Dev Tip #1

A Clean Git Workflow for Team Collaboration

Why this matters

Many teams still push code directly to the `main` branch.

This often leads to: - broken production code - conflicts between developers - unstable releases - a messy Git history

A clean Git workflow helps teams collaborate efficiently **without breaking** `main`.

The Golden Rule

> **Never push directly to** `main`.

The `main` branch must always remain: - stable - deployable - protected

All work should happen in **feature branches**.

Step 1 – Start from an up-to-date main branch

Before creating a new branch, always sync with `main`.

```
git checkout main
git pull origin main
```

This ensures your work starts from the latest stable version.

Step 2 – Create your own working branch

Create a dedicated branch for your task.

Recommended naming conventions

- `feature/login-ui`
- `fix/payment-bug`
- `refactor/auth-service`

Example

```
git checkout -b feature/auth-improvement
```

You are now working safely on your own branch.

Step 3 – Make small, focused changes

Best practices: - focus on **one feature or fix per branch** - avoid mixing unrelated changes - test your code locally if possible

Small changes are easier to review and merge.

Step 4 – Commit your changes properly

Check modified files:

```
git status
```

Stage your changes:

```
git add .
```

Write a clear commit message:

```
git commit -m "feat: improve authentication flow"
```

Good commit messages are:

- short and descriptive
- written in English
- prefixed with `feat`, `fix`, `refactor`, `docs`, etc.

Step 5 – Push your branch to the remote repository

```
git push origin feature/auth-improvement
```

This publishes your branch so others can review it.

Step 6 – Keep your branch up to date

If `main` has changed while you were working:

```
git checkout main
git pull origin main
git checkout feature/auth-improvement
git merge main
```

Resolve conflicts if needed, then commit again.

What NOT to do

- ❌ Push directly to `main`
- ❌ Work without a branch
- ❌ Create vague commits like `update` or `fix stuff`

Quick Workflow Summary

```
git checkout main
git pull origin main
git checkout -b feature/my-branch
# work on code
git add .
git commit -m "clear message"
git push origin feature/my-branch
```

Final Notes

This workflow: - keeps `main` clean - reduces conflicts - improves team collaboration - makes code reviews easier

Use it consistently, and your team will thank you.

Created by Sena Software Engineer Practical Dev Tips – Every Tuesday