DevOps Project Implementation

CA-2

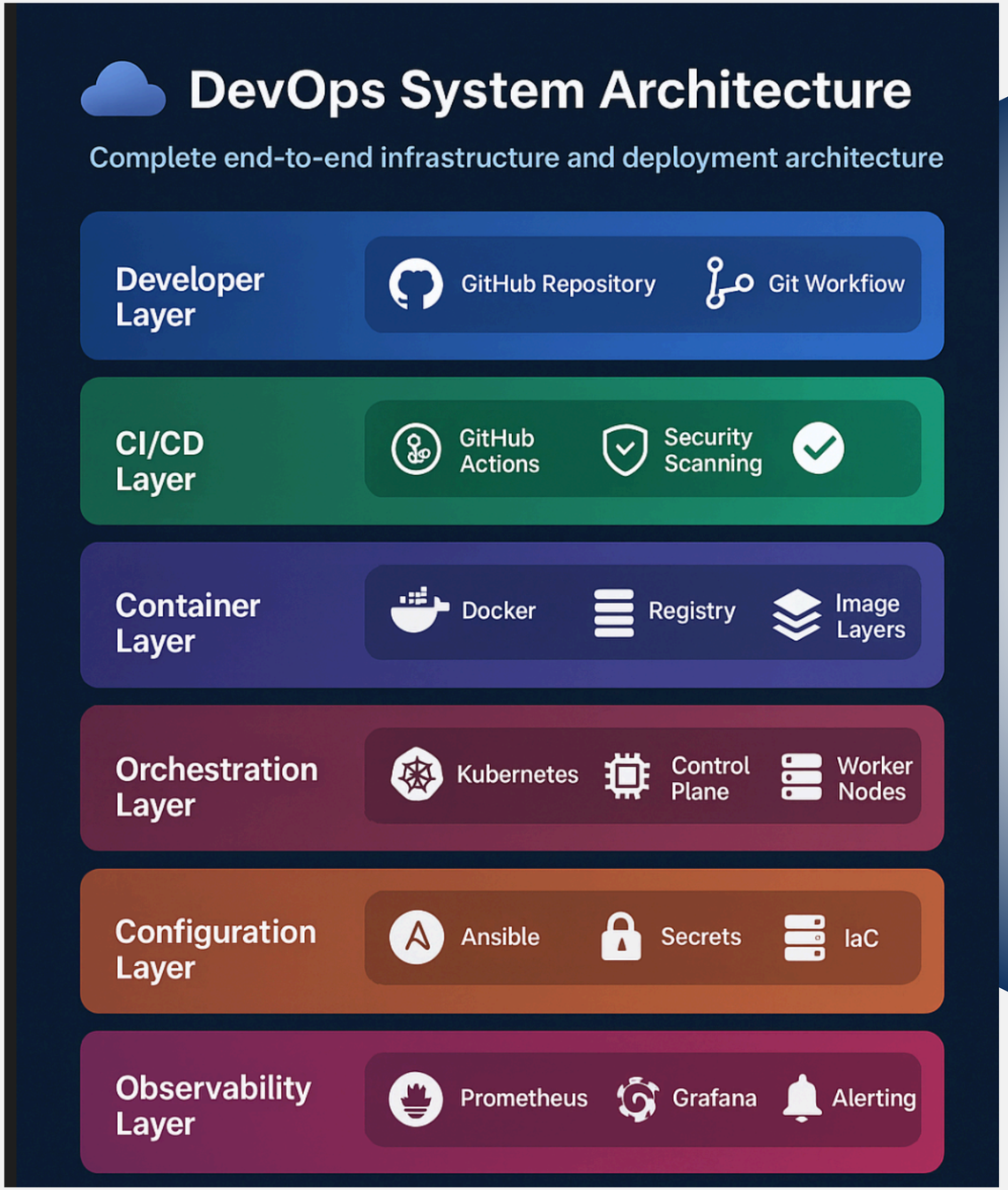
Group Members:

1. 22070122176- Samidha Manjrekar
2. 22070122177- Samruddhi Borhade
3. 22070122206- Shruti Bist
4. 22070122219- Srishti Parulekar

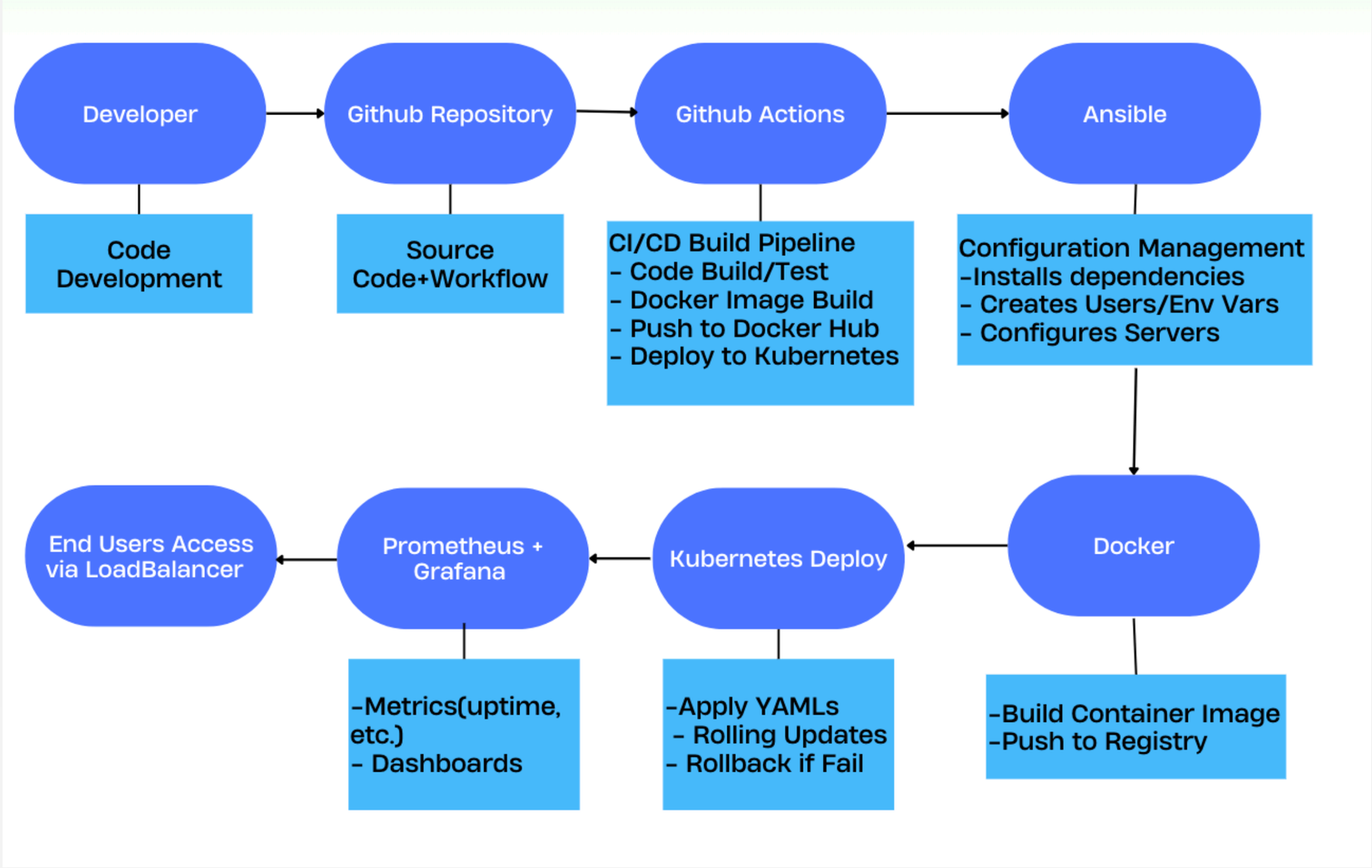
Architecture Diagram

Complete System Flow

1. Developer commits code to GitHub
2. Container image pushed to registry
3. GitHub Actions pipeline executes
4. Kubernetes pulls and deploys
5. Container image pushed to registry
6. Monitoring begins tracking metrics



Pipeline Flow Diagram



Challenges Encountered

Dependency Conflicts: The use of npm install --legacy-peer-deps for the frontend suggests that there were strict peer dependency mismatches in the project's package.json file, requiring a workaround to complete the installation.

Environment Parity: Ensuring the local development environment (configured with Ansible) perfectly matches the containerized environment (defined in Dockerfiles) can be challenging. Minor differences in package versions or system libraries can lead to "it works on my machine" issues.

Kubernetes Networking: Correctly configuring Kubernetes Service and Deployment manifests is complex. Ensuring that pods can communicate with each other and are correctly exposed outside the cluster via a NodePort requires a precise understanding of Kubernetes networking concepts.

Monitoring Configuration: Setting up the monitoring stack involves multiple steps. After deploying Grafana, it must be correctly configured to connect to the Prometheus data source using the right internal cluster DNS name which requires Prometheus to be deployed and running correctly first.

Lesson Learned

- **Automation Reduces Errors:** By automating the build and environment setup processes with GitHub Actions and Ansible, the risk of manual error is significantly reduced, leading to more reliable and repeatable outcomes.
- **Infrastructure as Code (IaC) is Powerful:** Defining infrastructure (Kubernetes manifests) and configuration (Ansible playbooks) as code is a best practice. It allows for version control, peer review, and easy replication of the entire system architecture.
- **Containers are the Standard for Portability:** Dockerizing the application ensures that it runs the same way everywhere—from a developer's laptop to a production cluster. This solves many deployment-related issues.
- **Monitoring is Essential:** Setting up tools like Grafana from the start is crucial for gaining visibility into application health and performance. Proactive monitoring helps identify issues before they impact users.
- **The Power of an Integrated Toolchain:** The project successfully integrates multiple DevOps tools (Git, GitHub Actions, Ansible, Docker, Kubernetes, Grafana) into a cohesive pipeline, demonstrating how each tool plays a specific, vital role in the software delivery lifecycle.

THANK YOU