DevOps Technical Assessment: PHP GitFlow & Automation

Objective

To demonstrate the ability to manage a complex branching strategy, automate deployments to a Linux environment, and handle environment-syncing using PHP

1. Repository & Branching Setup

Create a public GitHub repository with a boilerplate **Laravel 11+** application. You must implement and demonstrate the following branching lifecycle:

- **Main Branches:** `Production` (Stable) and `Develop` (Integration/Testing).
- **Feature/Bugfix Branches:** Create a branch named `Feature/JIRA-123-login-audit` and `Bugfix/JIRA-456-session-fix`.
- **Release Flow:** Demonstrate merging `Develop` into a temporary `Release/v1.0.0` branch for final staging before merging into `Production`.
- **Hotfix Flow:** Create a `Hotfix/JIRA-789-critical-patch` branched directly from `Production`. You must merge this back into `Production`, `Develop`, and any active `Release/*` branches.

2. Task Requirements

1. **Jira Integration:** Simulate a Jira workflow. In your Pull Request (PR) descriptions, reference a mock Jira ticket ID (e.g., "Fixes JIRA-123").
2. **Conflict Management:** Ensure minimal use of **Cherry-picking**. Use standard merges to maintain a clean commit history and proper parentage across the Git tree.
3. **PHP Configuration:** Ensure the application uses separate `.env` files for `Develop` and `Production` environments.

3. Deployment Automation (The Script)

Write a Bash script (`deploy.sh`) or a PHP-based deployment script (using PHP) that performs the following on a remote Linux server:

- **Code Retrieval:** Pulls the latest code from the relevant branch.
- **Dependency Management**

- **Database Automation**
- **Optimization**
- **Permissions**

4. Deliverables

- **Link to GitHub Repository:** Showing the full commit history and closed Pull Requests.
- **Evidence:** A screenshot or log output of the deployment script running successfully.
- **Record the process**