EED git workflow

Core Principles

- **main branch is sacred:** The `main` branch exclusively holds code that is production-ready and suitable for immediate deployment to pre-production.
- **Feature Isolation:** Development of new features or changes takes place in isolated feature branches.
- **Pull Requests for Integration:** Merging features into `main` is strictly controlled through pull requests (PRs), enforcing code review and quality checks.

Branching Model

1. **main:** The primary branch representing the latest pre-production-ready code. All merges into `main` trigger pre-production deployments.
2. **develop:** Serves as the integration branch for completed features. Code in `develop` should be stable but may not be fully production-ready.
3. **feature/*:** Short-lived branches created off `develop` for developing individual features or fixes. Names follow a convention like `feature/new-widget`.
4. **fix/***: Short-lived branches created off `develop` for fixing bug or issues.

Workflow

1. **Start a Feature:**

    - Create a new branch from the latest main or whatever is your starting point:

      ```
      git checkout -b feature/my-awesome-change
      ```

2. **Develop the Feature:**

    - Make code changes and commit regularly to your feature branch.
    - Push your branch to the remote repository to share and back up code.
3. **Create a Pull Request:**

    - Once your feature is complete and tested locally:
        - Push your feature branch to the remote repository.
        - Create a PR targeting the **`main`** branch.
        - Describe your changes and the rationale behind them.
4. **Code Review & Testing:**

    - Collaborators review and suggest improvements.
    - Address feedback by making additional commits to your feature branch.
    - automated tests run creating temporary testing environment based on your feature branch for more thorough review.
5. **Merge into main:**

    - Once the PR is approved, merge the feature branch into `main`.
    - After the PR from `branch` to `main` is approved and merged, a deployment pipeline is automatically triggered to deploy the updated code to EED pre-production environment.

Additional Considerations

- **Hotfixes:** For critical production issues, create branches directly from `main` (e.g., `hotfix/critical-bug`). Merge hotfixes simultaneously back into both `main` (for immediate fix) and `develop` (to incorporate into ongoing development).
- **Release Branches:** For managing formal releases to production, you may introduce release branches forked from `develop` to harden features and prepare for production deployment.
- **Versioning:** Adopt a versioning scheme (e.g., Semantic Versioning) to track pre-production releases.

Tooling

- **Git clients:** Support pull requests and efficient branch management (e.g., GitKraken, command-line Git).
- **CI/CD Pipelines:** Implement automated testing and deployment to your pre-production environment, triggered by merges into `main`.

Advantages

- **Clean release history:** `main` maintains a well-defined history of deployable code.
- **Enforced review:** Changes undergo review before reaching pre-production.
- **Parallel development:** Multiple features can be developed in isolation.

- Develop a NodeJS Backend
- Develop a React UI frontend
- EED git workflow
- CI/CD implementation
- How to deploy webapp on kubernetes