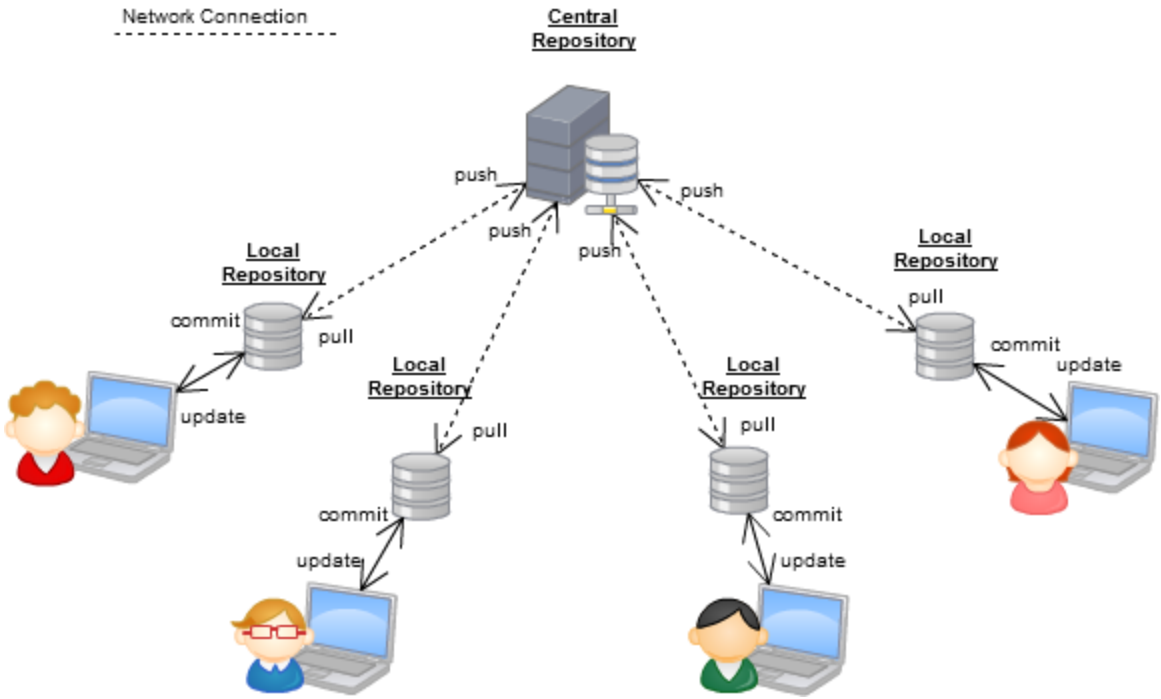
What is Git?

Git is a version control system that is used for software development and other version control tasks.
Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.

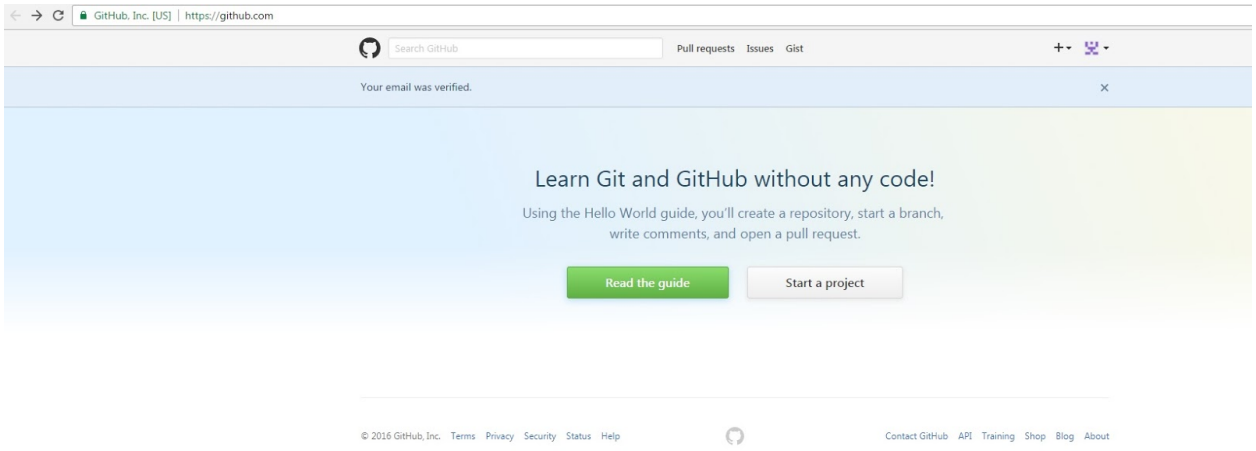
When you know you need it?

- When there are multiple people working on the same project
- When you want to keep a history for each version of your project
- When you want to keep your code on a safe place (not on you local PC)
- When you are working on a bigger project than “Hello world” :)



Getting started

1. Create an account on: https://github.com/join
2. Start the project



3.Name your *repository*
4.Initialize your repository

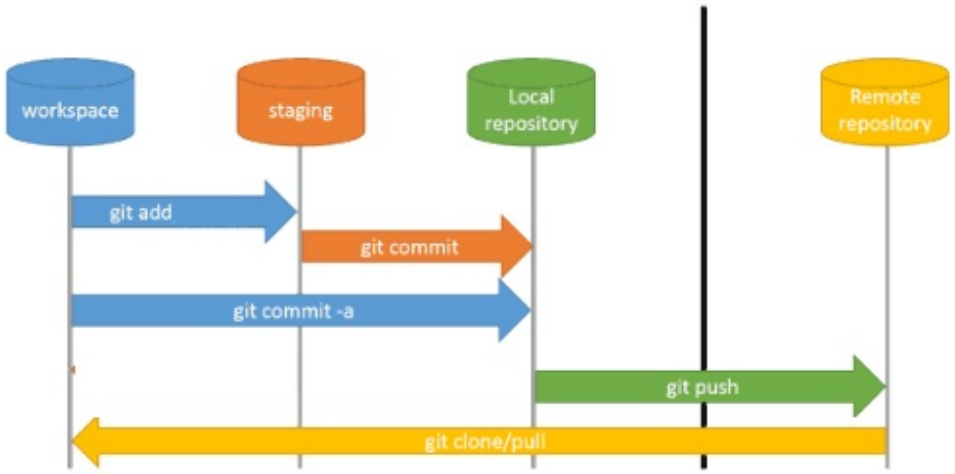
Go to command line and do the steps:

```
git init
git add .
git commit -m "first commit"
git remote add origin https://github.com/alexandraSIIT/test_web7.git
git push -u origin master
```

Congrats!You now have a brand new repository. That means your code is saved on a remote server, you will have a history with all your project version (backup) and you can add new team members to work with you on the same project.

Modifying files

After you finish modifying one or more files , you will want to *sync* your project with your team, so you will have to *push* your changes to the remote server.



1.Use **git status** command. To see all your modifications. The modified files may have one of the following statuses:

- **staged:**
  Files are ready to be committed.
- **unstaged:**
  Files with changes that have not been prepared to be committed.
- **untracked:**
  Files aren't tracked by Git yet. This usually indicates a newly created file.
- **deleted:**
  File has been deleted and is waiting to be removed from Git.

2.To *stage* a file use the command:

git add <filename>

or

git add .

3.To remove a file from the staging area use:

git reset HEAD <filename>

*The HEAD is a pointer that holds your position within all your different commits. By default HEAD points to your most recent commit.

4.After all our changes are staged, we should *commit* them! A "commit" is a **snapshot** of our repository. This way if we ever need to look back at the changes we've made (or if someone else does), we will see a nice timeline of all changes.

git commit -m “bugfix for login”

Note: If you want to **stage and commit** your changes on the same line you can use: **git commit -a -m “bugfix for login”**.

5.Great! You can see a commit history by using the command:

git log --summary

6.We’re quite there, but not really! Our commit was saved in our **local repository** , but we wanted to share it and save the code on the **remote repository** and to achieve that we should *push* our changes.

git push <repository><branch>

E.g: git push origin master

!Note: before pushing our changes to the server we should pull the latest code from the remote repository.

Done! Our code is somewhere safe and shared with all the team! But how will they get the latest version (the changes that we just pushed to the repo) ?

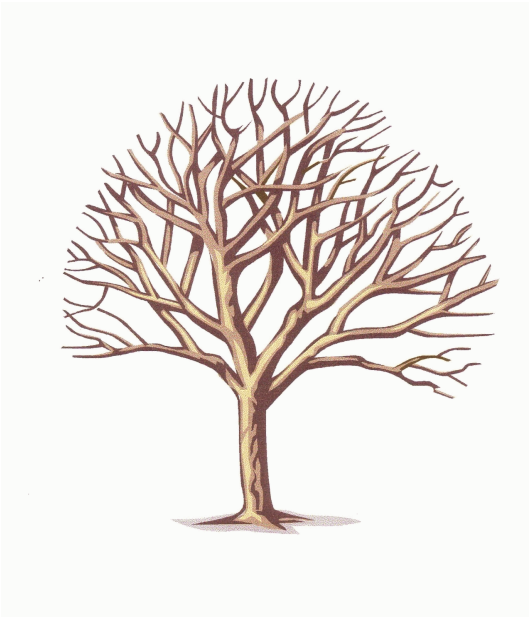
7.To get the latest version use the command:

git pull <repository><branch>

E.g: git pull origin master

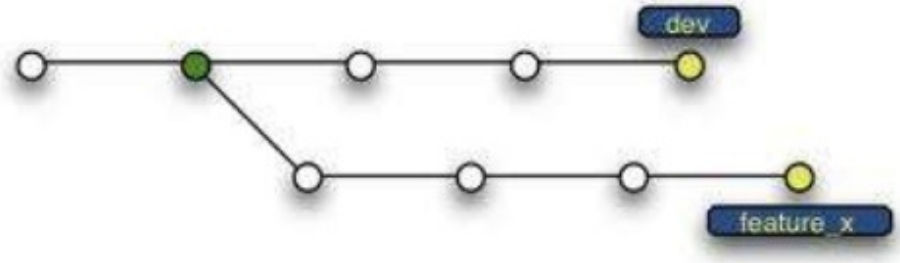
Everyone is in sync now!

Branches? What was that?



Not really….but:

Branching

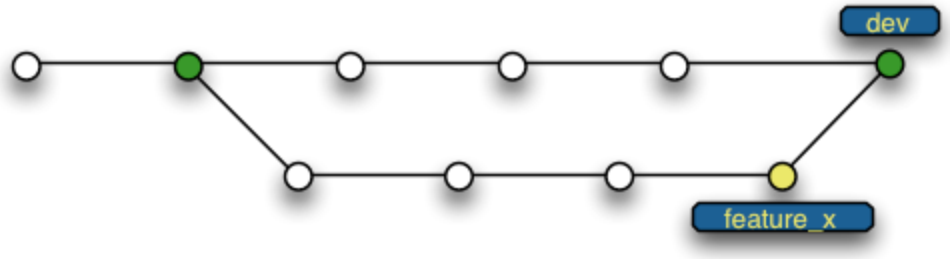



```
git branch <branch-name>
git checkout -b <branch_name>
git branch <branch-name> <start_point>
```

How do we switch branches?

git checkout <branch_name>

And eventually these branches , have to be *merged*… so the code that we wrote for feature_x is added to the main branch also.



git merge <brachname>

Note: Merge Conflicts can occur when changes are made to a file at the same time. A lot of people get really scared when a conflict happens, but fear not! They aren't that scary, you just need to decide which code to keep.

Other useful git commands are: git stash, git checkout -f, git branch -d, git branch -D, git fetch,git clone

Must do: https://try.github.io