Git - Version Control

Motivation for version control

What is it ?

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later (getting started with version control).
- Git is a version control system.

Why do I need it ?

- You're **always** working in a team:
  - Share code with your **coworkers**.
  - **Future you** will not remember why **past you** made that change!

- Save well and save often:
  - Reduces anxiety
  - Keeps a clean top level…
  - ...but retains entire history!

- Side benefit: easy code sharing

CAMBRIDGE SPARK

Installing Git

In a terminal (recommended for now)

- MacOS and Linux
  - Available by default in terminal
- Windows
  - Use Anaconda Prompt
  - (Alternatively you can use GitBash)

Not in a terminal

- Most IDEs have a git option (Pycharm, VSCode, Atom, …)
  - Check slide for git interface in VSCode.
- Check documentation GitKraken

CAMBRIDGE SPARK

Git’s structure

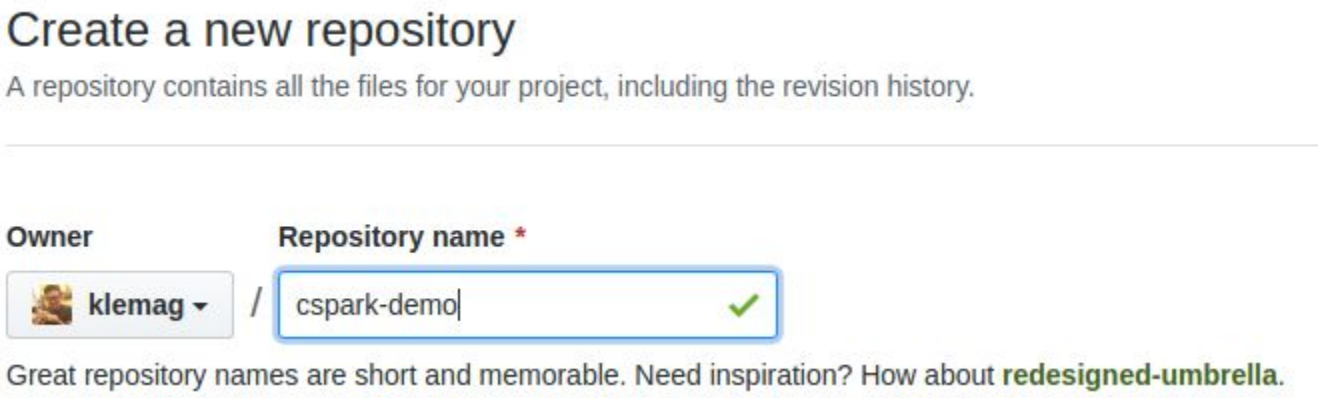
Working Directory (local)	Staging Area (local)	Git Directory (local)	Remote Repository (on a server)
Your **local** files, ready to be edited.	Where you move your files **after modifying** them.	Where you "**commit**" your changes. That creates a **new version** of the file locally.	"**Push**" your changes to a **server** where it will be **saved**. Common providers: • Github • Gitlab • BitBucket • Cspark’s Gitlab

https://git-scm.com/doc

CAMBRIDGE SPARK

Create a new repository

On **Github**:



CAMBRIDGE SPARK

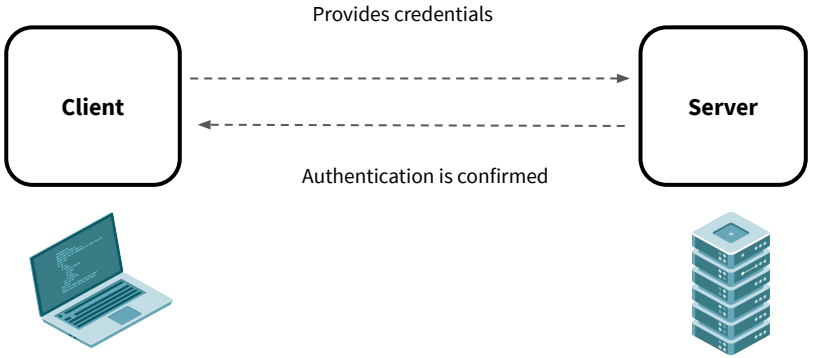
User authentication

SSH

- Protocol that uses encryption to secure the connection between a **client** and a **server**.
- Authentication works with a **private** and a **public** key that need to be generated.
- The private key must be kept confidential so that only the client knows it.
- The public key can be shared with any server.

HTTPS

- Authentication protocol that is used between a **client** and a **server**.
- Client provides authentication information (username, password) and server verifies it.

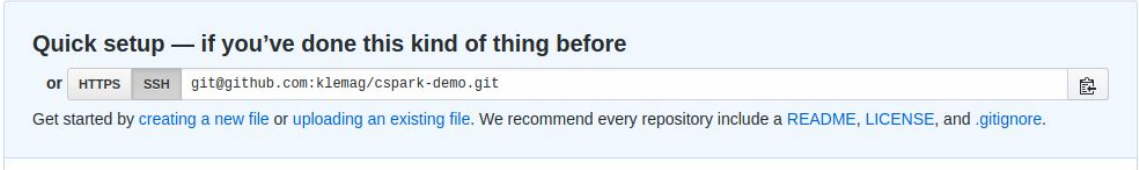


CAMBRIDGE SPARK

Clone your repository

⇒ Make a copy of the remote repository to your local machine.

On **Github,** first copy the URL provided



You can clone using one of the two protocols:

- **HTTPS**: No setup required but you will be prompt for your password.
- **SSH**: Need to add a key once, then authentication is automated.
  - For SSH authentication you will need to generate a public/private key pair (instructions)

Advantages of SSH

- More secure.
- No need to remember long/complicated passwords.

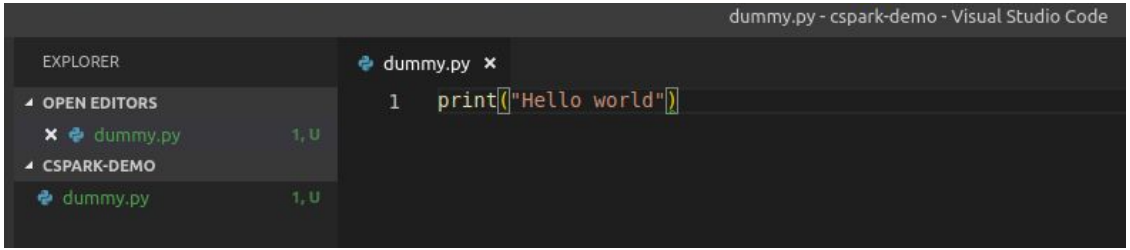
CAMBRIDGE SPARK

Clone your repository

```
$ cd where/you/want/to/clone/the/repo
$ git clone the/url/you/copied
$ cd cspark-demo
```

Once your repository is cloned, you can start working as you would do in any normal directory:

- Open your IDE and create a new file **`dummy.py`**
- Write some code and save the file.



CAMBRIDGE SPARK

Stage your changes

```
$ git status # to see files in working dir and staging area
$ git add dummy.py # to move dummy.py to staging area
```

```
[master]kevin:~/tmp/ads1/cspark-demo$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        dummy.py

nothing added to commit but untracked files present (use "git add" to track)
```

Your new file is shown as untracked change

`git add dummy.py` →

```
[master]kevin:~/tmp/ads1/cspark-demo$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   dummy.py
```

Your new file was added to the staging area

CAMBRIDGE SPARK

Commit your staged files

```
$ git commit -m "your message here"
$ git status # nothing new in working dir. and staging area
$ git log #see all commits
```

```
[master]kevin:~/tmp/ads1/cspark-demo$ git commit -m "add file"
[master (root-commit) 74b70e5] add file
 1 file changed, 1 insertion(+)
 create mode 100644 dummy.py
```

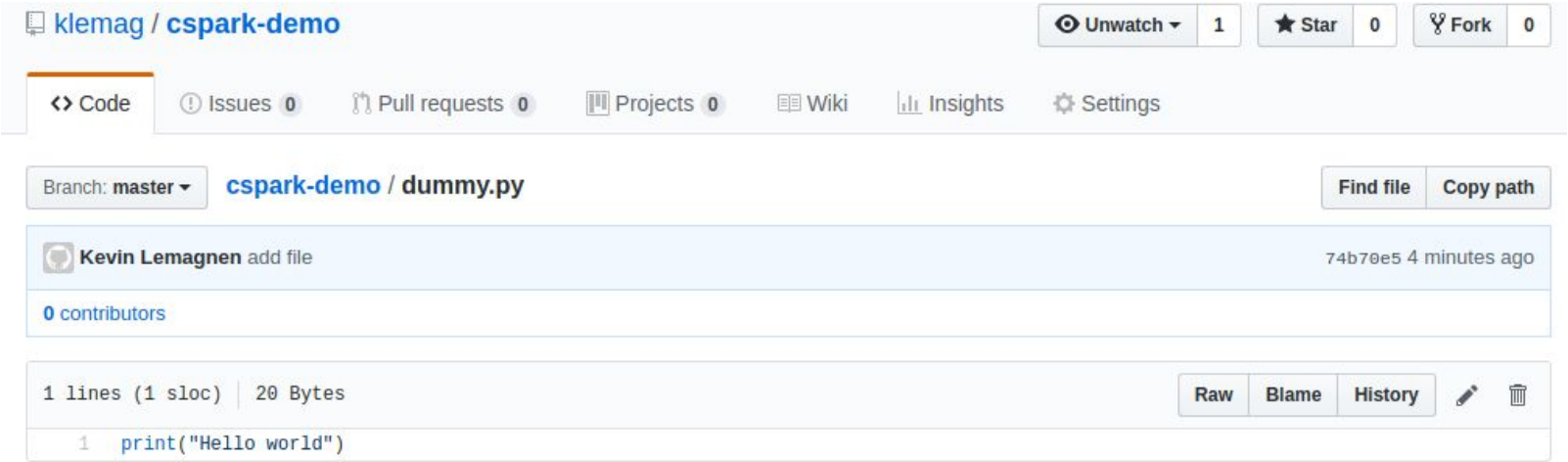
```
[master]kevin:~/tmp/ads1/cspark-demo$ git log
commit 74b70e561752819540c5ccbde0c1431520145a54 (HEAD -> master)
Author: Kevin Lemagnen <kevin@cambridgespark.com>
Date:   Fri Feb 1 14:27:48 2019 +0000

    add file
```

CAMBRIDGE SPARK

Push to the server

```
$ git push #push all your new commits to the server
```



CAMBRIDGE SPARK

Branches

In most repos, the default branch is called "**master**".

A common flow:

1. Keep master as the **stable** branch.
2. Create a **new branch** to develop a new feature.
3. **`merge`** to master when the code is stable.

```
# create new branch from current one
    $ git checkout -b new_branch_name
# checkout existing branch
    $ git checkout existing_branch_name
```

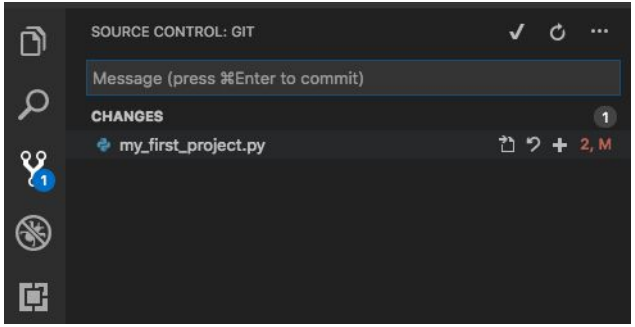
CAMBRIDGE SPARK

Terminology

- **Repository**: the online version of your git directory.
- **Fork**: a copy of someone else's repository.
- **Branch**: different versions of your repository.
- **Pull Request/Merge Request**: action to merge a branch into another
  - For instance if you finish working on a new branch and want to update your main branch with your changes.
- **Conflicts**: happens when multiple people work on multiple branch...
  - If coworkers make different changes on the same file that aren't compatible, it can lead to a conflict.

CAMBRIDGE SPARK

VSCode & git



VSCode provides a very good git interface for version control.

- Once you're ready to make a submission, click on the Git icon to the top left.
- This will show a list of the changes and all file modifications.
- Hovering on a file provides options to Open File, Discard Changes, or Stage Changes.
- To submit a file, click on "Stage Changes" on that file.
- Next, VS Code requires a commit message to be entered (useful when working on more complex/collaborative projects; for now, anything brief is fine)
- Click the tick icon to commit the code to your local master branch.
- Finally, push the updated local branch to the remote repository.
    - To do so, click on the up and down arrow icon pair to the right of the branch name to synchronise changes.

CAMBRIDGE SPARK

Demo with GitLab

GitLab

CAMBRIDGE SPARK

Appendix - git configuration

```
$ git --version
$ git config --list
$ git config --global user.name "csparkGenius"
$ git config --global user.email "csg@cambridgespark.com"
$ git config --global core.editor vim
```

> https://help.github.com/articles/associating-text-editors-with-git/

```
$ git config --list
```

CAMBRIDGE SPARK

Exercise

- Under your existing `cspark-demo` repository, create a new empty folder called **`git_exercise`**
  - Use `git status` and `git diff` to see the status of your repository.
- `Stage` , `commit` your changes and then `push` them to the remote repository.
  - Make sure your commit messages are clear, short and self explanatory.
- Now, go to github's UI and make a few changes to your **`dummy.py`** file.
  - Modify it so that instead of `print('Hello world')` it contains a function named **`hello`** that takes a `student_name` string and returns a `'Hello <student_name>'` string message (similar the one you were asked to implement as part of KATE's first python project).

CAMBRIDGE SPARK

Exercise (cont'd)

- Your local branch does not have the latest changes you've made so from your terminal `git pull` the remote repository changes.
- Checkout a new branch of your repository locally and give it any name like **`my-branch`**
- **`dummy.py`** is not an appropriate name anymore so let's rename the file to **`hello_student.py`**.
  - Use `git status` and `git diff` again to see the status of your repository.
- `Stage` , `commit` your changes and then `push` them to the remote repository.
- Go to github and create a merge request from your branch to the `master` branch.
- Then `merge` it from github's ui and go to your terminal and `pull` the latest changes.
- Bonus: Why not try to revert all the changes you've made so far and repeat the exercise in the VSCode Git interface ?

CAMBRIDGE SPARK