Below is the **COMPLETE, SERIAL, END-TO-END DevOps Pipeline — from first code commit to production monitoring** — explained **step by step**, **in order**, **with real commands**, **file locations**, **what changes**, **where things are stored**, and **why each step is critical**.

This is **100% interview-ready** — perfect for **freshers & intermediate** roles.
You can **say this in 5–7 minutes** or **draw it on a whiteboard**.

FULL DEVOPS PIPELINE: SERIAL FLOW (2025)

> ***Tools Used***:
> *`GitLab → Jenkins → Terraform → Maven → Docker → Helm → Kubernetes (EKS) → AWS CloudWatch`*

> ***App Example***: *Java Spring Boot API (`/hello`)*
> ***Infra***: *AWS (EKS cluster, S3, ECR, IAM)*

STAGE 0: INITIAL SETUP (One-Time)

Goal: Prepare environment before first run

	Step		Command / Action		What Changes	Where Stored	Impor
1	Launch **3 EC2 instances** (Amazon Linux 2)	3 VMs created	AWS Console → EC2	Jenkins, Ansible, K8s control			
2	Install tools on **Jenkins EC2**	`sudo yum install java-17 maven git jenkins docker`	`/usr/bin/`	Build + orchestrate			
3	Start Jenkins	`sudo systemctl start jenkins`	`http://ip:8080`	Pipeline engine			
4	Create **S3 bucket** for Terraform state	`aws s3 mb s3://myapp-state-2025`	AWS S3	Prevent state loss			
5	Create **ECR repo**	`aws ecr create-repository --repository-name myapp`	AWS ECR	Store Docker images			
6	Create **GitLab repo**	`myapp.git`	`gitlab.com/user/myapp`	Source of truth			

STAGE 1: CODE → BUILD (Developer + GitLab + Jenkins + Maven)

#	Step		
1	**Developer writes code**	`src/main/java/Hello.java` , `pom.xml` , `Dockerfile`	New files
2	**Commit & Push**	```bash	
git add .			
git commit -m "Add hello endpoint"			
git push origin main			
``	Commit object created (SHA: `a1b2c3`)	`.git/objects/a1/b2c3...` → GitLab server	Triggers pipeline
3	**GitLab sends webhook**	POST → `http://jenkins-ip:8080/gitlab-webhook/`	Jenkins job triggered
4	**Jenkins: Checkout**	```groovy	
git url: 'https://gitlab.com/user/myapp.git'			
``	Code cloned	`/var/lib/jenkins/workspace/job/`	Fresh code
5	**Maven: Build**	`mvn clean package`	

- `target/classes/` → `.class` files
- `target/myapp.jar` → final JAR | `/var/lib/jenkins/workspace/job/target/` | Artifact ready | | 6 | **Upload artifact to S3** | `aws s3 cp target/myapp.jar s3://artifacts-bucket/v1.jar` | JAR stored | `s3://artifacts-bucket/v1.jar` | Backup + audit |

> ***Pipeline Status**: **Build Success***
> ***Output**: `myapp.jar` in S3*

STAGE 2: INFRA → PROVISION (Terraform + AWS)

#	Step	Command	What Changes	Where Stored	Why Important
7	**Jenkins: Terraform Init**	```bash			
cd infra/					
terraform init -backend-config="bucket=myapp-state-2025"					
``	Downloads AWS provider	`.terraform/`	Setup		
8	**Terraform Plan**	`terraform plan`	Shows **what will be created**	Console output	Review before apply
9	**Terraform Apply**	`terraform apply -auto-approve`			

- EKS cluster created
- Node group (2 t3.medium)
- ECR repo
- IAM roles | AWS Cloud | Infra as Code | | 10 | **Terraform State** | `terraform.tfstate` | **Lock + current state** | `s3://myapp-state-2025/terraform.tfstate` | Drift detection |

> ***File**: `infra/main.tf` (in Git)*
> ***Output**: EKS cluster ready, `kubeconfig` generated*

STAGE 3: PACKAGE → IMAGE (Docker + GitLab Registry)

#	Step		
11	**Download JAR**	`aws s3 cp s3://artifacts-bucket/v1.jar .`	JAR in workspace
12	**Docker Build**	```dockerfile FROM openjdk:17-slim COPY myapp.jar /app.jar CMD ["java", "-jar", "/app.jar"] ```	Image layers created
		`docker build -t myapp:v1.0.$BUILD_NUMBER .`	`/var/lib/docker/overlay2/`	Portable app
13	**Tag & Push**	```bash docker tag myapp:v1.0.123 registry.gitlab.com/user/myapp:v1.0.123 docker push registry.gitlab.com/user/myapp:v1.0.123 ```		

```
language-|                                                    Collapse  Wrap  Copy
```

```
> **Output**: `registry.gitlab.com/user/myapp:v1.0.123`

---

## STAGE 4: DEPLOY → K8s (Helm + EKS)

| # | Step | Command | What Changes | Where Stored | Why Important |
|---|---|---|---|---|---|
| 14 | **Update Helm values** | `helm/values.yaml` → `image.tag: v1.0.123` | | | |
| 15 | **Helm Upgrade** | ```bash
helm upgrade --install myapp helm/chart/ \
  --set image.tag=v1.0.123 \
```

  --namespace prod
``` |
- Deployment updated
- Pods rolling restart | etcd (K8s) | Zero downtime |
| 16 | **K8s Objects** |
- `Deployment/myapp`
- `Service/myapp` (LoadBalancer)
- `Ingress` (optional) | `kubectl get all -n prod` | Live app |

> **File**: `helm/chart/templates/deployment.yaml`
> **Output**: App live at `http://a123.elb.amazonaws.com/hello`

STAGE 5: TEST → VERIFY (Jenkins + K8s)

#	Step	Command	What Changes	Where Stored	Why Important
| 17 | **Smoke Test** | ```bash
curl -f http://$(kubectl get svc myapp -n prod -o jsonpath='{.status.loadBala
``` | Returns `Hello!` | Console | Health check |
| 18 | **Rollback if fail** | `helm rollback myapp 1` | Reverts to v1.0.122 |

> **Output**: **Deploy Success**

STAGE 6: MONITOR → OBSERVE (CloudWatch + K8s)

#	Step	Action	What Changes	Where Stored	Why Important
19	**CloudWatch Agent**	Installed on EKS nodes	Logs → CloudWatch	`/
20	**HPA (Auto-Scale)**	`kubectl autoscale deployment myapp --cpu-perce
21	**Alarm**	CPU > 80% → SNS → Slack	Alert sent	CloudWatch	Proac

---

FINAL SERIAL FLOW (1–21)

```mermaid
flowchart TD
    A[1. Code] --> B[2. Git Push]
    B --> C[3. Jenkins Trigger]
```

```
C --> D[4. Maven Build]
D --> E[5. S3 Artifact]
E --> F[6. Terraform Apply]
F --> G[7. EKS Ready]
G --> H[8. Docker Build]
H --> I[9. Push to GitLab Registry]
I --> J[10. Helm Deploy]
J --> K[11. K8s Pods Running]
K --> L[12. Smoke Test]
L --> M[13. CloudWatch Logs]
M --> N[14. HPA Scales]
```

FILE LOCATIONS SUMMARY (Where Everything Lives)

File / Object	Location	Purpose
`pom.xml`	GitLab repo root	Maven config
`Dockerfile`	GitLab repo	Image build
`myapp.jar`	`s3://artifacts-bucket/v1.jar`	Backup
`terraform.tfstate`	`s3://myapp-state-2025/`	Infra state
Docker Image	`registry.gitlab.com/user/myapp:v1.0.123`	Deployable
Helm Chart	`helm/chart/` in Git	K8s templates
K8s Objects	etcd (EKS)	Live state
Logs	CloudWatch Logs	Monitoring

WHY EACH STEP IS CRITICAL

Step	If Missing?
Git Push	No trigger
Maven	No artifact
Terraform	Manual infra → errors
Docker	Not portable
Helm	Manual `kubectl` → slow
Test	Broken app in prod
Monitor	Blind to failures

INTERVIEW DELIVERY: 5-Minute Story

> *"Let me walk you through a real pipeline I built — from code to production."*

1. **"Developer pushes to GitLab → Jenkins starts."**
2. **"Jenkins runs Terraform → creates EKS cluster from code."**
3. **"Maven builds JAR → Docker packages it → pushes to GitLab Registry."**
4. **"Helm deploys to EKS → 3 pods → zero downtime."**
5. **"CloudWatch watches → auto-scales → alerts on Slack."**

> *"Everything is in Git. I can swap Jenkins with GitLab CI, or EKS with GKE — in 5 mins."*

Interviewer Follow-Up Questions & Your Answers

Question	Answer
Where is the image?	`registry.gitlab.com/user/myapp:v1.0.123`
How do you rollback?	`helm rollback myapp 1`
What if Terraform fails?	`terraform plan` shows diff → fix code
How is state safe?	`tfstate` in S3 + DynamoDB lock
Can you show one file?	`helm upgrade --install myapp helm/chart/ --set image.tag=$BUILD_NUMBER`

PRINTABLE CHEAT SHEET (Copy-Paste)

text Collapse Wrap Copy

```
STAGE 1: CODE → GitLab → git push → Jenkins webhook
STAGE 2: BUILD → Maven → target/myapp.jar → S3
STAGE 3: INFRA → Terraform → EKS + ECR → tfstate in S3
STAGE 4: IMAGE → Docker build → GitLab Registry
STAGE 5: DEPLOY → Helm upgrade → K8s pods
STAGE 6: MONITOR → CloudWatch + HPA
```

You now have the FULL STORY — serial, detailed, file-level, interview-proof.

Want:

- **PDF version**? → Say: **“PDF”**
- **GitLab + Terraform + Helm repo template**? → Say: **“REPO”**
- **Live demo video script**? → Say: **“DEMO”**

You are ready to explain this in any interview — and get hired.

Go crush it.