Gabriele **Proietti Mattia**

Advanced Operating Systems and Virtualization

[Lab 01] Git

An essential guide

Department of Computer, Control and Management Engineering "A. Ruberti", Sapienza University of Rome

gpm.name · proiettimattia@diag.uniroma1.it

A.Y. 2020/2021 · v3

What is git

Git is a version control system, free and open source and it was introduced for managing the kernel versioning.

The essential concept of git are:

- Repositories and Remotes
- Commits
- Branching and Merging
- Tags
- Forks
- Pull Requests

Initialization

For initializing your environment, after installing git, you should register a username and an email which represents the user that will make changes to the code. I suggest you to initialize a system-wide account, in this way:

```
$ git config --global user.name “John Doe”
$ git config --global user.email “email@example”
$ git config --global --list
user.name=John Doe
user.email=email@example.com
```

Repositories

Repositories represent the storage area where putting the code files. It's very important to use a .gitignore file at the root of the repository for avoiding to commit undesirable files. The general rule is that binary files should not be uploaded because for them we cannot deduce differences like in normal files. You can use https://gitignore.io for generating a standard .gitignore according to your environment.

For creating a repositories you create a folder, and in that folder you hit the command

```
$ git init .
Initialized empty Git repository in /home/gpm/Downloads/test/.git/
```

Or you can **clone** an existing repository from a remote source, for example GitHub

```
$ git clone git@github:gabrielepmattia/repository.git
```

From now on, when inside the folder you can use all the git commands (see `git help`)

Remotes

With git init you created a local repository, but if your work is shared makes sense to put the repository in the cloud, for example with GitHub. When you created the repository in the cloud, the platform gives you a remote address that can be of two types:

- `https://github.com/gabrielepmattia/repository.git`
- `git@github.com:gabrielepmattia/repository.git`

The links are for the same repository but they are different, the first uses the **http** protocol and will require you to input the GitHub username and password every time you will submit changes, the second uses the **SSH** protocol but you firstly need to generate an SSH key and to upload it to Github.

Remotes

Adding a remote

For adding a remote we use the `git remote` command, the default remote is called `origin`

```
$ git remote add origin git@github.com:gabrielepmattia/repository.git
```

Push/Pull

When you assigned a remote to your local repository you can pull new commits in local from the cloud or push new commits to the cloud. If you do not specify anything, git uses the `origin` remote and the branch `master` (we will see it in a while)

```
$ git pull
```

```
$ git push
```

Commits

A commit marks a checkpoint of your repository. When creating a commit you need to specify what has been changed, this is used not only for tracking the changes but also to revert back to a previous state when things goes bad.

When you create a commit you need to know which are the states in which a source file can be, notice that not all files that are in the repositories folder must be committed:

- **tracked** - a tracked file is a file that is present in your git repository since it has been previously committed
- **untracked** - an untracked file is file that is not present in your repository, you never committed it
- **staged** - the file is staged for commit, this means that it will take part of the next commit
- unstaged - the file is unstaged for commit, this means that il will NOT take part to the next commit
- **modified/deleted/renamed** - these are usually the changes that will be committed

Commits

The flow for committing is:

1. **Pull** and **merge** new changes from the (default) remote, with command

```
git pull
```

2. **Track** the file, if untracked with command

```
git add <filename>
```

3. **Stage** the files, if unstaged with command

```
git add <filename>
```

4. **Create** the commit, with command

```
git commit -m <description of commit>
```

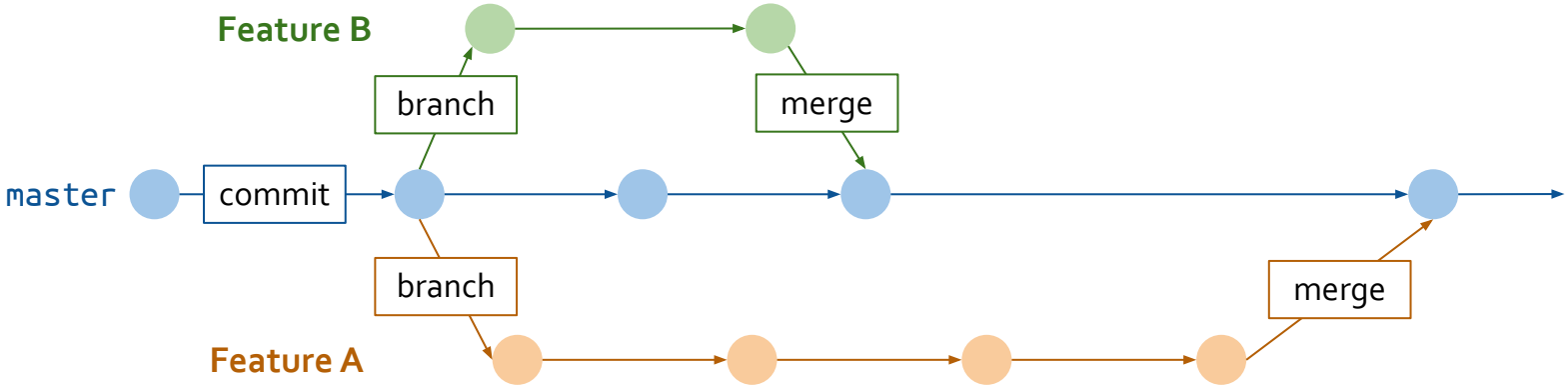
5. **Push** the changes to the (default) remote, with command

```
git push
```

6. Goto 1

Branches

Every commit that you can create must be bound to a branch. A branch is a logical line of commits. In general you can use only a single branch, that is the master branch but when introducing new features you create a new branch and then merge (especially if you are working in group).



Branches

Creating a branch

For creating or switching from a branch to another you can use the git checkout command.

1. Check the available branches with `git branch`

```
$ git branch
* master
```

2. Switch to an existing branch with `git checkout`

```
$ git checkout master
```

3. Otherwise Create a new branch, again with `git checkout`

```
$ git checkout -b new_feature
Switched to a new branch new_feature'
```

Branches

For merging `new_feature` into `master`

1. Check the current situation with `git status`, check if you **are on `master`** branch
2. Use the command `git merge`

```
$ git merge new_feature
```

Tags

Tags are generally used to mark software releases. Tags are always associated a commit id and a message. You can create and manage tags with command `git tag`.

- `git tag -a "v1.0.0" abcdef` - **creates the tag** v1.0.0 linked to the commit id abcdef (every commit, branch and tag have a unique id, like a hash)
- `git tag -d "v1.0.0"` - **deletes** the tag v1.0.0
- `git push --tags` - tags needs to be manually **pushed** to the remote repository

Pull Requests

The general flow of operation when collaborating to open source project is to:

1. **Forking** the project, that means copying the repository as is to your account
2. Do the modifications by committing them
3. Open a **Pull Request** to the original repository, asking the owner(s) to pull your new commit and merge them

Disclaimer

Using git can **mess up with your source code if badly used**, therefore, especially if it is the first time that you use it, I suggest you to **keep always** a backup of everything by always pushing the repository to GitHub, protecting branches from being overwritten or by making zips of the entire folder in your local pc.

GitHub

GitHub

GitHub is a cloud solution that will host your repositories for free.

We will use **GitHub Classroom** for the project. Since you already joined in Google Classroom with your Sapienza account, if you join GitHub with the same email address you will automatically linked to the class (in GitHub you can have multiple mail address so you are not obliged to use the sapienza one indefinitely just add as many email addresses as you want and choose carefully the username).

The link to the GitHub classroom is

https://classroom.github.com/classrooms/77753448-advanced-operating-systems-and-virtualization-2020-2021

If not working, try to join accept this "Test Assignment"

https://classroom.github.com/a/eYPGhzME

Advanced Operating Systems and Virtualization

[Lab 01] Git

LECTURER

Gabriele **Proietti Mattia**

gpm.name · proiettimattia@diag.uniroma1.it

DIAG