkout -b, git add, git commit, git push, git pull, git rebase, git merge

SECTION 4: Interview Questions & Answers

Category 1: Background & Transition Questions

Q: Why are you moving from Manual Testing to DevOps?

A: During my QA career, I worked closely with CI/CD pipelines, deployments, and release processes. I realized my passion lies in automating and streamlining those processes. I started learning Docker, Jenkins, and Linux, built my own pipelines in

personal projects, and got certified in AWS. DevOps is a natural progression that combines my quality mindset with infrastructure automation.

Example: *In my last role, I worked with the DevOps team during release cycles. I started automating test execution in Jenkins myself, which sparked my interest in the entire pipeline.*

Q: How does your QA background help in DevOps?

A: QA gives me a quality-first mindset which is core to DevOps culture. I understand SDLC deeply, I know what can break during deployments, I have experience validating releases, and I'm comfortable with tools like JIRA, Git, and API testing. This makes me better at building reliable pipelines with proper test gates and monitoring.

Category 2: CI/CD & Jenkins

Q: What is CI/CD? Explain with a real example.

A: CI (Continuous Integration) means developers frequently commit code, and each commit triggers automated build and test. CD (Continuous Delivery/Deployment) means the tested code is automatically deployed to staging or production. Together, they reduce manual effort and catch bugs early.

Example: *Developer pushes code to GitHub → Jenkins triggers → Maven builds JAR → Unit tests run → Docker image created → Pushed to ECR → Deployed to EKS cluster.*

Q: What is a Jenkinsfile? Write a basic pipeline.

A: A Jenkinsfile is a text file that defines the Jenkins pipeline as code. It lives in the repository alongside the code.

```
pipeline {
  agent any
  stages {
    stage('Build') { steps { sh 'mvn clean package' } }
    stage('Test') { steps { sh 'mvn test' } }
    stage('Docker Build') { steps { sh 'docker build -t myapp:latest .' } }
    stage('Deploy') { steps { sh 'kubectl apply -f deployment.yaml' } }
  }
}
```

Q: What is the difference between Continuous Delivery and Continuous Deployment?

A: Continuous Delivery means the code is always in a deployable state and deployment to production requires manual approval. Continuous Deployment goes one step further - every passing change is deployed to production automatically without human intervention. Most companies use Continuous Delivery for controlled releases.

Category 3: Docker & Containers

Q: What is Docker? How is it different from a Virtual Machine?

A: Docker is a containerization platform that packages applications with their dependencies into lightweight containers. Unlike VMs that virtualize hardware and need a full OS, containers share the host OS kernel making them faster to start, use less memory, and are more portable. A VM might take minutes to boot; a Docker container starts in seconds.

Q: What is the difference between CMD and ENTRYPOINT in Dockerfile?

A: ENTRYPOINT defines the main command that always runs when the container starts - it cannot be overridden easily. CMD provides default arguments that can be overridden at runtime. Best practice: use ENTRYPOINT for the executable and CMD for default arguments.

Example: *ENTRYPOINT ['python', 'app.py'] and CMD ['--port', '8080'] - you can override the port but app.py always runs.*

Q: What is Docker Compose? When would you use it?

A: Docker Compose is a tool to define and run multi-container applications using a YAML file (docker-compose.yml). You use it when your app needs multiple services like a web app + database + cache running together. A single 'docker-compose up' starts all services with correct networking between them.

Example: *A microservices app with a React frontend, Node.js API, PostgreSQL DB, and Redis cache - all defined in one docker-compose.yml.*

Category 4: Kubernetes

Q: What is Kubernetes? What problem does it solve?

A: Kubernetes (K8s) is a container orchestration platform that automates deployment, scaling, and management of containerized applications. It solves problems like: how to run hundreds of containers across multiple servers, how to restart failed containers automatically, how to scale up when traffic increases, and how to do zero-downtime deployments.

Q: What is the difference between a Pod, Deployment, and Service?

A: A Pod is the smallest unit in Kubernetes - it runs one or more containers. A Deployment manages Pods declaratively, ensures the desired number of replicas are running, and handles rolling updates. A Service is an abstraction that provides a stable network endpoint to access Pods, since Pod IPs change. Think: Pod = single instance, Deployment = manages many instances, Service = load balancer for those instances.

Q: What is a Kubernetes Namespace?

A: A Namespace is a logical partition within a Kubernetes cluster that allows you to

isolate resources. For example, you can have 'development', 'staging', and 'production' namespaces in the same cluster with separate resource quotas, RBAC permissions, and network policies.

Category 5: Linux & Shell Scripting

Q: Write a shell script to check if a process is running and restart it if not.

A: This tests practical scripting knowledge.

```
#!/bin/bash
PROCESS='nginx'
if ! pgrep -x "$PROCESS" > /dev/null; then
  echo "$PROCESS not running. Starting..."
  systemctl start $PROCESS
  echo "$PROCESS started."
else
  echo "$PROCESS is running."
   fi
```

Q: What is the difference between soft link and hard link in Linux?

A: A hard link is a direct reference to the file's inode - both the original and hard link are identical entries pointing to same data. If original is deleted, data persists. A soft link (symbolic link) is a pointer to the file's path - it breaks if the original file is deleted. Use 'ln target link' for hard link and 'ln -s target link' for soft link.

Category 6: AWS / Cloud

Q: What AWS services are commonly used in DevOps pipelines?

A: CodePipeline and CodeBuild for CI/CD, ECR for Docker image registry, EKS or ECS for container orchestration, EC2 for compute, S3 for artifacts and static hosting, IAM for access control, CloudWatch for monitoring and logs, Route53 for DNS, and Secrets Manager for credentials.

Q: What is the difference between IAM Role and IAM Policy?

A: An IAM Policy is a document that defines permissions (what actions are allowed/denied on which resources). An IAM Role is an identity that can be assumed by services or users - roles have policies attached to them. For example, an EC2 instance assumes a Role that has a Policy allowing it to write to S3.

Category 7: Monitoring & Logging

Q: How do you monitor an application in production? What tools do you use?

A: Metrics monitoring with Prometheus (collects numerical metrics) + Grafana (visualizes dashboards and alerts). Log aggregation with ELK Stack (Elasticsearch stores logs, Logstash processes them, Kibana provides search/visualization) or CloudWatch Logs on AWS. Application Performance Monitoring (APM) with Datadog or New Relic for tracing. Set up alerts for: high CPU/memory, error rate spike, latency increase, pod crashes.

Category 8: Behavioral Questions (HR Round)

Q: Tell me about a time you automated something to save time.

A: Use the STAR method: Situation, Task, Action, Result.

Example: *S: Our regression test suite took 3 days manually before every release. T: I was asked to reduce this. A: I set up Selenium Grid integrated with Jenkins triggered on every commit. R: Test execution reduced from 3 days to 4 hours, enabling 2x faster releases.*

Q: Describe a production incident you helped resolve.

A: STAR method: Describe the incident, your role, steps taken (check logs, metrics, rollback, fix), and outcome. Even if you were in QA during this, you can describe how you helped the DevOps team validate the fix.

SECTION 5: Mock Interview Practice

How to Use This Section

Treat these as real interview questions. Write your answer before reading guidance. Record yourself speaking answers out loud - this is the fastest way to get interview-ready.

Round 1: Technical Screening (30 minutes)

Practice These Questions - Try Answering First

1. Explain the difference between Blue-Green and Canary deployment strategies.
2. What is Infrastructure as Code? Why is Terraform preferred over shell scripts?
3. How would you set up a CI/CD pipeline for a microservices app from scratch?
4. What is Helm in Kubernetes? Why is it used?
5. Explain how you would handle secrets management in a Kubernetes deployment.

6. What is the difference between horizontal and vertical scaling?
7. How does a load balancer work? What is the difference between Layer 4 and Layer 7?

Round 2: Practical/Coding Round

Common Tasks Asked in DevOps Technical Rounds

- Write a Dockerfile for a Node.js application
- Write a Jenkins pipeline that builds, tests, and pushes Docker image to ECR
- Write a Terraform script to create an EC2 instance with a security group
- Debug a failing Kubernetes deployment (they show you kubectl get pods output)
- Write a bash script to monitor disk usage and send alert if above 80%

Round 3: System Design (Senior/Mid-level)

Sample Design Question

Design a CI/CD pipeline for a 3-tier application (Frontend + API + Database) that needs to support 5 developer teams committing code simultaneously.

Framework to Answer System Design

- Step 1 - Clarify requirements: How many deployments per day? Environments needed? Rollback requirement?
- Step 2 - Draw pipeline flow: Dev commit → PR → CI (build+test) → staging deploy → QA sign-off → prod deploy
- Step 3 - Tool selection: GitHub → GitHub Actions → Docker → ECR → EKS → Helm → ArgoCD (GitOps)
- Step 4 - Handle failure scenarios: automatic rollback, alerting, feature flags for risky changes
- Step 5 - Monitoring: CloudWatch + Prometheus + Grafana + PagerDuty for on-call alerts

Questions to Ask the Interviewer

Always ask 2-3 questions at the end - it shows genuine interest:

- What does the current deployment pipeline look like, and what are the main pain points the team is trying to solve?
- How is the DevOps team structured - is it a central platform team or embedded

in product teams?

- What does the on-call rotation look like for this role?
- What are the main tech stack and tools your team uses day-to-day?

SECTION 6: Salary & Negotiation

Expected Salary Range for DevOps (India, 2024-25)

Role / Profile	Salary Range
Junior DevOps (0-2 yrs)	6-12 LPA
Mid-level DevOps (3-6 yrs) - Your Target	12-25 LPA
Senior DevOps (6+ yrs)	25-45 LPA
DevOps with AWS certifications (+ 20%)	15-30 LPA
DevOps at MNC / Product companies	20-40 LPA
Remote DevOps (for global companies)	$40-80k USD
Manual Testing (your current benchmark)	5-12 LPA typically

Negotiation Tips

- Never accept the first offer - always counter at 15-20% above their offer
- Lead with: 'I'm excited about this role. Based on my research and skills, I was expecting X. Is there flexibility?'
- Certifications justify higher salary - get AWS or CKA certification before negotiating
- Consider total comp: base + bonus + ESOPs + learning budget, not just base salary
- Use competing offers - if you have multiple interviews, let them create urgency

SECTION 7: Quick Reference Card

30-Second Explanation of Key DevOps Terms

Term	Quick Explanation
GitOps	Using Git as the single source of truth for infrastructure and deployments
ArgoCD	GitOps tool for Kubernetes - watches Git repo, syncs to cluster
Helm	Package manager for Kubernetes - bundles K8s YAML into reusable 'charts'
Service Mesh	Istio/Linkerd - handles microservice communication, auth, observability
SRE	Site Reliability Engineering - applies software eng to operations, defines SLOs/SLAs
Shift Left Security	Integrate security testing early in pipeline, not at the end
Feature Flags	Toggle features on/off without code deployment
Chaos Engineering	Deliberately break things in prod (Netflix Chaos Monkey) to find weaknesses

Certifications Roadmap

- Start: AWS Cloud Practitioner (CLF-C02) - 2-3 weeks prep, proves cloud basics
- Next: AWS Solutions Architect Associate (SAA-C03) - most recognized cert by Indian recruiters
- DevOps Specific: Certified Kubernetes Administrator (CKA) - premium, shows serious commitment
- Also valuable: HashiCorp Terraform Associate, Jenkins Certified Engineer

Good luck with your DevOps journey! Your QA background is an asset - own it.

Build projects. Get certified. Practice out loud. You've got this!