DEVOPS AND INFRASTRUCTURE AS CODE

By Babatunde Victor

21/10MSS005

INTRODUCTION TO DEVOPS

- What is DevOps? A practice combining development (Dev) and operations (Ops) to accelerate the software delivery lifecycle and ensure continuous, high-quality releases.
- Key Principles: Focuses on collaboration, automation, lean flow, feedback, and knowledge sharing across teams.

OPEN-SOURCE DEVOPS TOOLS AND PRACTICES

1.

- Version Control: Git is essential for tracking code changes and enabling collaboration. Platforms like GitHub/GitLab host repositories and integrate CI/CD.
- Build Automation: Tools like Maven/Gradle (Java) and npm/Yarn (JavaScript) automate compilation and packaging

2.

- Configuration Management: Ansible, Chef, Puppet define and manage infrastructure as code.
- Monitoring & Logging: Prometheus/Grafana for metrics, ELK Stack for centralized logging.

CONTAINERIZATION WITH DOCKER

- What is Containerization? Packaging an application and its dependencies into isolated containers for consistent execution across environments.
- Why Docker? Provides portability, isolation, efficiency, and reproducibility.
- Key Concepts: Dockerfile builds a Docker Image, from which Containers are run. Docker Hub is a registry, and Docker Compose manages multi-container apps.

ORCHESTRATION WITH KUBERNETES

- What is Kubernetes? An open-source system from Google that automates the deployment, scaling, and management of containerized applications.
- Why Kubernetes? Offers automated deployments/rollbacks, self-healing, load balancing, resource management, and scalability.

- Key Concepts: Pods are the smallest deployable units, Deployments manage Pods, Services expose applications, Namespaces organize resources, and kubectl is the command-line tool.

CONTINUOUS INTEGRATION (CI)

- What is CI? Developers frequently merge code into a central repository, triggering automated builds and tests.
- Goals: Detect errors early, reduce integration issues, maintain a stable codebase, and provide quick feedback.
- Workflow: Code commit triggers build, tests run, and feedback is provided.

CONTINUOUS DEPLOYMENT (CD)

- What is CD? Extends CI by automatically deploying code to production once tests pass.
- Continuous Delivery: Ready for deployment, but manual trigger.
- Continuous Deployment: Automated to production.
- Benefits: Faster time to market, reduced release risk, improved quality, higher productivity.

CI/CD PIPELINE STAGES

- Commit: Code push, static analysis.
- Build: Compile, create artifacts (e.g., Docker images).
- Test: Run unit, integration, end-to-end tests.
- Deploy (Staging): Deploy to staging for UAT.
- Release (Production): Automated production deployment with monitoring.

LAB OVERVIEW

- Objective: Simulate a CI/CD pipeline, containerize an app, and simulate cloud deployment.
- Steps: Simulate build, test, deploy stages; conceptual Docker demonstration; simulated cloud deployment endpoint.
- Tools (Simulated): Web interface for pipeline status and conceptual updates.

CASE STUDY: KUBERNETES - FROM GOOGLE PROJECT TO CLOUD-NATIVE FOUNDATION

- Origins: Evolved from Google's internal Borg system.
- Open-Sourcing: Open-sourced in 2014, donated to CNCF in 2015, fostering a vast community.
- Impact: Became the standard for container orchestration, enabling portability and driving microservices adoption, crucial for modern cloud-native applications.

THANK YOU