Introduction to GIT

What is Version Control?

Time

Your Project

VCS

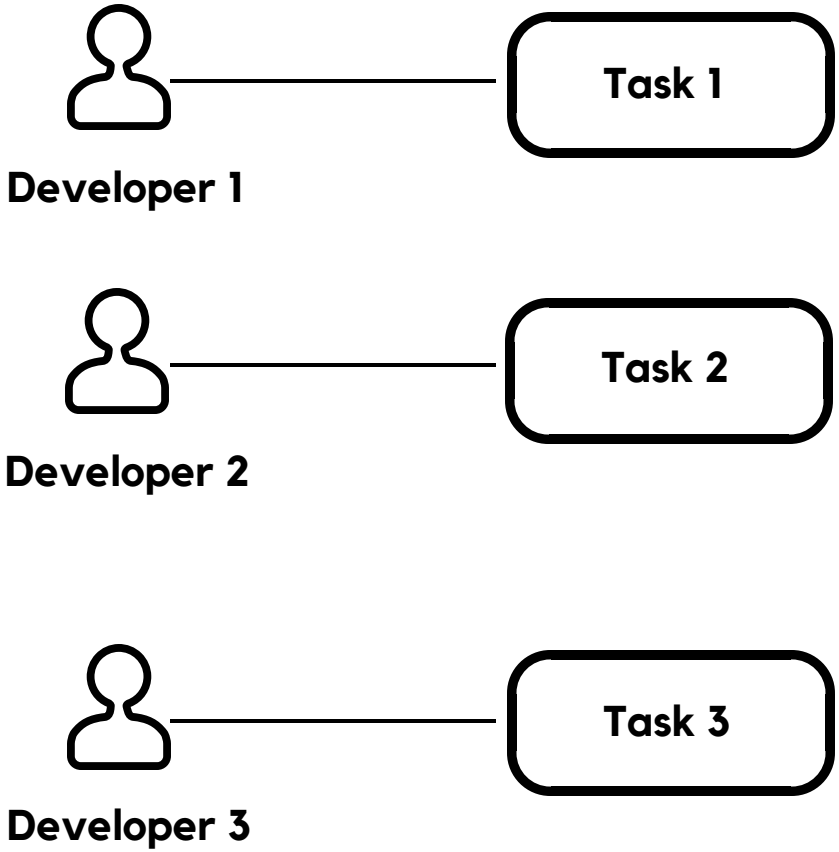
On each of these days, a 'snapshot' of the entire project with the changes is taken. Each of these snapshots is called a 'version'.

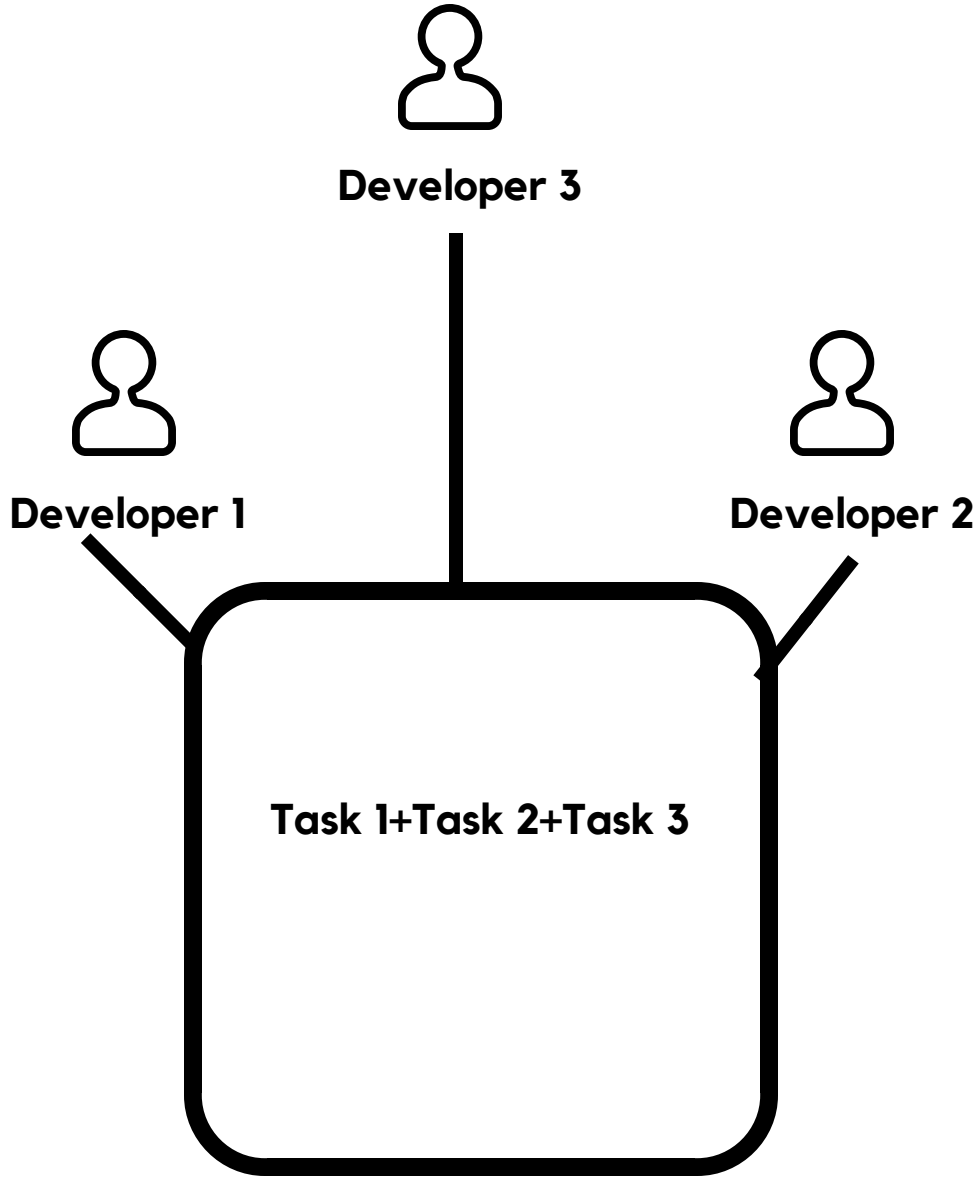
You can revert to a previous date (version) if you so wish

Introduction to GIT

Why Version Control?

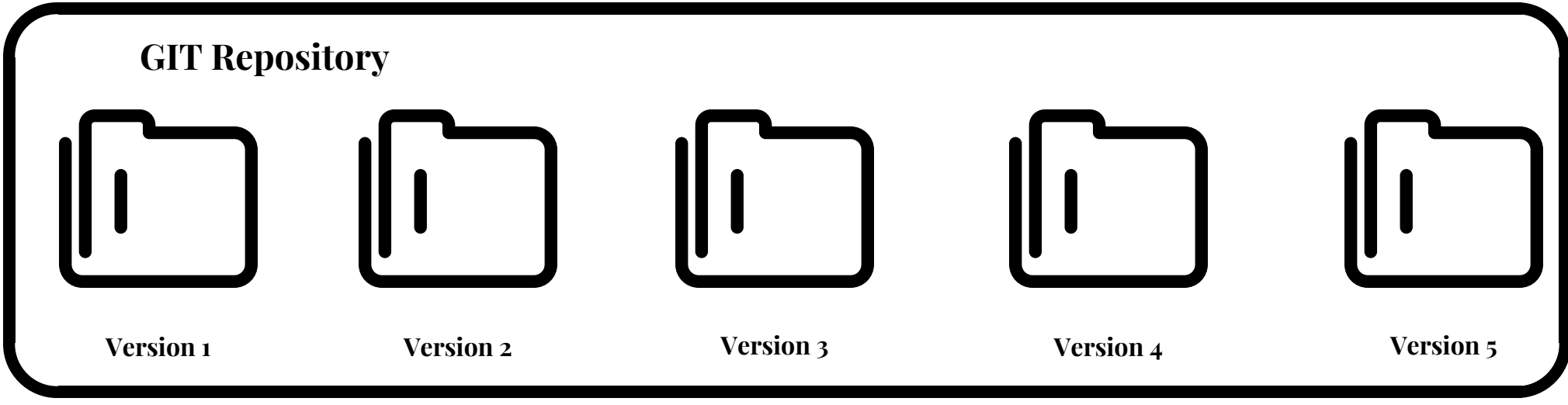
Collaboration





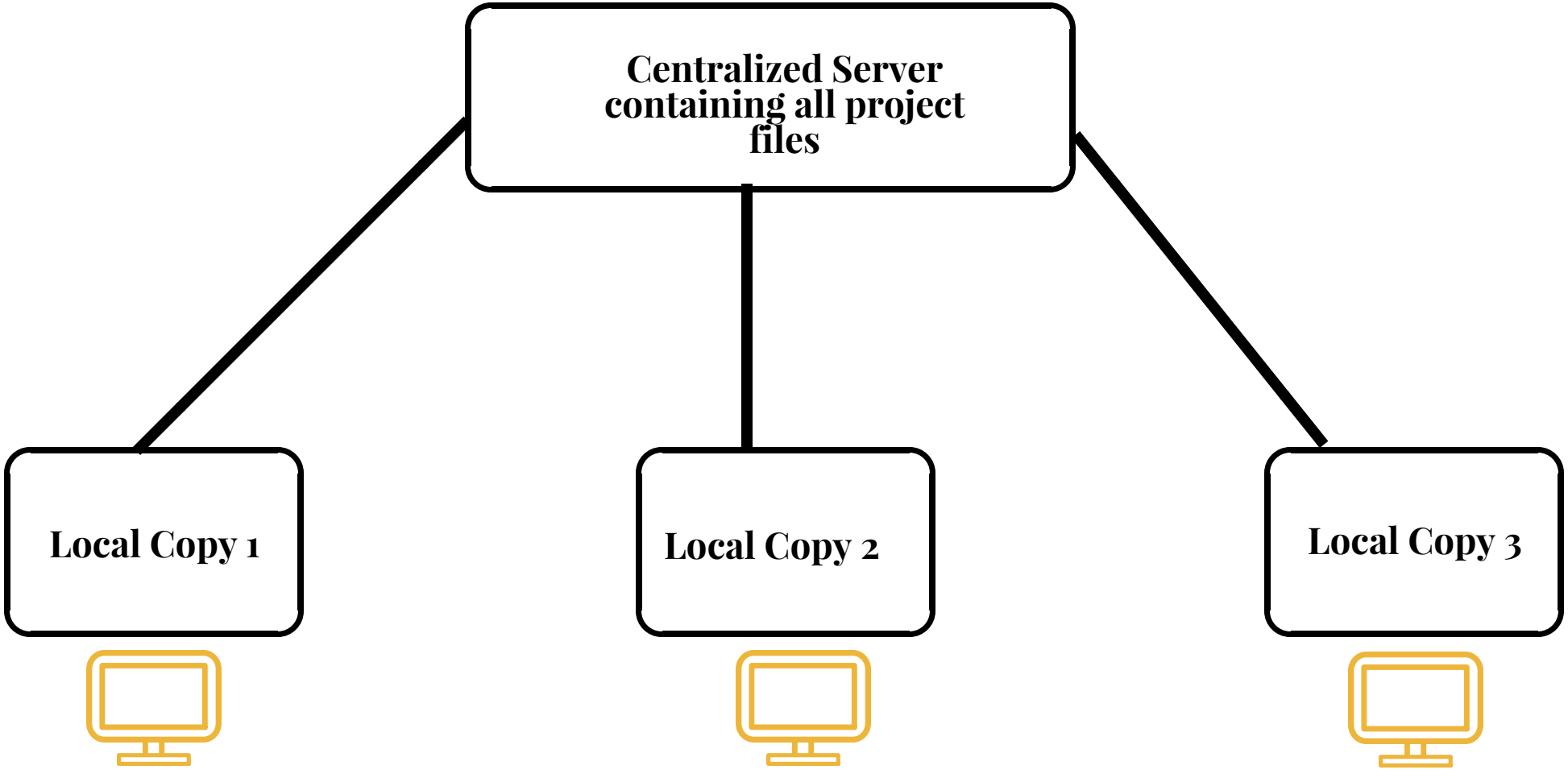
Introduction to GIT

Storing Versions



Introduction to GIT

Backup



Introduction to GIT

VCS

What was changed.

When was it Changed.

Description of the change

Introduction to GIT

Other VCS

Introduction to GIT

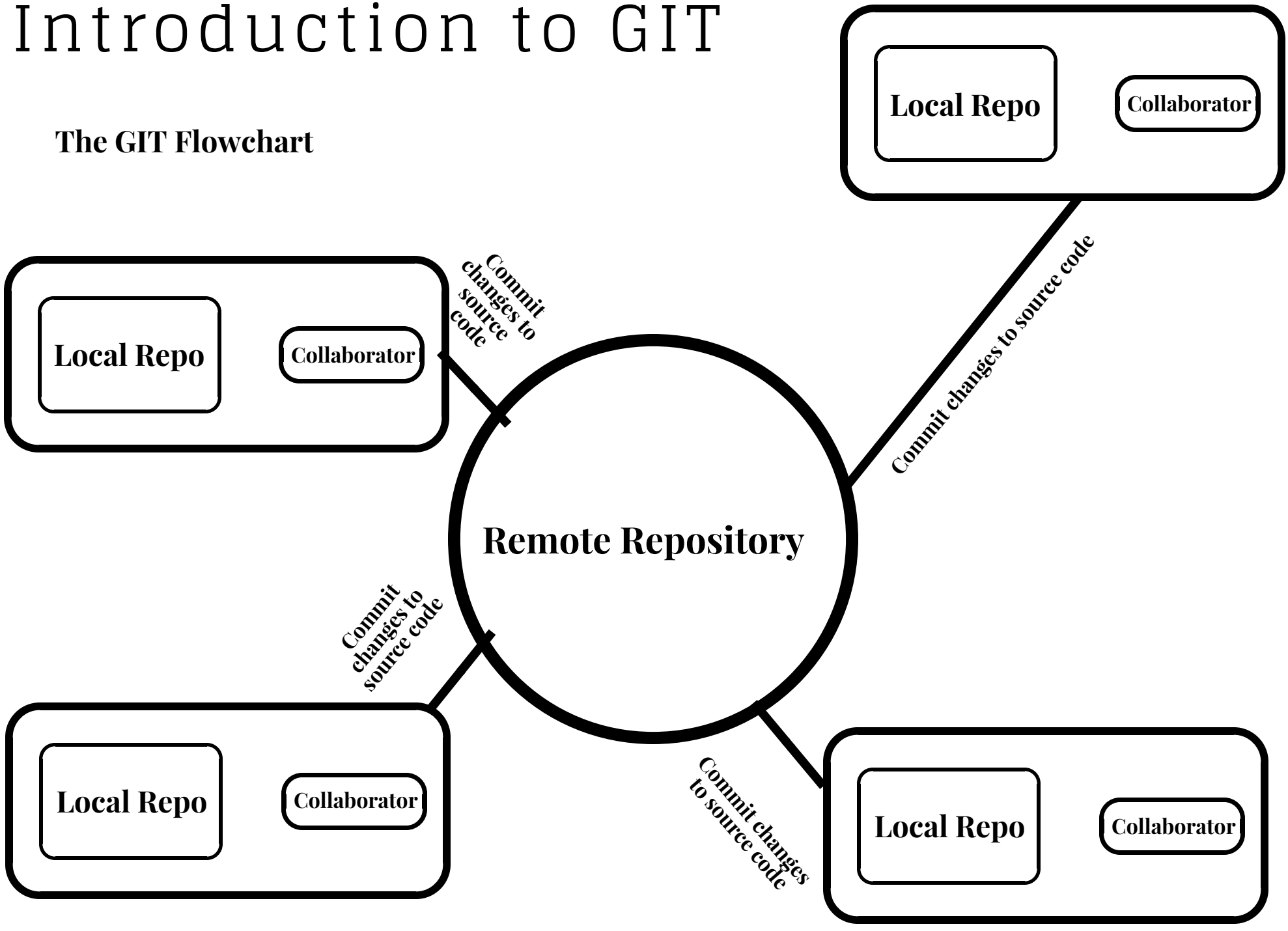
GITHub

Git is a Distributed Version Control System because the central repository keeps a copy of itself on each of the developers' machines known as local repositories

Github on the other hand is a code hosting platform for collaboration. It's like a social network for developers

Introduction to GIT

The GIT Flowchart



Introduction to GIT

Git Features

- **Distributed**
- **Compatible**
- **Non-Linear**
- **Branching**
- **Ligtweight and Speed**

Introduction to GIT

Repository

- **Central**
  - **Typically stored on a remote server**
  - **Meant for Team exchange of Data**
- **Local**
  - **Resides in your local machine as a .git folder in your project's root folder**

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 1: Creating a repo

Step 2: Syncing repos

Step 3: Making Changes

Step 4: Parallel Development; Branching, Merging and Rebasing

Step 5: Git Flow

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 1: Creating a repo

Create a central repo

Go to github.com, create an account

Click on start a project

Make your repo public

Name your repo. We will name it greyatom7july

We will also initialize this with a ReadMe.

We also have options to add licences. We will leave it as None

Our first repo is ready

We will edit the readme file. Add our name perhaps

"Commit the changes".

The repo shows the last commit timestamp.

Step 2: Syncing repos

Create a local repo

Install git

Create a folder called greyatom_git

For mac right-click, click terminal in folder.

For windows, right click, click on git bash here

Type git init and press enter

For windows we will have a .git hidden folder once we do this

Now to link this local to the central repo

Go to our github project webpage, click on clone/download, copy the link

Go to your terminal or gitbash

Type git remote add origin "your git link" and press enter

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 3: Making Changes

Doing a GIT Pull
(Always do this befoer starting to code)

After our local and remote repos have been linked using the remote add origin method, we will type this:

git pull origin master. Press enter

All files are now fetched onto our local repo

Step 3: Making Changes

Doing a GIT Push (Always at the end of the day's code session)

Create a new text file. Name it after your name. Fill it with something.

type git status in the terminal

The text file needs to be committed to the local repo before adding to the central repo.

type git add your text_file

Now git status again

Changes are ready to be committed now as they are in the index

git commit -m "your message"

For multiple files, the process remains the same.

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 3: Making Changes

To add multiple files type

git add -A

Create a few more text files in our local repo. Modify the first text file.

type git status. Notice the changes in red

type git add -A

Notice the changes in green

To commit them all at once.

git commit -a -m "your message'

Step 3: Making Changes

Git log

type git log

Displays all the changes for the repo with hashkey and timestamps and authors. This is how Git **STORES ALL YOUR COMMITS**

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 4: Parallel Development

Branching

Main branch is the master branch

It' s good practice to always work on a branch from the master rather than the master itself.

There are local branches and remote tracking branches.

Your master branch has all the prod-level code.

Say you want to add a new feature to your code, however you do not want to edit the master code. So you "branch" a copy of the master and edit that. This copy of your master now is a branch of the master.

Let's work on an example to make it clear.

On git terminal/bash type git status

It shows master in bracket

To add a new branch type git branch "name of branch"

However it shows we are still in the master branch.

To go to the new branch we have to checkout from the master branch and go to the new branch

git checkout name of branch

Now we are in the new branch

Let's go to our local repo and create a new text file with some content.

Now going back to the terminal if we hit ls (in mac) or dir in windows we see the new file as part of the directory. However if we revert to our master branch using
git checkout master and then hit ls the new text file is missing.

This works according to what we have learnt.

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 4: Parallel Development

Merging

Combining the work of various different branches onto your master is called merging.

Before Merging always make sure you are in the destination branch using git checkout

git merge branch-to-merge

Post this, any changes to the first branch will not get reflected in the master unless you merge, branches are still a separate entity.

Git pull pulls all the file from the remote repo to your local.

Git fetch does the same, but stores it in a different branch and is not connected to your remote master.

Git pull= git fetch+git merge

Time to get our hands dirty.

Step 4: Parallel Development

Rebasing

Another kind of merge.

Just places the entire commit history of your new branch at the tip of the master.

Reduces number of branches, makes the project history tree much cleaner

An example will help make this clear.

git checkout ourbranch

create a few more files in the local repo

Add them all

git add -A

git commit -m "your message"

now instead of merging, we will do a rebase

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 4: Parallel Development

Rebasing (contd)

git rebase master

Achieves the same as git merge, however the tree is more linear

Time to get our hands dirty.

Step 5: Git Push

Our final step. Now that we are all done, with the changes on our local, let's push all our changes to the remote.

We will need the access rights to the remore repo we need to push in.

We will have to add the ssh public key before pushing code. SSH Makes sure that only authorized people can push code to the remote repo.

Get the SSH key from the remote repo webpage and then go to your terminal and type

ssh-keygen

Copy the new ssh key to your remote repo webpage by clicking on new ssh key

Introduction to GIT

Git operations

Time to get our hands dirty.

Step 5: Git Push (contd)
Now that the SSH key is added to our remote repo we are ready to push

ssh -T git@github.com

This will authenticate your account and push

Refresh your webpage post this step. The black ssh key turns green.

Assuming the branch we created was called ourbranch, to push the same onto the remote repo, all we need to type is

git push origin ourbranch

Refresh the webpage and verify

We will have to git checkout to the branch we need to push before the push command

Introduction to GIT

Git operations

To revert back to a previous version

In our local repo add a text file and commit.

git checkout ourbranch

add the file in the local repo

git add -A

git commit -a -m "our final edit"

Change the text in the new file

type git commit -a -m "revert1"

Now if you want to revert to the text before the new edit was done, type

git log

which will throw up a series of commits or the commit history. copy the commit hash key of the version you want to revert back to (highlighted in yellow)

git checkout your_hash_key(first 8 chars) your_file

End of Lesson