Basic Rules

1. **Lowercase and Hyphen-separated**: Stick to lowercase for branch names and use hyphens to separate words. For instance, `feature/new-login` or `bugfix/header-styling`.
2. **Alphanumeric Characters:** Use only alphanumeric characters (a-z, A-Z, 0–9) and hyphens. Avoid punctuation, spaces, underscores, or any non-alphanumeric character.
3. **No Continuous Hyphens:** Do not use continuous hyphens. `feature--new-login` can be confusing and hard to read.
4. **No Trailing Hyphens:** Do not end your branch name with a hyphen. For example, `feature-new-login-` is not a good practice.
5. **Descriptive:** The name should be descriptive and concise, ideally reflecting the work done on the branch.

Branch Prefixes

Using prefixes in branch names helps to quickly identify the purpose of the branches. Here are some common types of branches with their corresponding prefixes:

1. **Feature Branches:** These branches are used for developing new features. Use the prefix `feature/`. For instance, `feature/login-system`.
2. **Bugfix Branches:** These branches are used to fix bugs in the code. Use the prefix `bugfix/`. For example, `bugfix/header-styling`.
3. **Hotfix Branches:** These branches are made directly from the production branch to fix critical bugs in the production environment. Use the prefix `hotfix/`. For instance, `hotfix/critical-security-issue`.
4. **Release Branches:** These branches are used to prepare for a new production release. They allow for last-minute dotting of i's and crossing t's. Use the prefix `release/`. For example, `release/v1.0.1`.
5. **Documentation Branches:** These branches are used to write, update, or fix documentation eg. the README.md file. Use the prefix `docs/`. For instance, `docs/api-endpoints`.

Sample Branch Names

Here are some samples of good branch names following the above conventions:

1. `feature/T-456-user-authentication`

2. `bugfix/T-789-fix-header-styling`
3. `hotfix/T-321-security-patch`
4. `release/v2.0.1`
5. `docs/T-654-update-readme`

Commit

The “type” field must be chosen from the options listed below:

✅ Accepted Commit Types	
Type	Purpose
feat	Introduces a new feature feat(auth): add Google OAuth login
fix	Fixes a bug fix(api): resolve 500 error on user registration
refactor	Code refactoring (no functionality change) refactor(db): optimize query performance for user lookup
perf	Performance improvement perf(cache): improve caching mechanism to reduce load time
style	Code style changes (whitespace, formatting) style(ui): fix button alignment on mobile view

✅ Accepted Commit Types

Type	Purpose
test	Adding or updating tests test(payment): add unit tests for PayPal integration
chore	Miscellaneous tasks (CI/CD, build scripts) chore(deps): update eslint to latest version
docs	Documentation updates docs(readme): update installation guide for Windows users
build	Build system changes (dependencies, compilers) build(webpack): upgrade to Webpack 5 for better bundling
ci	Changes to CI/CD configuration ci(github-actions): add workflow for automated testing
revert	Reverts a previous commit revert(auth): rollback OAuth implementation due to security concerns