Git

- Level: Novice
- By Guillaume
- Weight: 1
- Your score will be updated as you progress.

Concepts

For this project, students are expected to look at these concepts:

- Source code management
- Git and Github cheat sheet - Everything in less than 30 seconds
- The Framework
- Approaching a Project

Resources

Read or watch:

- Resources to learn Git
- About READMEs
- How to write a Git commit message

Resources for advanced tasks (Read only after finishing the mandatory tasks):

- Learning branching
- Effective pull requests and other good practices for teams using GitHub

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What is source code management
- What is Git
- What is GitHub
- What is the difference between Git and GitHub
- How to create a repository
- What is a README
- How to write good READMEs
- How to commit
- How to write helpful commit messages
- How to push code
- How to pull updates

- How to create a branch
- How to merge branches
- How to work as collaborators on a project
- Which files should and which files should not appear in your repo

Requirements

General

- A `README.md` file at the root of the repo, containing a description of the repository
- A `README.md` file, at the root of the folder of *this* project (i.e. `git`), describing what this project is about
- **Do not use GitHub's web UI**, but the command line to perform the exercise (except for operations that can not possibly be done any other way than through the web UI). You won't be able to perform many of the task requirements on the web UI, and you should start getting used to the command line for simple tasks because many complex tasks can only be done via the command line.
- Your answer files should only contain the command, and nothing else

More Info

Install `git`

If `git` is not already installed on your terminal:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install git
```

Basic usage

At the end of this project you should be able to reproduce and understand these command lines:

```
$ git clone <repo>
$ touch test
$ git add test
$ git commit -m "Initial commit"
$ git push origin main
```

Quiz questions
Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Create and setup your Git and GitHub account

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Step 0 - Create an account on GitHub

You will need a GitHub account for all your projects. You can create an account for free here

Step 1 - Create a Personal Access Token on Github

To have access to your repositories and authenticate yourself, you need to create a Personal Access Token on Github.

You can follow this tutorial to create a token.

Once it's created, you should have a token that looks like this:

Step 2 - Update your profile on the Intranet

Update your Intranet profile by adding your Github username here

If it's not done **the Checker won't be able to correct your work**

Step 3 - Create your first repository

Using the graphic interface on the github website, create your first repository.

- Name: Look at the bottom of the project to see the name of the repository
- Description: `This is my first repository as a full-stack engineer`
- Public repo
- No `README`, `.gitignore`, or license

Step 4 - Open the sandbox

On the intranet, just under the task, click on the button and `run` to start the machine.

Once the container is started, click on to open a shell where you can start work from.

Step 5 - Clone your repository

On the webterm of the sandbox, do the following:

- Clone your repository

```
root@896cf839cf9a:/# git clone https://{YOUR_PERSONAL_TOKEN}@github.com/{YOUR_USERNAME}/{YOUR_REPO}.git

Cloning into '{YOUR_REPO}'...
```

```
warning: You appear to have cloned an empty repository.
```

Replace {YOUR_PERSONAL_TOKEN} with your token from step 1

Replace {YOUR_USERNAME} with your username from step 0 and 1

Replace {YOUR_REPO} with the name of the reposotiry at the bottom of the task

Step 6 - Create the README.md and push the modifications

- Navigate to this new directory. Tips

```
root@896cf839cf9a:/# cd {YOUR_REPO}/
root@896cf839cf9a:/{YOUR_REPO}#
```

-Create the file `README.md` with the content `My first readme`. Tips

```
root@896cf839cf9a:/{YOUR_REPO}# echo 'My first readme' > README.md

root@896cf839cf9a:/{YOUR_REPO}# cat README.md

My first readme
```

- Add this new file to git, commit the change with this message “My first commit” and push to the remote server / origin

```
root@896cf839cf9a:/{YOUR_REPO}# git add .
root@896cf839cf9a:/{YOUR_REPO}# git commit –m 'My first commit'
[master (root-commit) 98eef93] My first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
root@896cf839cf9a:/{YOUR_REPO}# git push

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 212 bytes | 212.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/{YOUR_USERNAME}/{YOUR_REPO}.git

 * [new branch]      master -> master
```

Good job!

You pushed your first file in your **first repository**.

You can now check your repository on GitHub to see if everything is good.

Repo:

- GitHub repository: `zero_day`
- File: `README.md`

Review your work Get a sandbox QA Review
5/5 pts
1. Repo-session
mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Create a new directory called `git` in your `zero_day` repo.

Make sure you include a not empty `README.md` in your directory:

- at the root of your repository `zero_day`
- AND in the directory `git`

And important part: **Make sure your commit and push your code to Github - otherwise the Checker will always fail.**

Repo:

- GitHub repository: `zero_day`

Review your work Get a sandbox QA Review
5/5 pts
2. Coding fury road
mandatory

Score: 100.00% (*Checks completed: 100.00%*)

For the moment we have an empty project directory containing only a `README.md`. It's time to code!

- Create these directories at the root of your project: `bash`, `c`, `js`
- Create these empty files:
  - `c/c_is_fun.c`
  - `js/main.js`
  - `js/index.js`
- Create a file `bash/best` with these two lines inside: `#!/bin/bash` and `echo "Best"`
- Create a file `bash/school` with these two lines inside: `#!/bin/bash` and `echo "School"`
- Add all these new files to git
- Commit your changes (message: "Starting to code today, so cool") and push to the remote server

Repo:

- GitHub repository: `zero_day`
- Directory: `git`
- File: `bash/best, bash/school, c/c_is_fun.c, js/main.js, js/index.js`

Review your work Get a sandbox QA Review
5/5 pts

3. Collaboration is the base of a company

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

A branch is like a copy of your project. It's used mainly for:

- adding a feature in development
- collaborating on the same project with other developers
- not breaking your entire repository
- not upsetting your co-workers

The purpose of a branch is to isolate your work from the main code base of your project and/or from your co-workers' work.

For this project, create a branch `update_script` and in this branch:

- Create an empty file named `bash/98`
- Update `bash/best` by replacing `echo "Best"` with `echo "Best School"`
- Update `bash/school` by replacing `echo "School"` with `echo "The school is open!"`
- Add and commit these changes (message: "My personal work")
- Push this new branch Tips

Perfect! You did an amazing update in your project and it's isolated correctly from the **main** branch.

Ho wait, your manager needs a quick fix in your project and it needs to be deployed now:

- Change branch to `main`
- Update the file `bash/best` by replacing `echo "Best"` with `echo "This School is so cool!"`
- Delete the directory `js`
- Commit your changes (message: "Hot fix") and push to the origin

Ouf, hot fix is done!

Repo:

- GitHub repository: `zero_day`
- Directory: `git`
- File: `bash/best, bash/school, bash/98`

Review your work Get a sandbox QA Review
5/5 pts

4. Collaboration: be up to date

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Of course, you can also work on the same branch as your co-workers and it's best if you keep up to date with their changes.

For this task – **and only for this task** – please update your file `README.md` in the main branch from GitHub.com. It's the **only time** you are allowed to update and commit from GitHub interface.

After you have done that, in your terminal:

- Get all changes of the main branch locally (i.e. your `README.md` file will be updated)
- Create a new file `up_to_date` at the root of your directory and in it, write the git command line used
- Add `up_to_date` to git, commit (message: "How to be up to date in git"), and push to the origin

Repo:

- GitHub repository: `zero_day`
- Directory: `git`
- File: `README.md, up_to_date`

Review your work Get a sandbox QA Review

5/5 pts

5. HAAA what did you do???

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Collaboration is cool, but not really when you update the same file at the same time…

To illustrate that, please merge the branch `update_script` to `main`: "Cool, all my changes will be now part of the main branch, ready to be deployed!"

HHHHHHHAAAAAAAA

```
CONFLICT (content): Merge conflict in bash/best
```

As you can see, you have conflicts between two branches on the same file.

Your goal now is to resolve conflicts by using the version of the branch `update_script`, and push the result to the origin.

At the end, you should have all your work from the branch `update_script` (new file and two updated files) and all latest `main` commits (new files, delete folder, etc.), *without* conflicts.

Repo:

- GitHub repository: `zero_day`
- Directory: `git`

Review your work Get a sandbox QA Review

6/6 pts

6. Never push too much

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Create a `.gitignore` file and define a rule to never push `~` files (generated by Emacs). Tips

Repo:

- GitHub repository: `zero_day`
- Directory: `git`
- File: `.gitignore`

Review your work Get a sandbox QA Review

5/5 pts

Score

200%

Score

Congratulations! You made it!

Next project: Shell, basics

Open the next project