You gave them **push** access to the repository.

(and [other](https://help.github.com/articles/permission-levels-for-a-user-account-repository/#collaborator-access-on-a-repository-owned-by-a-user-account) rights)

Now, collaborators can **clone** the project on their own computer:

$ mkdir ~/code/OWNER\_GITHUB\_USERNAME $ cd ~/code/OWNER\_GITHUB\_USERNAME $ git clone git@github.com:OWNER\_GITHUB\_USERNAME/PROJECT\_NAME.git $ cd PROJECT\_NAME

Clone with **SSH**, **not** HTTPS.

**Working as a team**

**Think of features (user stories)**

**Problems**

* Overlap (we both need to change the RestaurantsController)
* Dependency (I need your feature done to start mine)

**Solutions**

* Technical (git branching)
* Human (communication)

**Git Branching**

When cloning a repository, you're by default on a branch, the originating branch.

In Git this branch is called **master**.

So basically, any collaborator can create a **new branch from master** branch and start working on a **new feature** (or bug, etc).

**Listing local and remote branches**

$ git branch -a \* master remotes/origin/master

We're currently on master branch (the one with an asterisk). There's also a remote master branch (the one pushed on GitHub).

**Working on a new branch**

$ git checkout -b add-description-to-restaurant $ git branch -a \* add-description-to-restaurant master remote/origin/master

We've created a new local branch called **add-description-to-restaurant**. Locally, we're currently on this new branch.

Any new commit will only be applied to this branch.

**Pushing a branch to GitHub**

$ git push origin add-description-to-restaurant $ git branch -a \* add-description-to-restaurant master remote/origin/add-description-to-restaurant remote/origin/master

We've pushed the branch to GitHub.

**Finishing a feature**

Using branches, we work on different parts of a project at the same time.

When a feature is finished, we'd like to have its related **commits back on master**.

**How?**

**GitHub Pull Requests**

**Usages**

* Get feedback on code written in a given branch (code review)
* Merge the branch back into master

**A Pull Request is a conversation**

Exemple: [lewagon/www-sinatra#45](https://github.com/lewagon/www-sinatra/pull/45).

**Creating a Pull Request (1)**

As soon as you push a new branch, a pull request button appears on your GitHub repository page.

Just click on this button, review the diff and add clear title and description.

**Creating a Pull Request (2)**

* Take some time to write a proper **title** and **description**
* Explain what you did on the branch (gem added, code tricks, etc...)
* Ease the reviewer's job
* You can poke people with @..., like @ssaunier or @papillard to get their feedback

**Reviewing a Pull Request**

* Look at the diff, comment on errors (indentation, style, useless code, etc.)
* Comment **inline** or at the pull request level
* When done:
  + If code is fine, click on "Merge Pull Request" then "Delete Branch"
  + If not, add a general comment "Review done"

**Looping over**

**Getting master up to date**

When a Pull Request is **merged**, a new commit is created on master.

You need to fetch it on your **local** repository.

**Be very careful**

First, you need a **CLEAN** git status.

$ git status # On branch master # Your branch is up-to-date with 'origin/master'. # nothing to commit, working directory clean

**Get the latest master**

$ git checkout master $ git pull origin master

Then you can clean up local unused branches

$ git sweep

**Merging master in your branches**

Do you need something in master back into your curent branch?

$ git status # MAKE SURE THIS IS CLEAN $ git checkout add-description-to-restaurant $ git merge master

**2 rules**

* **Never** commit directly to master. Use feature branches
* **Always** make sure git status is **clean** before pull, checkout or merge.

**Project Management**

Week objective: Implement your own version of AirBnB

* Teams of 2 to 4

**Day One**

* Write 5 to 10 user stories (week backlog) + Validate with teacher
* Brainstorm Data Model + Validate with teacher
* Brainstorm Routes / Draw Mockups + Validate with teacher
* Lead Dev creates rails app, github repo, invite collaborators
* Development starts. Pair program if too much dependencies at the beginning

**User stories**

As a <ROLE> I can <ACTION> So that <VALUE>

**DO NOT** commit to master. Exception, the rails app creation:

$ cd ~/code/$GITHUB\_USERNAME $ rails new rbnb -T --database=postgresql $ cd rbnb $ git init $ git add . $ git commit -m "rails new" $ hub create $ git push origin master # And now, `checkout -b` a new branch BEFORE writing code.