Devops Assignment

DevOps Engineer

Division

Tech

Location

New Delhi (South)

Objective:

This assignment builds upon the use of technology stack to include Go, Next.js, TypeScript, PHP, and WordPress. The focus is on implementing CI/CD pipelines for applications developed using these technologies while enforcing coding standards using appropriate tools.

Scenario:

Your web application, "MyApp," consists of independent components developed in Go, Next.js (TypeScript), and WordPress. To ensure quality and efficiency, each component should have its own dedicated CI/CD pipeline. Additionally, enforce coding standards within each component using appropriate tools specific to the chosen language or framework.

Tasks:

Version Control:

- Create a new public GitHub repository to include directories for Go, Next.js, and WordPress components.
- Initialise the repositories for each component.

Continuous Integration:

- Implement CI workflows for Go, Next.js (TypeScript), and PHP (WordPress).
- Configure pipelines to trigger on pushes to the respective branches (e.g., main, feature branches).

- Integrate linting and unit testing for each technology.
- Ensure that the CI pipelines fail if coding standards or tests are not met.

Containerization:

- Dockerize the Go application, the Next.js (TypeScript) application, and the WordPress site.
- Create Dockerfiles for each application.
- Push the Docker images to a container registry (e.g., Docker Hub, AWS ECR).

Coding Standards Enforcement:

- Implement PHPCS (PHP_CodeSniffer) for the WordPress component.
- Configure PHPCS to enforce WordPress coding standards.
- Integrate PHPCS checks into the CI pipeline for the WordPress component.
- Implement GolangCI-Lint for the Go application.
- Configure GolangCI-Lint to enforce coding standards for Go.
- Integrate GolangCI-Lint checks into the CI pipeline for the Go component.
- Implement ESLint and Prettier for the Next.js (TypeScript) application.
- Configure ESLint and Prettier to enforce coding standards for TypeScript.
- Integrate ESLint and Prettier checks into the CI pipeline for the Next.js (TypeScript) component.

Orchestration:

- Update the Docker Compose file to include services for Go, Next.js, and WordPress.
- Ensure that the Compose file can be used to spin up the entire extended application stack locally.

Continuous Deployment:

- Extend the CI/CD pipelines to include deployment stages for the Go, Next.js, and WordPress components.
- Set up automatic deployment to a staging environment for successful builds.

Documentation:

- Update the README.md file with instructions on how to set up and run the extended application locally.
- Document any changes made to the existing documentation based on the additional technologies.

Submission:

Provide a link to your updated GitHub repository containing the extended CI/CD pipelines and Dockerized applications. Include any additional documentation or notes regarding your implementation choices and challenges.

Evaluation Criteria:

Your submission will be evaluated based on the effective integration of coding standards enforcement tools for each technology. Ensure that each component's coding standards align with best practices for the respective language or framework.