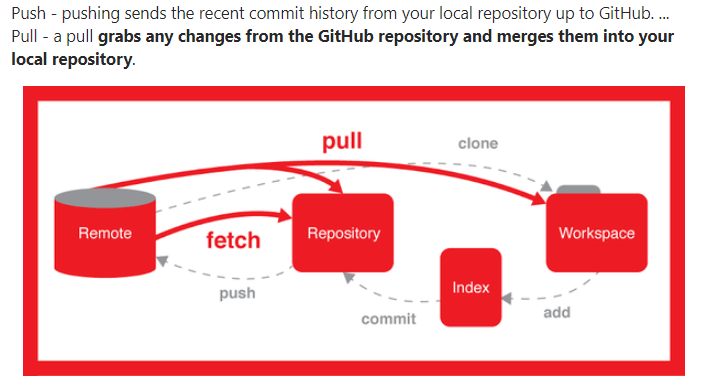
**GIT** refers to **VersionControl** . The term Version control refers to a system that records changes to a file or set of files over time called the ‘**versions**’. In other words, these versions will help you in **tracking** the changes in your **codes/project**and if necessary, undo those changes as well. Using**version control** prevent these conflicts.

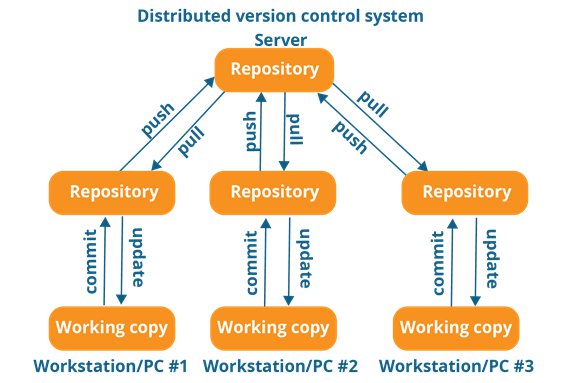
**GITHUB** is a repository hosting service. It provides all of the distributed version control and source code management . It is a Meant for the developers where they can store their projects and get connected with other **Developers .**



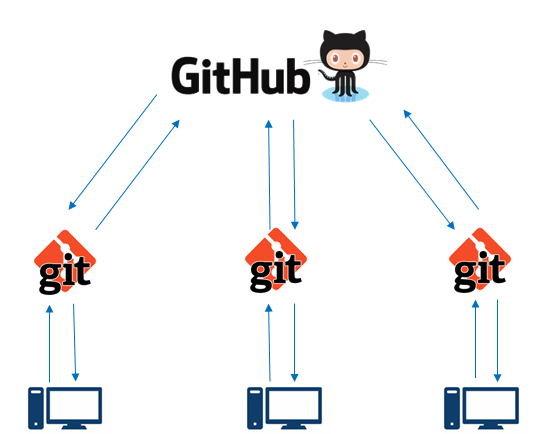
When you are using Git, you are dealing with two repositories.

1. Your local repository
2. Central repository

This is how your Distributed version control system will look like:



Let’s say your central code repository is GitHub and you are using Git as VCS. This will be the scenario:



So, when you are “**pulling**”, it means that you are fetching code from GitHub onto your local repository. All your corresponding branches in your local repo will be updated according to the changes in the central repo.

Don’t get the idea that “**git pull**” is same as “**git fetch**”. Well, they do the same thing only **git pull** means a **git fetch** which is followed by a**git merge**.

To put it in simple terms, take a look at this equation-

git pull= git fetch + git merge

If you want to learn more about Git, take a look at this video of mine. Hope you will find it useful.

“Git rules all, Git for all” ;)

Basic thumb rule of git to add files

1. git add

“Add the changes in the specified files/directories to the staging area.”

1. git commit

“Bundle up all the changes in the staging area into a single commit object, and add that commit to the current branch, in the local repository.”

1. git push

“Find the remote repository specified in the local repository's configuration, find out what commits on the local branch are not present in the upstream branch, and send those commits.”

**git pull = git fetch + git merge**

The objective of both these commands is the same - to get updates from the remote repository to my local repository.

But, they work in a different way.

**Git pull**: when you do a***git pull***, it gets all the changes from the remote or central repository and attaches it to your corresponding branch in your local repository.

Git pull just checks just existence of remote file vs local file it wont check content.

**Git fetch**: when you do a ***git fetch***, it gets all the changes from the remote repository, stores the changes in a separate branch in your local repository and if you want to reflect those changes in your corresponding branches, use a***git merge*** to do that.

**Example.sv in local**

module example (

input a\_1, // Clock

input b\_1, // Clock Enable

output c1 // Asynchronous reset active low

);

always@(posedge a\_1)

begin

c1<=b1;

end

endmodule"#newline"

#pull check

**Example.sv in remote**

module example (

input a\_1, // Clock

input b\_1, // Clock Enable

output c1 // Asynchronous reset active low

);

always@(posedge a\_1)

begin

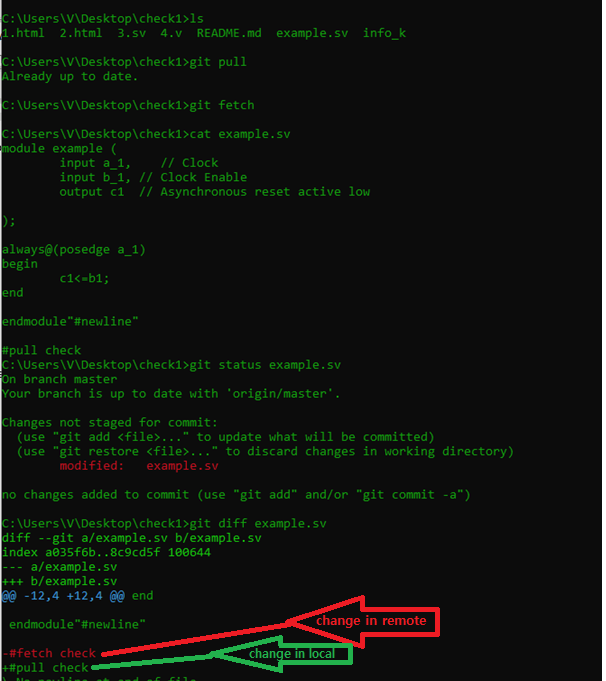
c1<=b1;

end

endmodule"#newline"

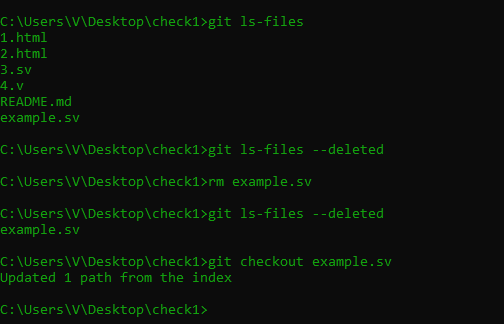
#fetch check

See below observation of pull fetch and diff

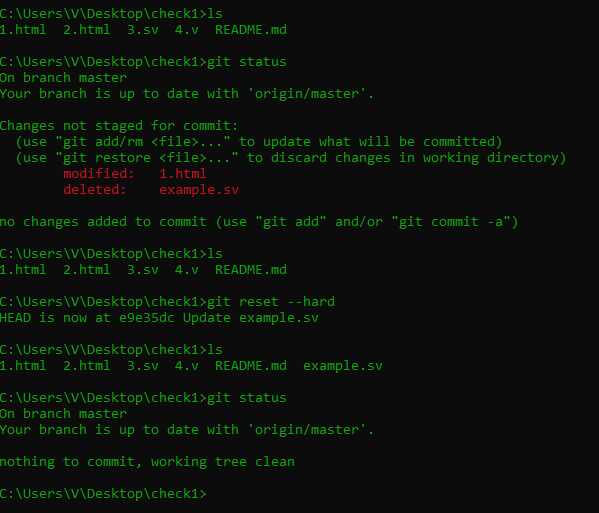


How to update deleted files in local which were present in remote “**git checkout example.sv**”

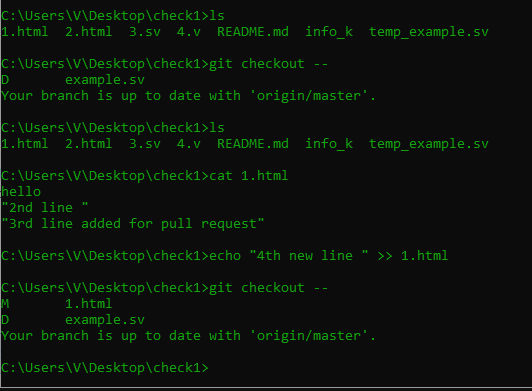
Or “**git restore example.sv**”



Another way of updating as per the previous commit “**git reset –hard**”



To check what are changes happened in the local “**git checkout --**” lists as similar to “**svn st”**



Alert:To clean your directory removing all unnecessary files “**git clean -fd**”

Note: it removed two local files which are not in remote, so it is removed.

