Deployment of Online Ticketing and Event Management Applications and Infrastructure

A DevOps Implementation for Scalable Movie Ticketing Platform

Domain: E-Commerce and Entertainment

Project Overview:
BookMyShow is a popular online movie ticketing platform that allows users to browse movies, book tickets, and manage their bookings. The company is transitioning from a monolithic architecture to a microservices-based architecture to improve scalability, deployment efficiency, and infrastructure management.

To achieve this, BookMyShow has decided to adopt DevOps practices with CI/CD pipelines, containerization, and automated infrastructure provisioning. The project will use AWS as the primary cloud provider for hosting servers, databases (if any), and deploying applications.

GitHub URL: https://github.com/staragile2016/Book-My-Show.git

Business Goals:
- Deliver frequent and reliable updates to production.
- Accelerate software delivery speed while maintaining high quality.
- Reduce manual intervention in deployments and testing.
- Improve scalability and monitoring of the application.

Current Challenges:
- Complex Builds: Managing dependencies and incremental builds is difficult.
- Manual Testing: Testing multiple components manually is time-consuming.
- Infrastructure Setup: Manual server provisioning and configuration delays deployments.

Monitoring Challenges:
- Lack of real-time monitoring leads to delayed issue detection.
- CI/CD Pipeline Implementation

The following tools will be used to automate the deployment process:
Tools &amp; Technologies
- Git – Version control for tracking code changes.
- Jenkins – CI/CD pipeline automation (build, test, deploy).
- Docker – Containerization of the application.
- Kubernetes (K8s) – Orchestration for scalable deployments.
- Ansible – Configuration management for infrastructure automation.
- AWS – Cloud infrastructure (EC2, EKS for Kubernetes).
- Prometheus &amp; Grafana – Monitoring and alerting.

Pipeline Workflow:

- GitHub Webhook Trigger
- When a developer pushes code to the master branch, Jenkins automatically triggers the pipeline.
- Build &amp; Test Phase
- Jenkins checks out the code, creates docker images, pushes docker images, creates containers, deploys to EKS cluster
- TestNG generates an HTML test report.
- Dockerization: The application is packaged into a Docker image and pushed to AWS ECR (Elastic Container Registry).
- Kubernetes Deployment (Dev / Stage Environment)
- Ansible deploys the Docker container to a Kubernetes cluster (AWS EKS).
- Monitoring &amp; Validation
- Prometheus collects metrics (CPU, memory, API latency).
- Grafana visualizes performance dashboards.
- If the deployment is stable, it proceeds to production.

Production Deployment:

- After validation in staging, the same Docker image is deployed to the production Kubernetes cluster.

Expected Outcome:

✅ Automated CI/CD Pipeline – No manual intervention required for deployments.
✅ Scalable Microservices – Kubernetes ensures high availability.
✅ Real-time Monitoring – Prometheus &amp; Grafana provide observability.
✅ Faster Releases – Frequent, reliable deployments with rollback capability.