Secure Software Design

Andey Robins

Spring 23 - Supplemental 2

Distributed and Open Development

Difficulties of OSS

- Communication
- Toolchains
- Organization
- Contributing

Communication

- IRC
- Mailing Lists
- Issues

IRC (and Discord, Slack, Teams, etc.)

Instant messaging is usually the place an OSS project will send you for direct help with setup or contributing. A small group of regulars usually emerges and this will often be the first place where people share ideas. Discussion of ideas is common to prepare ideas before formally proposing them.

Mailing Lists

Mailing lists are much more common in older OSS projects. Things like the Linux kernel were developed over mailing lists. I'm not aware of modern/new projects using these, though you can still view the email threads of famous discussions, making them a nice historical tool and good for that purpose.

Issues

The most common coordination tool in any projects I've worked on. Ideas, problems, or fixes are discussed in an issue and then someone can claim the issue as something they'll be working on. Big projects will use issues as their atomic planning elements and coordinate using milestones, goals, and tags.

Git Tricks

- Rebasing
- Good branching
- Good forking

Rebasing

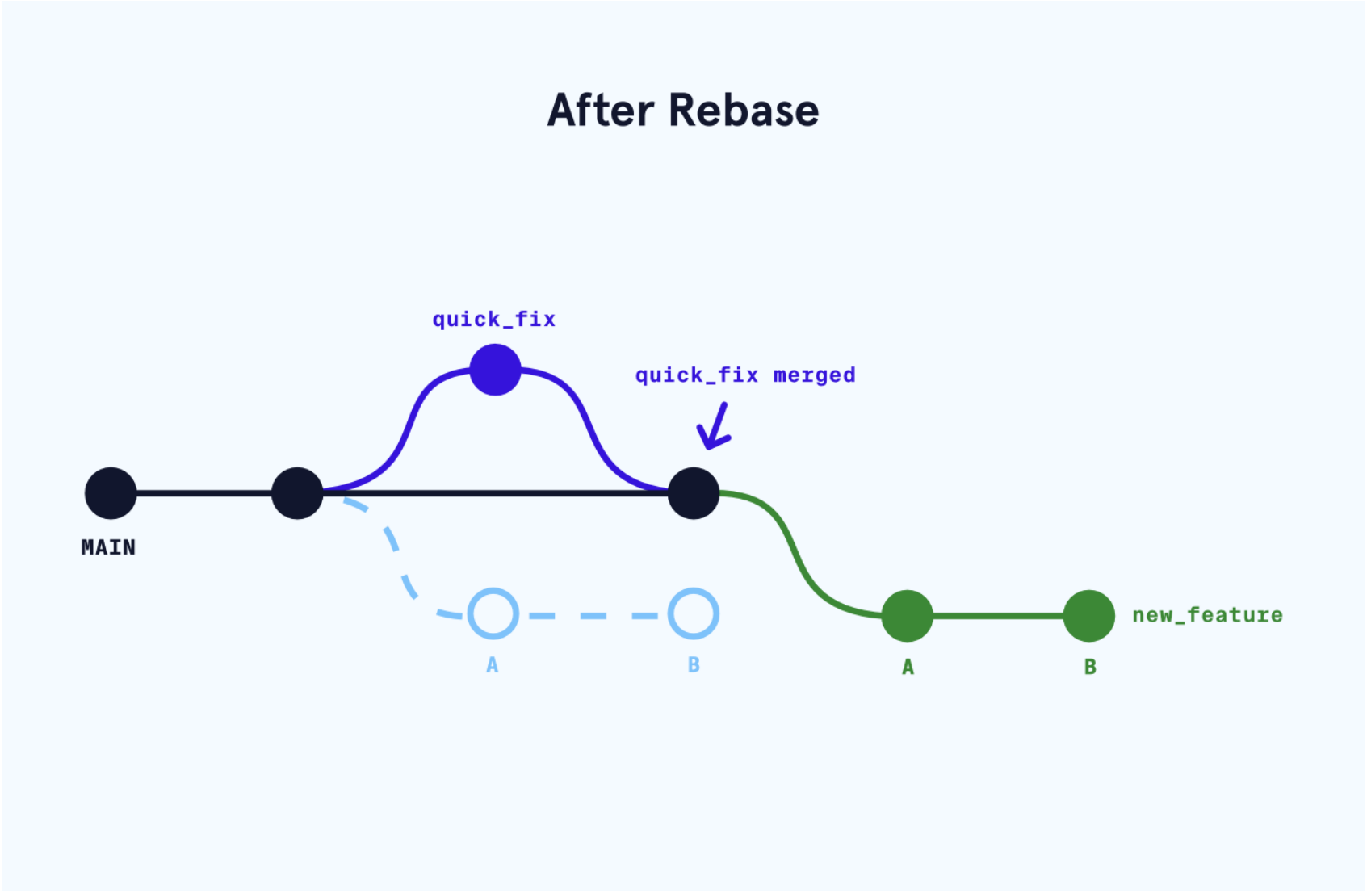

Figure 1: Rebasing Visual

Branching Strategies

- Feature branching
- Personal branching

Feature branching is where you create a branch where you work on a single, atomic feature. How many people made a `logging` branch for deaddrop?

Forking

Forking is why personal branching really falls out of favor. Instead of having your own branch where you can work on everything, just fork it and then you can do exactly that.

Long Term Organization

Multiple strategies are used:

1. Foundation ownership
2. Milestone targets
3. RFCs
4. Ad hoc issue triage

How do Things Get Prioritized?

If someone is willing to work on it, it gets done, otherwise, it probably doesn't. Sometimes project maintainers will pick up needed jobs in the interest of the project, but usually if someone doesn't want to work on it, it dies in *issue purgatory*.

How to Get Started

1. Find a project
2. Read onboarding information
3. Find an issue
4. Code
5. Open a PR
6. Review, edit, rinse, and repeat

Finding Projects

Github topics and discovery

What to Look For

1. “good first issue”
2. README.md
3. CONTRIBUTING.md
4. CODE_OF_CONDUCT.md

A good example: dotenv-linter

Pull Requests

Demo

Code Review

1. Mentality
2. Focus on improvement
3. Verify CI/tests first

Questions?