Kubernetes: Pod Running but App Not Opening

- Problem: Pod was running but the application was not loading.
- I checked service and endpoints, and endpoints were 0.
- Service selector labels did not match pod labels.
- I updated labels and restarted deployment.
- Application started working immediately.

Kubernetes: CrashLoopBackOff After Config Change

- Pods kept restarting after deployment.
- kubectl logs showed missing environment variable.
- ConfigMap key changed earlier and deployment still used old key.
- I updated mapping and restarted pods.
- Pods stabilized and service recovered.

Docker: Works Locally but Fails in Container

- App ran fine locally but container crashed immediately.
- Container logs showed missing dependency.
- Dependency was installed locally but not in Dockerfile.
- Added dependency in Dockerfile and rebuilt image.
- Container started and app worked.

Jenkins: Pipeline Failed After Commit

- Pipeline started failing right after pushing code.
- Console output showed Jenkinsfile syntax error.
- I corrected syntax and reran pipeline.
- Pipeline executed successfully.

Jenkins: Build Stuck Due to Offline Agent

- Pipeline was stuck waiting for an agent.
- Agent node disk was full due to old workspaces.
- I cleaned workspace and restarted agent service.
- Agent reconnected and pipeline resumed.

AWS EC2: High CPU During Peak Traffic

- Application became slow during peak usage.
- CloudWatch showed CPU spike.
- I enabled AutoScaling and optimized heavy DB queries.
- Performance improved and app ran smoothly.

AWS S3: Access Denied Even with IAM Role

- Application couldn't fetch files from S3.
- IAM role was correct, but bucket policy was denying access.
- I added role ARN to bucket allow policy.
- Access was restored.

AWS ALB: Unhealthy Targets

- Service was not reachable because targets were unhealthy.
- Health check endpoint was returning unauthorized.
- I allowed 200 OK on health endpoint.
- Targets became healthy and app worked.

Linux: Server Out of Disk Space

- Server stopped responding due to full disk.
- Logs in /var/log were consuming large space.
- I enabled logrotate and cleared old logs.
- Server performance returned to normal.

Linux: High Memory Usage

- System was running slow due to memory leak.
- top showed one process consuming excessive memory.
- I restarted service temporarily and later fixed code leak.
- System became stable.

Git: Merge Conflicts During Release

- Conflicts happened frequently when merging release branch.
- I implemented feature branch workflow with PR reviews.
- Conflicts reduced significantly.

Terraform: Plan Showing Resource Destroy

- terraform plan showed resource destroy action.
- Someone updated resource manually in console.
- I imported resource back into state and applied safely.
- Environment remained stable.

Terraform: State Lock Issue

- terraform apply was blocked by locked state.
- I verified no active job and used terraform force-unlock.
- Pipeline resumed normally.

Terraform: Multi Environment Setup

- I used modules + separate tfvars for each environment.
- This helped maintain separation between dev, stage, prod.
- Deployment became safer and more structured.

Microservices: One Service Timing Out

- API calls were timing out between microservices.
- I added retry, timeout, and backoff logic.
- System became more stable under load.

CI/CD: Deployment Successful but App Down

- Deployment succeeded but application not responding.
- I rolled back quickly and checked environment values.
- I fixed the variable and redeployed successfully.

Kubernetes: Readiness Probe Issue

- Pods were marked ready before app fully started.
- I adjusted readiness probe delay and endpoint path.
- Service stop/start during deploy reduced.

Docker: Large Image Causing Delays

- Docker image was very large causing slow deploys.
- I implemented multi-stage builds and used .dockerignore.
- Image size reduced and deployment speed improved.

AWS RDS: Slow Database Queries

- Application response time was high.
- DB performance insights showed slow queries.
- I added indexing and caching layer.
- Application performance improved.

Incident Handling Mindset

- When service breaks: First restore service, then do analysis.
- Do root cause analysis after recovery, not during outage.
- Share learnings to avoid repeating the issue.