# DevOps Engineer Interview Preparation - Fulfillment IQ

## 1. Project Story (Aligned with FIQ JD)

\*\*Question:\*\* Can you describe one of your recent DevOps projects and your role in it?

\*\*Answer:\*\*   
In my previous role, I was part of a team responsible for automating the deployment and infrastructure of a customer-facing logistics web application.   
Initially, deployments were mostly manual, which caused inconsistencies, long release cycles, and downtime during updates.  
  
\*\*Challenges Faced:\*\*  
- Manual provisioning of cloud resources led to environment drift.  
- Frequent deployment delays due to manual intervention.  
- Difficulty managing multiple environments (dev, QA, prod).  
- Limited rollback options during deployment failures.  
  
\*\*What I Did:\*\*  
- Designed and implemented a fully automated CI/CD pipeline using Jenkins integrated with GitHub.   
- The pipeline handled build, test, Docker image creation, and deployment stages.  
- Integrated automated code-quality checks and unit tests before deployment.  
- Introduced Docker for consistent builds and Kubernetes for container orchestration.  
- Implemented Infrastructure as Code using Terraform for provisioning AWS infrastructure — including EC2, VPC, S3, and EKS clusters.  
- Used Ansible for configuration management and system updates post-provisioning.  
- Deployed Blue-Green and Rolling Update strategies in Kubernetes to minimize downtime.  
- Integrated notifications with Slack and email alerts for build and deployment statuses.  
- Implemented cost optimization using auto-scaling, S3 lifecycle rules, and tagging unused resources for cleanup.  
- Collaborated closely with developers and QA to ensure smooth versioned deployments and consistent environments.  
  
\*\*Results:\*\*  
- Reduced deployment time from 45 minutes to around 8 minutes.  
- Improved system stability and eliminated manual setup errors.  
- Achieved 99.9% uptime for production environments.  
- Lowered cloud spending by roughly 18% through automation and optimization.  
  
\*\*Key Tools & Technologies:\*\*  
AWS (EC2, S3, EKS, CloudWatch), Jenkins, Terraform, Ansible, Docker, Kubernetes, GitHub, Python/Bash scripting.  
  
\*\*Closing Line:\*\*  
This project gave me hands-on experience in automating end-to-end delivery pipelines, managing infrastructure through code, and ensuring scalability and reliability — which aligns closely with how FIQ manages cloud-based logistics applications.

## 2. Troubleshooting Scenario (Production Issue)

\*\*Question:\*\* Can you describe a real-life production issue you resolved and how you approached it?

\*\*Answer:\*\*   
During one of our production deployments, we noticed that the newly deployed application pods were going into a CrashLoopBackOff state in Kubernetes.   
This caused partial downtime for one of the services, which impacted the order-tracking functionality for end-users.  
  
\*\*Task:\*\*  
As part of the DevOps team, my responsibility was to identify the root cause, restore service as quickly as possible, and ensure it didn’t recur in future deployments.  
  
\*\*Action:\*\*  
- Checked deployment logs in Jenkins and Kubernetes to identify which version and configuration were deployed.  
- Used `kubectl logs` and `describe pod` to analyze the issue.  
- Discovered a wrong environment variable (missing API endpoint) causing the service to fail during startup.  
- Rolled back the deployment using the previous stable image tag, which restored service immediately.  
- Implemented an automated configuration validation step in the CI/CD pipeline to ensure all environment variables were validated before deployment.  
- Worked with the development team to store configuration values securely in AWS Secrets Manager instead of manually defining them in YAML files.  
  
\*\*Result:\*\*  
- Service was restored within 10 minutes, minimizing downtime.  
- Prevented similar configuration issues in future releases.  
- Improved deployment reliability by adding pre-validation checks to the pipeline.  
  
\*\*Key Tools Used:\*\*  
Jenkins, Docker, Kubernetes, AWS Secrets Manager, Bash scripting, GitHub, Terraform (for infra consistency).  
  
\*\*Closing Line:\*\*  
This incident taught me how crucial validation and rollback automation are in a DevOps environment, especially when multiple teams are pushing code frequently.