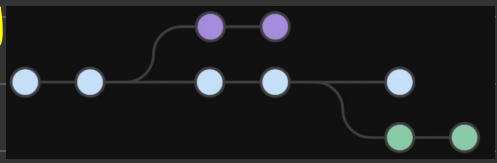
Branching, Merging and Rebasing

Nitin Singh

Non-linear Development: Branching



What is Branching.

A Branch in git is simply a light weight movable ptr to one of the commits.

The default branch name in git is master.

Branching: diverging from the main branch.

git actually follows the Non-linear development.

Why Branching is required.

The idea of branching is that we can diverge from main line Project and continue our work without messing the actual code or Project.

We can take a branch from a particular commit, to try some changes without impacting the actual Proj. or we can say that, Trying out working on other features without messing the main Project code.

In other VCS, this operation is expensive.

Killer feature of git.

Merging in git is very light weight.

Creating New Branches

Head: It is a pointer which points at which branch you are in.

It will always be at the last commit.

```
git branch
```

get all the branches in your Project.

```
git branch -v
```

To know what was the last/recent Commit in that branch.

Creating a new branch.

```
git branch branchname
```

Switching b/w branches

- Checkout an existing branch.

```
git checkout branchname
```

- Creating and checkout to a new branch.

```
git checkout-b branchname
```

Deleting branches

Idea of deleting a branch, is to when we have Merged all the changes.

Deleting a Merged branch.

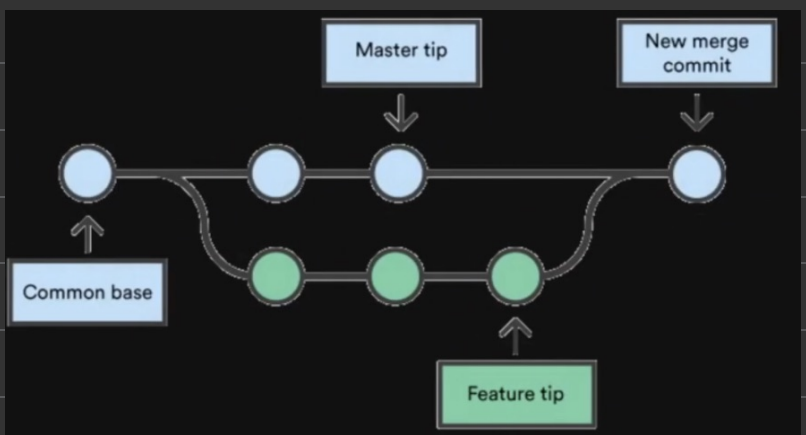
```
git branch -d branchname
```

Deleting a Non-merged branch.

```
git branch -D branchname
```

Merging

How to Merge Branches.



(Say)

IF we want to Merge, Impfix into master branch.

impfix → Master

then we need to checkout to master first.

git merge impfix.

This will Merge (impfix branch) to master branch.

(Simplest Merging)

Merge Conflicts.

Merge conflicts happens when 2 diff. branches tries to change the same line of code in the same file.

When one file is being made changed by 2 diff. branch at the same location, then git doesn't know which change should it accept.

Resolving Conflicts

As a programmer, we have to resolve the merge conflicts Manually.

git branch —merged

git branch —no-merged

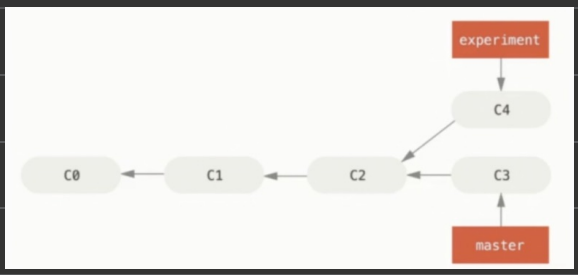
To display merged & Non-merged branches only.

Git Branching Workflow in Production

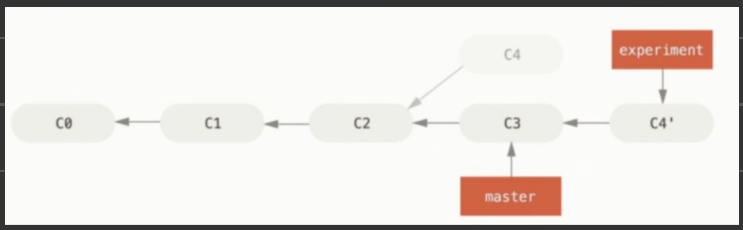
Branching Workflow.

- **Long Running Branches**
  - → Master Branch.
  - → Development Branch.
- **Topic Branches.**
  - → Authentication
  - → UI changes.

Rebasing



With the rebase command, we can take all the changes that were commited one one branch and replay them on a different branch.



Purpose of rebasing is to take the changes from one branch & integrate into diff. branch.

Eventual outcome of rebasing will be same as Merging., but the process of integrating the changes are diff. in Merging & Rebasing.

How git Performs rebase internally.

1. Pointer goes to the common ancestor of the two branches (the one you're on and the one you're rebasing into.
2. Gets the diff. introduced by each commit of the branch you're on.
3. Saves those diffs to temporary files.
4. Resetting the current branch to the same commit as the branch we are rebasing onto.
5. Finally applying each change in turn.

Key points on Rebasing.

No difference in the end product of the integration but rebasing Makes for a cleaner History.
The log History looks like a linear History. It appears that all the work happened in series. even when it originally happened in parallel.
Often, we'll do this to make sure our commits apply cleanly on a remote branch.

When Not to use Rebase

Don't rebase commits that exist outside your repo and that people may have based work on.

When we rebase stuff, we are abandoning existing commits & creating new ones that are similar but different.

Merge vs Rebase.

One point of view is.

Commit history is a record of what actually Happened.

Opposing point is

Commit history is the story of how your project was Made.