Git for busy people

Your collaborator invited you to contribute to a project which is hosted in a remote git repository (let's say, github or bitbucket)? You never used git or have very little experience with it. What do you do now?

This is a very quick guide on how to use git for collaborating on an existing project.

First time using git

You want to setup your identity (name and e-mail) for the future. You only need to do this once in each machine where you work:

```
git config user.name "Your Name"
git config user.email "me@example.com"
```

Download files for the first time

In other words, create a local copy ("clone") of the project in your machine:

```
git clone https://user@bitbucket.org/user/projectrepo.git
```

where of course you replace the username and project name in the command above. You probably already received the appropriate address from your collaborator. This will create a folder called "projectrepo".

Let's see the history of "saved snapshots" (commits) of the project so far:

```
git log
```

Changing things, "saving the game"

Now you decide to edit some file(s). To add a file which is not in the project, you can issue

```
git add newfile
```

To inspect what you changed with respect to the last saved state of the project:

```
git diff
```

Once you are done with your editing, you want to save the state of the project. I like to think of this using an analogy with videogames: you save your game and can easily go back in time, if things go wrong later. In the git world, saving the game is called "commit":

```
git commit -am "added the discussion section and figure with time variability"
```

When you commit, don't forget to include a short description of what you changed. If you want to describe things in more details, just issue `git commit`. This will open a text editor which will prompt you for a description of what changed. It is standard practice to begin the commit message with a single short (less than 50 character) line summarizing the change, followed by a blank line and then a more thorough description.

Uploading your changes to collaborators

So far, your changes have been saved locally in your machine and the remote server (i.e. your collaborators) do not know about them. To upload the changes to them, *first commit*, then issue

```
git push
```

Downloading updates on the project

After a while, your collaborators probably did some work and changed things. To update the state of the project, you can type

```
git pull
```

This will download the files from the server and update the project on your machine.

Creating alternate, exploratory, "savegames"

This concept is a little more advanced. Suppose you have some awesome idea which is tricky and may put the project at risk and/or introduce bugs. Instead of having one single save state which is continuosly evolving and updated by everybody (in the git language, the *master branch*), you can create a parallel save state which will have a copy of the project and keep the stable version of the project safe aside. Let's call this alternate save state "explore":

```
git checkout -b explore
```

The command above creates a *branch* (the alternate save state) called explore and switches to the new branch. Now everything you do is saved in the explore branch.

When you are done with your awesome exploratory idea and want to go back to the "stable" branch, first *commit* then

```
git checkout master
```

Once you find out that your exploration ended up improving the project and sorted out all bugs, you can merge the stable and the exploratory branches. First commit, then switch back to the master branch as in the command above, and type

```
git merge explore
```

More

I skipped here some concepts, such as creating a local repository in your machine, uploading your new local repo to a remote server, fixing a wrong commit message, defining workflows and many other things, for the sake of brevity.

You can find useful material in the following websites:

- Bitbucket 101 and their Git tutorials
- More in depth tutorial: Git Magic
- Super quick Git guide at archlinux
- A simple guite to Git (check out their cheatsheet)
- One cool cheatsheet here and here

Visit the author's web page and/or follow him on twitter (@astrorho).