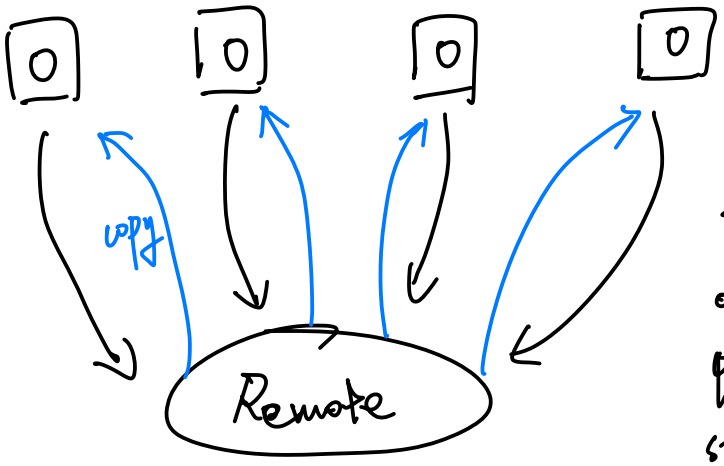
? Repository . { Remote / Local. } > Sync automatically.

Imagine 4 developers have their own code on their machine.

And, there's a remote one.



they make changes on their local ones and push them to the remote so that someone else make some changes

GitHub tracks

COMMIT

A commit is simply a change that you've made.

It's actually different versions you manually made.

Let's say I had a new feature, a bug fixed. So I commit that. And in that way, if anything messed up in the future, I can always come back to that commit. And find that at this time, that was functioning.

In GitHub. we have branches.

The master branch. We pull commits up and down.

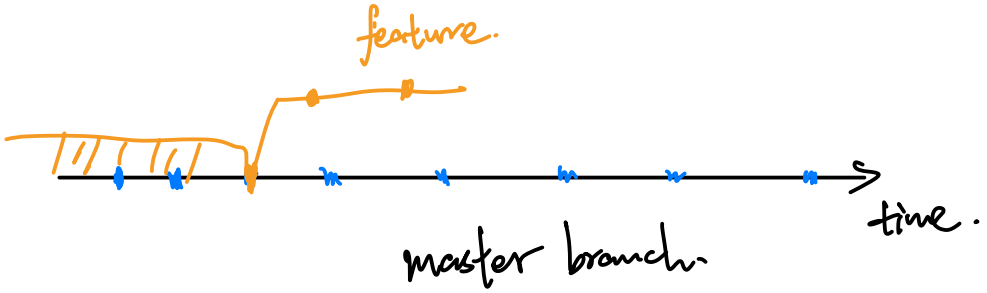
The remote repository, and even your local repository as well, keeps track of all of these commits and everything that happens to the repository.

So at any point in time, something goes wrong. I can always roll back to a previous commit. Or I can just look at all of the history to see what's happened in the code base, which is very valuable.

We wanna make sure the master branch is always working!!!

Instead always making change to the master branch. Sometimes we use what is known as another branch.

A branch is essentially a copy of repository at a certain point in time that has different changes.

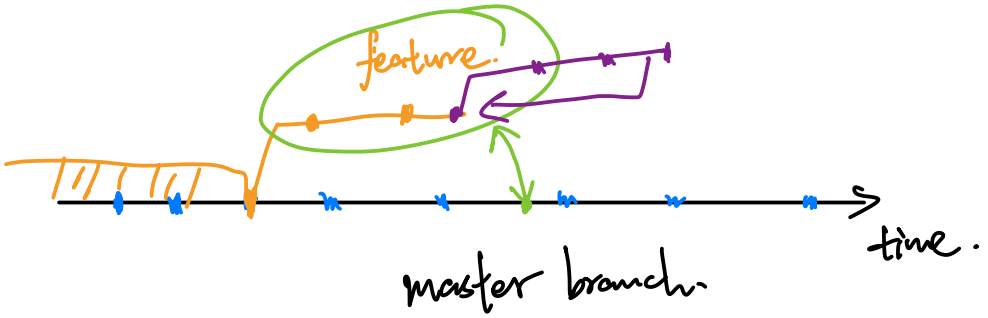


The point of this branch (let's just say feature) is essentially to be able to work on a new feature, push it up to the remote repository without actually directly putting it into the master branch.

So if I make a new branch (feature). Now this branch has whatever the repository had up until this point. It has all of the changes before. And the dots are new changes on the new branch.

So I'm just making different changes to the branch so I'm not affecting the master branch until I'm sure the branch is fully functioning.

So soon as the new branch is actually working, what I'll do is to merge it with the master branch.



I can ever branch off branches.