So why is Git so important?

Let's first start by quoting the first line on Git's Wikipedia page:

> "***Git*** *(/gɪt/) is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people.*"

So that means that the most basic and important function of Git is to allow teams to add (and merge) code at the same time to the same project. By adding this ability to projects it makes teams more efficient and gives them the ability to work on bigger projects and more complex problems.

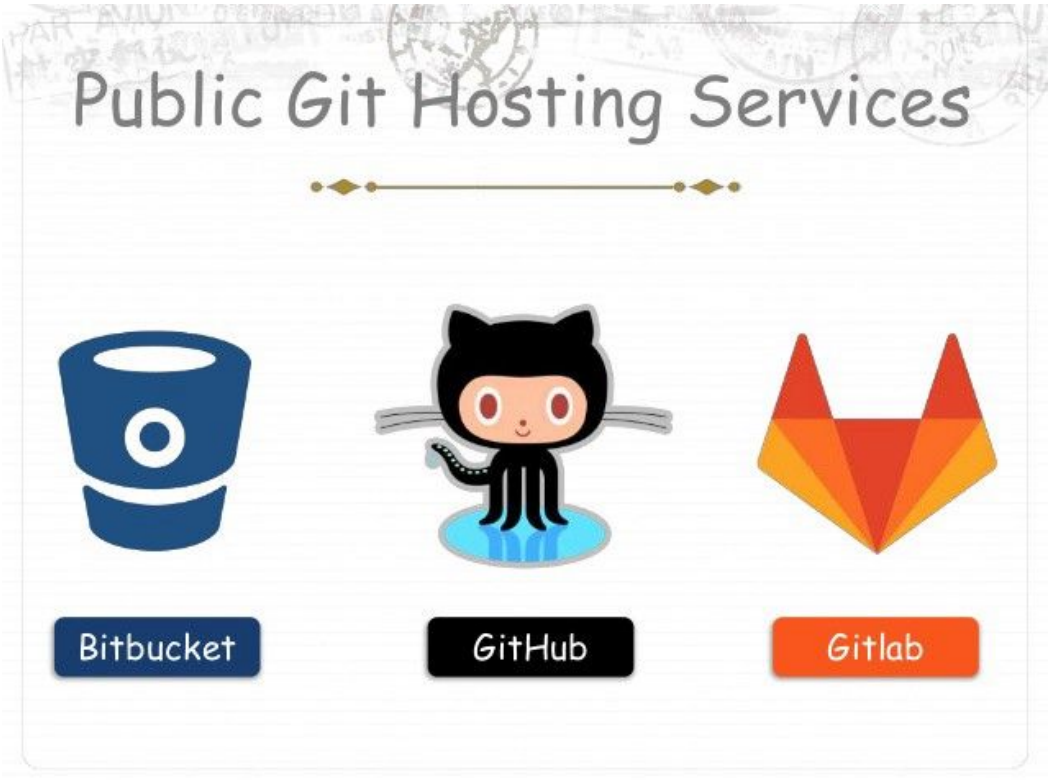
There are also many other things Git does really well: it allows us to revert changes, create new branches for adding new features, resolve merge conflict, and so on.

How Git works

Git stores projects in **repositories**. **Commits** are made to the project and they tell Git that you are satisfied with the new or changed code you created.

New code/changes are committed on branches. Most of the work is committed on other branches and then merged with the master branch. All this is stored in the same directory as the project but in a sub-folder called **.git**.

To share the code with your colleagues you **push** the changes to the repository. To get the new code from your colleagues, you **pull** changes from the repository.



Then what are GitHub, GitLab and Bitbucket?

Well, I am glad you asked! These kinds of applications are called repository management services. They play a crucial role in modern software development.

Even though Git and GitHub are the go-to version control solutions for most companies, GitHub has some strong competitors such as GitLab and Bitbucket. However, if you know how to use GitHub, you won't have any problem working with GitLab or Bitbucket.

So, to be clear: Git is the tool, and GitHub is the service for projects that use Git.

Where can I discover interesting projects and connect to other developers?

GitHub, GitLab and Bitbucket have public repository search options and the ability to easily follow other users.

Can you now see why is it important to know Git and Github (GitLab/Bitbucket)? The only thing left before talking about commands is to tell you few simple rules to always follow when using Git:

- **Rule 1:** Create a Git repository for every new project
- **Rule 2:** Create a new branch for every new feature

Commands

To get started with Git you must have it on your computer. If you don't already have it you can go here and follow the instructions.

Initialize a new Git repository: Git init

Everything you code is tracked in the repository. To initialize a git repository, use this command while inside the project folder. This will create a .git folder.

```
git init
```

Git add

This command adds one or all changed files to the staging area.

To just add a specific file to staging:

```
git add filename.py
```

To stage new, modified, or deleted files:

```
git add -A
```

To stage new and modified files:

```
git add .
```

To stage modified and deleted files:

```
git add -u
```

Git commit

This command records the file in the version history. The -m means that a commit message follows. This message is a custom one and you should use it to let your colleagues or your future self know what was added in that commit.

```
git commit -m "your text"
```

Git status

This command will list files in green or red colors. Green files have been added to the stage but not committed yet. Files marked as red are the ones not yet added to the stage.

```
git status
```

Working with branches

Git branch branch_name

This will create a new branch:

```
git branch branch_name
```

Git checkout branch_name

To switch from one branch to another:

```
git checkout branch_name
```

Git checkout -b branch_name

To create a new branch and switch to it automatically:

```
git checkout -b branch_name
```

This is short for:

```
git branch branch_name
git checkout branch_name
```

Git branch

To list all the branches and see on what branch you currently are:

```
git branch
```

Git log

This command will list the version history for the current branch:

```
git log
```

Push & Pull

Git push

This command sends the committed changes to a remote repository:

```
git push
```

Git pull

To pull the changes from the remote server to your local computer:

```
git pull
```

For more commands and a detailed explanation of those listed, I would recommend that you check out the official Git documentation.

Tips & Tricks

Throw away all your uncommitted changes

Just as it says, this command will throw away all your uncommitted changes:

```
git reset --hard
```

Remove a file from git without removing it from your computer

Sometimes, when using the “git add” command, you might end up adding files that you didn’t want to add.

If you are not careful during a “git add”, you may end up adding files that you didn’t want to commit. You should remove the staged version of the file, and then add the file to .gitignore to avoid making the same mistake a second time:

```
git reset file_name
echo filename >> .gitignore
```

Edit a commit message

It is very easy to fix a commit message:

```
git commit --amend -m "New message"
```