Getting started with Git

1

Introduction to Git & Github

What is Git?

An open source multi-platform version control system originally developed by our one and only... Linus Torvalds

which means...

a practical tool that helps a software team easily manage changes to source code over time

Great because:

- Create multiple independent project versions
- Revert files to previous state
- Keep track of file modifications
- Easier debugging
- In case of FUBAR there is recovery!

All this for very little overhead!

Hosting services...

... is where all our files live.
Remote servers that store our projects and history of modifications.

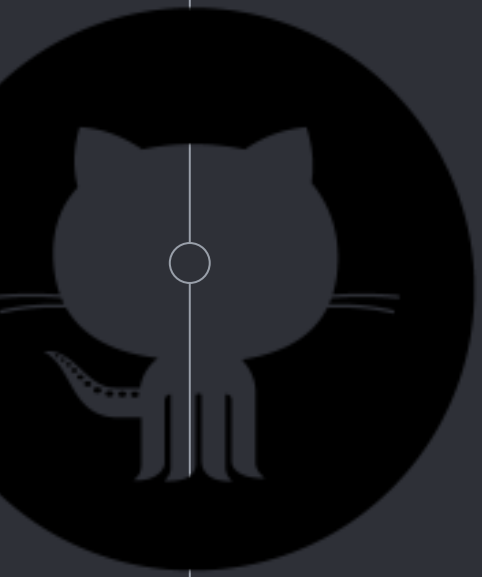
Most notable ones:

- GitHub
- BitBucket
- Sourceforge

And many more...

Github though is the most popular one.

What is GitHub?



A web-based Git repository hosting service providing all the functionality of git plus many more useful features.

Some of them are:

- ➔ Access control
- ➔ Bug tracking
- ➔ Easy to use
- ➔ Cmd + GUI version
- ➔ Central point of collaboration for millions of developers
- ➔ Free(*)

2 Learning the basics...

Practice makes Perfect

Installation / Configuration

Ubuntu / Fedora

```
(Ubuntu) $ sudo apt-get  install git-all

(Fedora) $ sudo yum install git-all

$ git config --global user.name “User Name”

$ git config --global user.email user@example.com
```

Windows

git-scm.com/download/win

Mac OS X

git-scm.com/download/mac

Create a github account at github.com

- Let's create our first project!

1) Go to Github and create a repo

OR

1) Find an existing project you might like

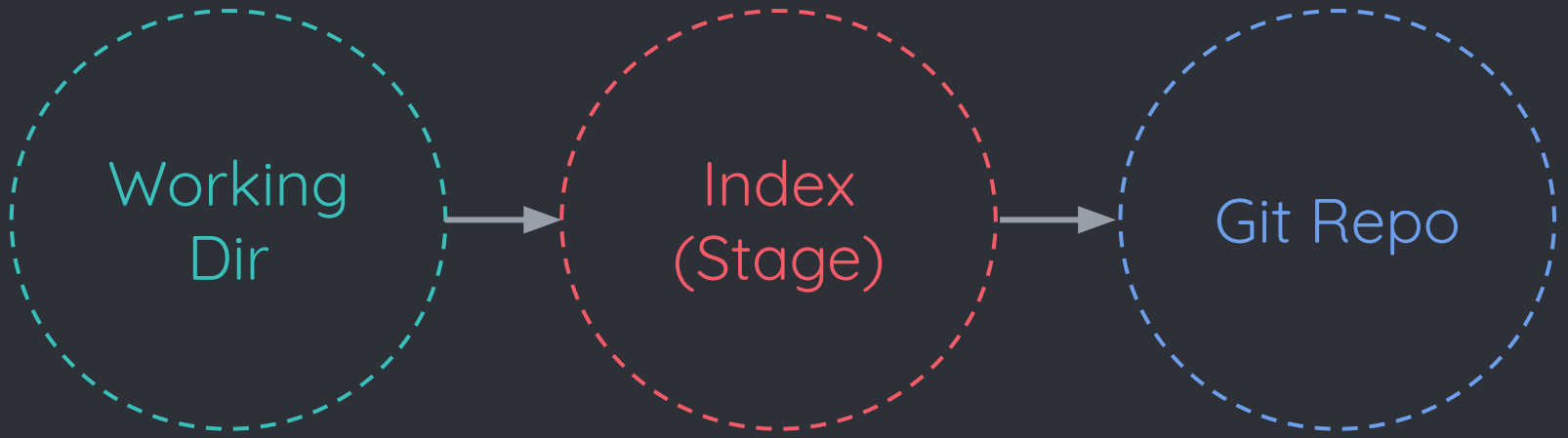
2) Copy the link of the repository

3) Connect your local directory to the remote directory of GitHub

```
~/Project $ git clone https://github.com/USER/REPO.git
```

Replace caps with the username and the repository's name

Structure of a repository



This is your ***local*** repository where your actual files are.

This place acts like a ***staging area*** just before the final commit of the files.

This is the ***remote*** directory of git and stores all the files that have been pushed here.

Check the current state of your repository with ~$ git status

Great! Lets add some files to working directory!

Move to the project folder

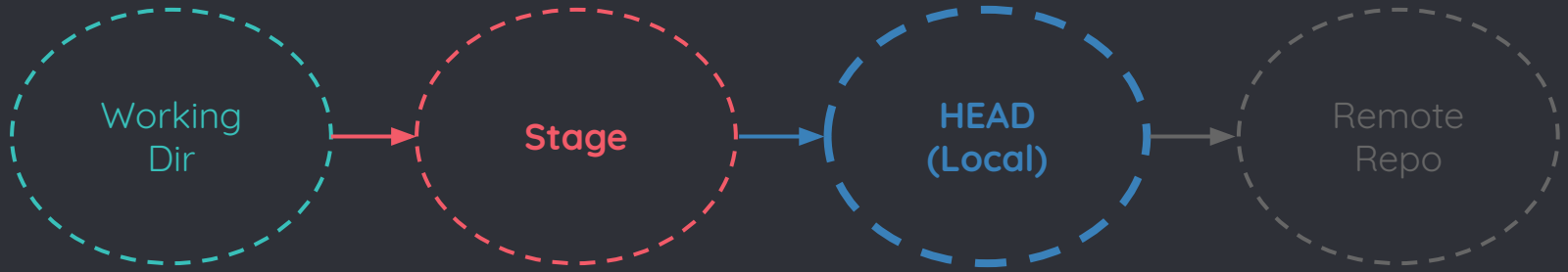
```
~$ cd Project
```

Create a file and add some text inside

```
~$ echo “Learning Git”>file.txt
```

Add the file to the stage area

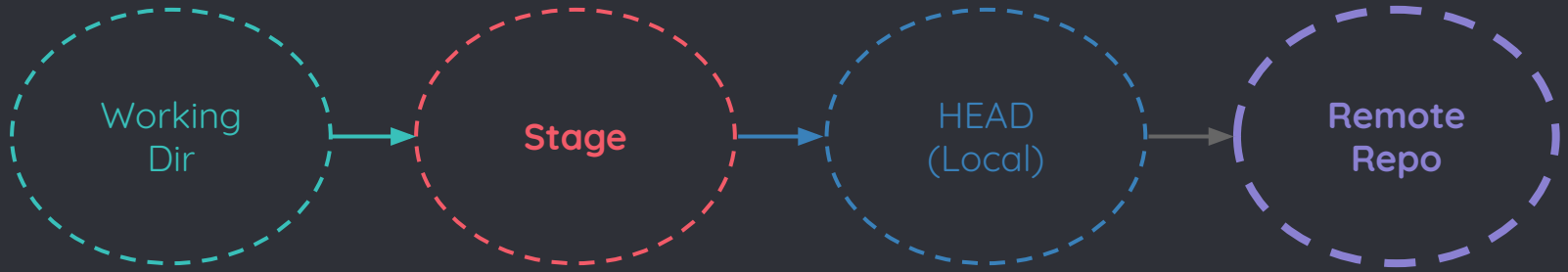
```
~$ git add file.txt
```



Now commit the files to the stage...

Commit the file to the local HEAD
-m: write a sort explicit message
avoid opening a text editor
use imperative language

```
~$ git commit -m “Add my first file”
```



and finally push them to the remote repository!

Push file to the remote git repository. "origin master" is the name of the main branch of the repository

```
~$ git push origin master
```

Fetch the new files of the repo to your local repository

```
~$ git pull origin master
```

Head & History

Each commit has a specific ID.
***HEAD** is the latest state of the repository.*
Checkout which version of your project's history you want on your local repo

~$ git checkout *ID_of_commit*

Hooray!

We have successfully created our first git repository!

Our files are now safe in the remote server of github. Any changes made in the files in our local directory will **NOT** affect the files in the github repository.

Try it!
Delete local files
~$ rm file.txt

Force the HEAD state of the repository to your local repository.
~$ git checkout HEAD -f
Everything should be back since the latest commit

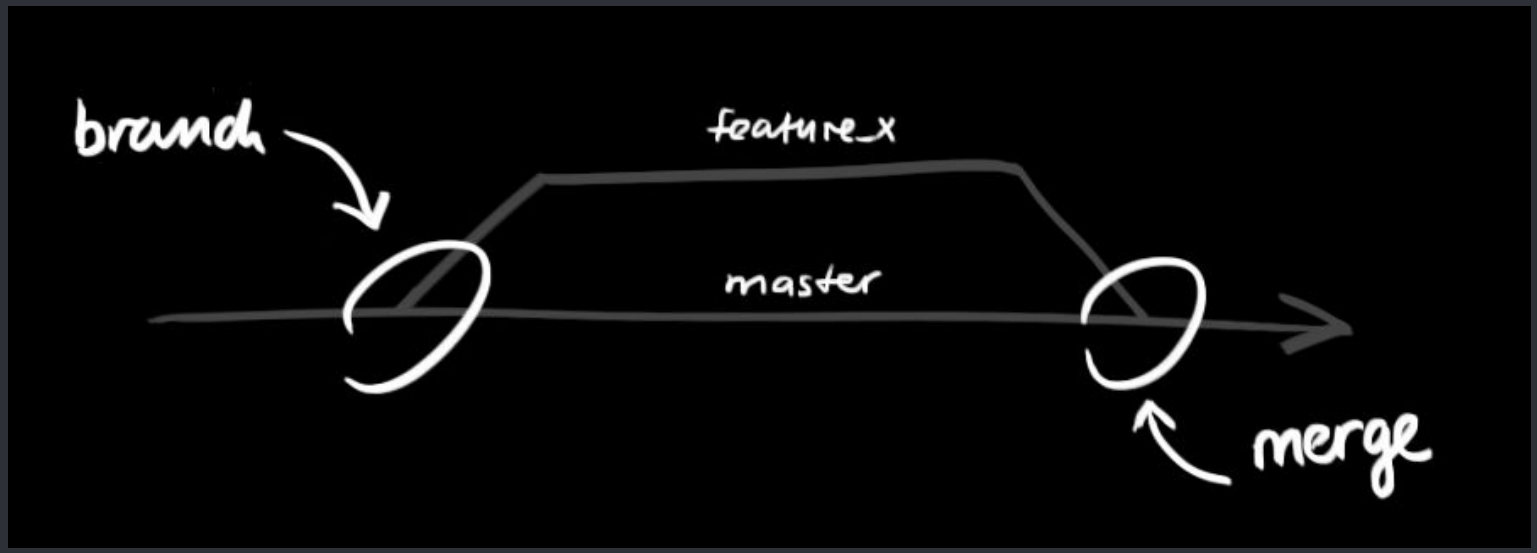
3

Let's go a bit deeper

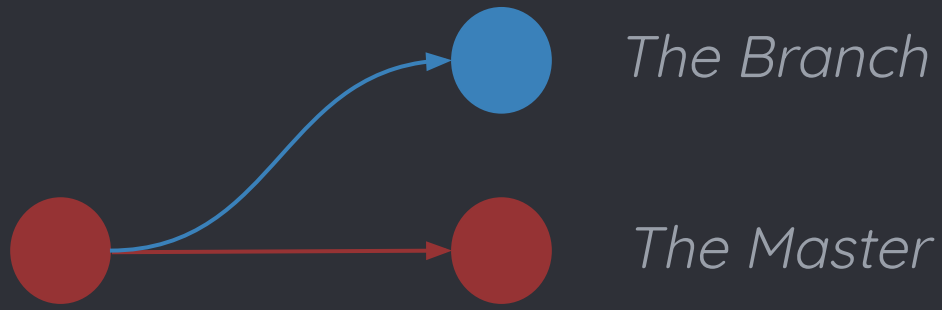
Branches & other useful stuff

Branches for the deviants

Branches are very useful when you want to create a differentiated version of the project without affecting its original form.



Creating a branch



Create a new branch in local repo
(and automatically switch in it)

```
~$ git checkout - b name_of_branch
```

Push branch to remote git repo

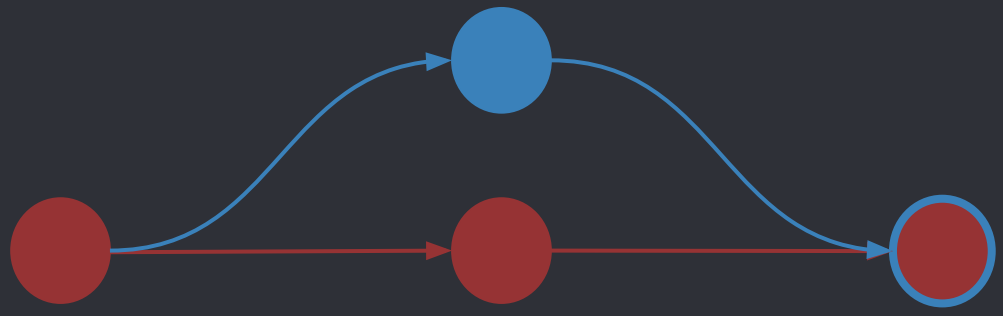
```
~$ git push origin name_of_branch
```

Show all branches and which you're currently working in

```
~$ git branch
```

You can now create files and push them as normal

The Merge



Lets merge the new branch with the master

Switch the working directory to master branch

```
~$ git checkout master
```

Merge the files of the branch to master

```
~$ git merge name_of_branch
```

Push the merge to the remote repository

```
~$ git push origin master
```

The Stash

A very common scenario...

a)There have been changes in the remote repo and we are out of date.

b)We made changes in some files and tried to push them to the remote repo with no luck.

Another common scenario

.gitignore

You have some files that you don’t want to put in your repo.

You do not want them to disturb you when you commit and push changes.

Create .gitignore, a local or global file which stands for an exception list.

Other useful commands and tricks

Delete all untracked files in the local directory

```
~$ git clean -n      Will warn you which files will delete
~$ git clean -f      Will delete those files
```

Show the history of commits

```
~$ git log
```

Show current status of working dir, stage & head

```
~$ git status
```

Delete all local changes & fetch the HEAD of the repo

```
~$ git reset --hard origin/master
```

Thanks!

Any questions?

Kostis S-Z
Marinos Poiitis

Useful links!

https://git-scm.com/docs/
https://rogerdudler.github.io/git-guide/