



Ersilia



H3D Foundation and Ersilia present

Bringing data science and AI/ML tools to
infectious disease research

Event Sponsors



CS&S

Code for
Science
& Society





Session 4: Git version control and GitHub

Skills Development

Gemma Turon, @TuronGemma, gemma@ersilia.io

Miquel Duran-Frigola, @mduranfrigola, miquel@ersilia.io

Ersilia Open Source Initiative, @ersiliaio, <https://ersilia.io>

30th September 2022



What is Git and GitHub?



Git is a software that tracks changes to files. Changes include creating new files, deleting files or folders or editing the content of a file.

When can Git be useful?



GitHub is an internet hosting service for software development and version control using Git.

When can GitHub be useful?



Installing Git

<https://git-scm.com/downloads>

Windows set up:

- Adjusting your PATH environment: select *Run Git from the Windows Command Prompt*
- Configuring line ending: select *Checkout Windows-style, commit Unix-style line endings*

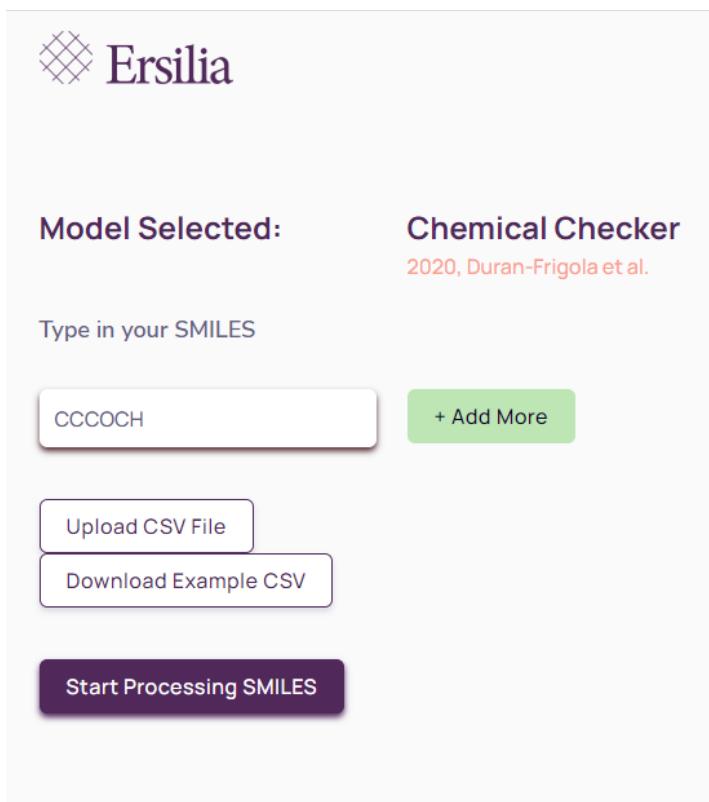
Linux: it is usually pre-installed

Mac: download and install following instructions



CLI vs GUI

Graphical user interface: clicking with the mouse via interactive menus



The screenshot shows the Ersilia web application interface. At the top left is the Ersilia logo. Below it, the text "Model Selected:" is followed by "Chemical Checker" and "2020, Duran-Frigola et al.". A text input field contains the SMILES string "CCCOCH". To the right of the input field is a green button labeled "+ Add More". Below the input field are two buttons: "Upload CSV File" and "Download Example CSV". At the bottom is a dark purple button labeled "Start Processing SMILES".



The screenshot shows a terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top. The terminal displays the following text:

```
#using ersilia through the CLI
ersilia fetch eos4e40
ersilia serve eos4e40
ersilia api predict -i "CCCCC"
ersilia close eos4e40
ersilia delete eos4e40
```

Git does not have a native GUI

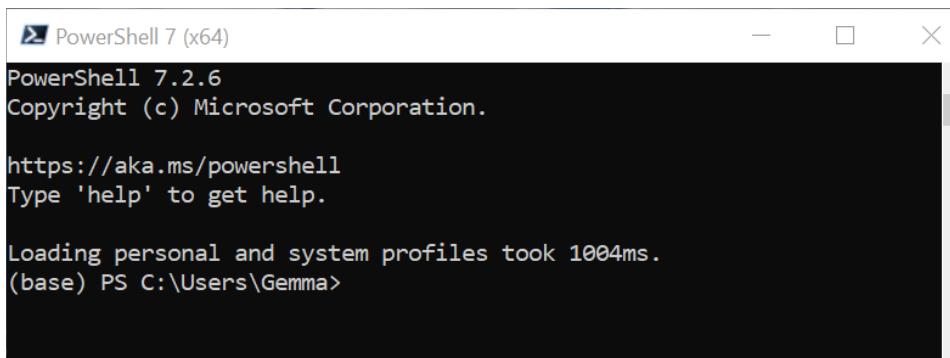


Command Line Interface

Send commands in the form of lines of text to the computer, to run programs, manage computer files and interact with the computer.

The language of the command varies between systems:

- Command Prompt (CMD): Windows
- Bash: Linux, Mac



```
PowerShell 7 (x64)
PowerShell 7.2.6
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

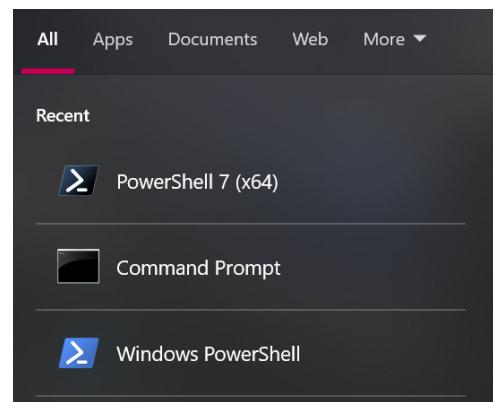
Loading personal and system profiles took 1004ms.
(base) PS C:\Users\Gemma>
```



Command Line Interface

Open your Command Line Interface:

- Linux: terminal
- Mac: terminal
- Windows:
 - Command Prompt ✖
 - Windows PowerShell (cmdlet) ✓
 - PowerShell 7 (cmdlet) ✓



**tip: use the search function to find the CLI*



Basic CLI commands



```
#move from the root to a directory
(base) PS C:\Users\Gemma> cd Desktop
(base) PS C:\Users\Gemma\Desktop>

#create a new directory (folder)
(base) PS C:\Users\Gemma\Desktop> mkdir test
```

```
Directory: C:\Users\Gemma\Desktop
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	21/09/2022 14:16		test

```
#enter the new folder we have created
(base) PS C:\Users\Gemma\Desktop> cd test
(base) PS C:\Users\Gemma\Desktop\test>
```

cd: move to a directory

mkdir: make a new directory



Basic CLI commands

```
● ● ●

#we can go back with ..
(base) PS C:\Users\Gemma\Desktop\test> cd ..
(base) PS C:\Users\Gemma\Desktop>
(base) PS C:\Users\Gemma\Desktop> cd test
(base) PS C:\Users\Gemma\Desktop\test>

#list files in a folder
(base) PS C:\Users\Gemma\Desktop\test>ls
(base) PS C:\Users\Gemma\Desktop\test>

#create a new file inside the folder
(base) PS C:\Users\Gemma\Desktop\test> notepad example.txt
(base) PS C:\Users\Gemma\Desktop\test> ls

    Directory: C:\Users\Gemma\Desktop\test

Mode                LastWriteTime         Length Name
----                -----          ---- -  
-a---       21/09/2022     14:24            7 example.txt
```

cd .. : to move back

ls: list files in a folder

notepad: creates file (PS7)

nano: creates file (bash)



Basic CLI commands



```
#we can make a new sub directory and move the file there
(base) PS C:\Users\Gemma\Desktop\test> mkdir subfolder

(base) PS C:\Users\Gemma\Desktop\test> ls

Directory: C:\Users\Gemma\Desktop\test

Mode                LastWriteTime         Length Name
----                -----              ----- 
d----        21/09/2022     14:42            subfolder
-a---        21/09/2022     14:24           7 example.txt

(base) PS C:\Users\Gemma\Desktop\test> mv example.txt subfolder

(base) PS C:\Users\Gemma\Desktop\test> cd subfolder
(base) PS C:\Users\Gemma\Desktop\test\subfolder> ls

Directory: C:\Users\Gemma\Desktop\test\subfolder

Mode                LastWriteTime         Length Name
----                -----              ----- 
-a---        21/09/2022     14:24           7 example.txt
```

mv: moves a file



Basic CLI commands



```
#copy a file instead of moving it
(base) PS C:\Users\Gemma\Desktop\test\subfolder> cp example.txt ../
(base) PS C:\Users\Gemma\Desktop\test\subfolder> ls

    Directory: C:\Users\Gemma\Desktop\test\subfolder

Mode                LastWriteTime         Length Name
----                -----          ---- - -
-a---        21/09/2022      14:24            7 example.txt

(base) PS C:\Users\Gemma\Desktop\test\subfolder> cd ..
(base) PS C:\Users\Gemma\Desktop\test> ls

    Directory: C:\Users\Gemma\Desktop\test

Mode                LastWriteTime         Length Name
----                -----          ---- - -
d---        21/09/2022      14:42            subfolder
-a---        21/09/2022      14:24            7 example.txt

(base) PS C:\Users\Gemma\Desktop\test>
```

cp: copies a file



Basic CLI commands



```
#remove a file instead of moving it
(base) PS C:\Users\Gemma\Desktop\test> ls

    Directory: C:\Users\Gemma\Desktop\test

Mode                LastWriteTime         Length Name
----                -----              ----- 
d----               21/09/2022      14:42      subfolder
-a---              21/09/2022      14:24       7 example.txt

(base) PS C:\Users\Gemma\Desktop\test> rm example.txt
(base) PS C:\Users\Gemma\Desktop\test> ls

    Directory: C:\Users\Gemma\Desktop\test

Mode                LastWriteTime         Length Name
----                -----              ----- 
d----               21/09/2022      14:42      subfolder

(base) PS C:\Users\Gemma\Desktop\test> cd ..
(base) PS C:\Users\Gemma\Desktop> rm -r test
```

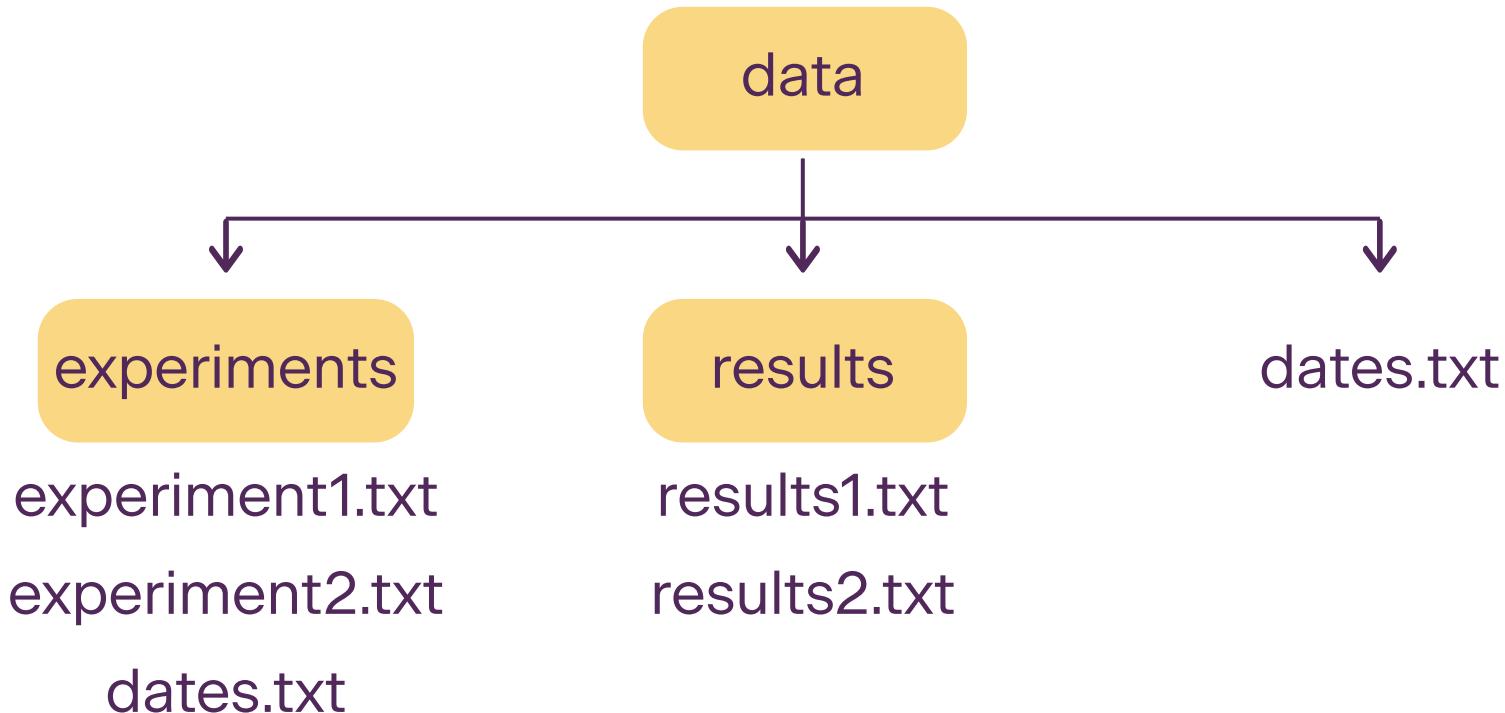
rm: removes a file

rm -r: removes a directory



A small exercise:

Using the Command Line create the following directory





Let's go back to Git

After setting up Git in the CLI, we will need to convert our new folder into a Git-tracked folder using the `git init` command

To track and make changes in Git, there are three basic commands:

- `git status`: check which files are being tracked by Git and which ones not
- `git add`: once a change has been made, you add it to the list of things to track with Git
- `git commit`: indicate which changes in the staging area you want to definitely move to the Git control



Set up Git in the CLI



```
# First, let's check we can use Git in the CLI

(base) PS C:\Users\Gemma\Desktop> git --version
git version 2.31.1.windows.1
(base) PS C:\Users\Gemma\Desktop>

#now, let's set our username (if you have a GitHub user, keep the same)
git config --global user.name "Gemma Turon"
git config --global user.email "gemma@example.com"
```

we type `git` on the CLI to indicate this is a git command
`git -version` which version of git we have installed
`git config` changes basic git settings on your system



Use Git



```
#Let's go into our folder
(base) PS C:\Users\Gemma\Desktop> cd data
(base) PS C:\Users\Gemma\Desktop\data>

#Now, we use the git init command to initialize it
(base) PS C:\Users\Gemma\Desktop\data> git init
Initialized empty Git repository in C:/Users/Gemma/Desktop/data/.git/

#let's see what we have at the moment:
(base) PS C:\Users\Gemma\Desktop\data> git status
On branch master

No commits yet

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

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

git init converts a folder into a Git tracked folder
git status checks which files are tracked by Git



Use Git



```
#Add files to git

#we can add the file one by one:
(base) PS C:\Users\Gemma\Desktop\data> git add dates.txt
(base) PS C:\Users\Gemma\Desktop\data> git status
On branch master

No commits yet

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

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

git add <filename> to add a file to the staging area



Use Git



```
#Add files to git

#we can add all at once using .
(base) PS C:\Users\Gemma\Desktop\data> git add .
(base) PS C:\Users\Gemma\Desktop\data> git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   dates.txt
  new file:   experiments/dates.txt
  new file:   experiments/experiment1.txt
  new file:   experiments/experiment2.txt
  new file:   results/results1.txt
  new file:   results/results2.txt

#we are now ready to commit those changes
```

git add . to add all files to the staging area



Use Git



```
#Add files to git

#we are now ready to commit those changes
(base) PS C:\Users\Gemma\Desktop\data> git commit -m "first commit"
[master (root-commit) 0289303] first commit
 6 files changed, 8 insertions(+)
 create mode 100644 dates.txt
 create mode 100644 experiments/dates.txt
 create mode 100644 experiments/experiment1.txt
 create mode 100644 experiments/experiment2.txt
 create mode 100644 results/results1.txt
 create mode 100644 results/results2.txt

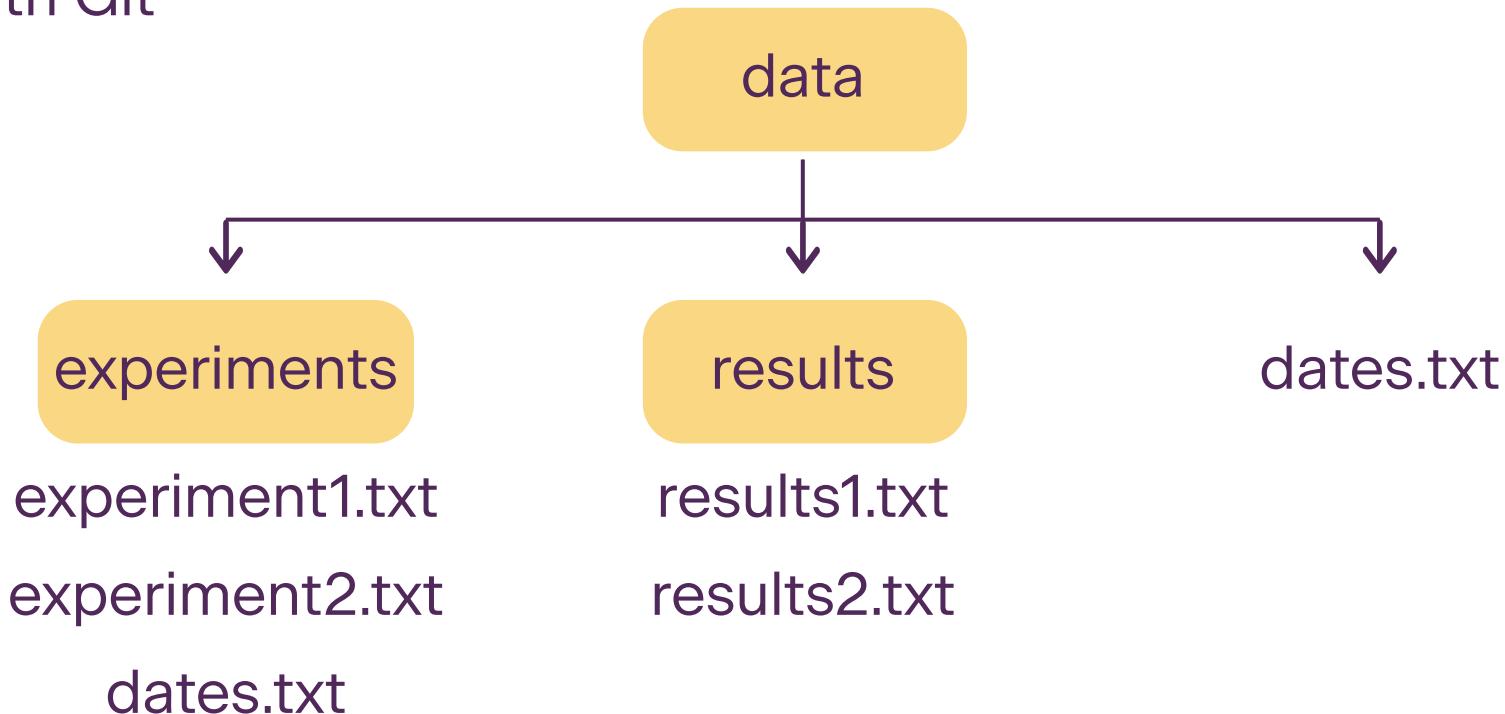
(base) PS C:\Users\Gemma\Desktop\data> git status
On branch master
nothing to commit, working tree clean
```

git commit to save changes in git (-m allows you to name the commit to know what is it about)



A small exercise:

Introduce some changes in your files and try to track them with Git





GitHub

"A code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere."



Create a GitHub account

<https://github.com/join>

A screenshot of the GitHub navigation bar. It includes the GitHub logo, links for Product, Solutions, Open Source, and Pricing, a search bar with a magnifying glass icon, and a Sign in button.

Join GitHub First, let's create your user account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)

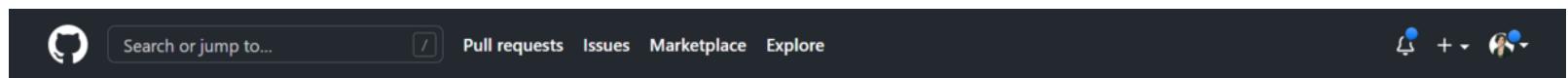
Email preferences

Send me occasional product updates, announcements, and offers.



Create a new repository

Select the New icon on the left hand-side of your profile 



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

[No template ▾](#)

Owner *

 GemmaTuron ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [potential-train](#)?

Description (optional)



Create a new repository



Select the New icon on the left hand-side of your profile

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

You are creating a public repository in your personal account.

Create repository

Private repositories are only accessible by invite

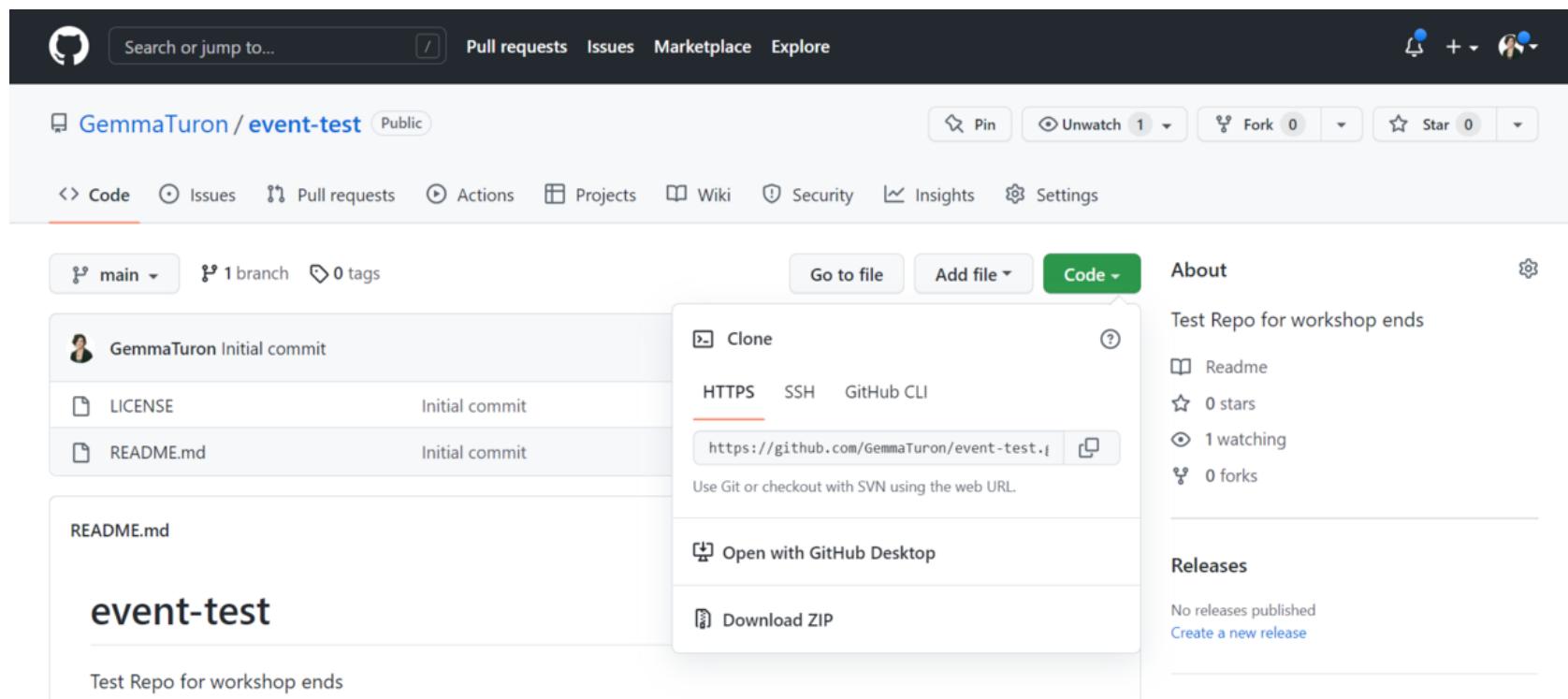
README: project description

.gitignore: depends on the end use of the repository

License: Open Source License



Clone the repository in your local computer



The screenshot shows a GitHub repository page for "GemmaTuron / event-test". The "Code" tab is selected. A "Clone" modal is open, displaying three cloning options: HTTPS, SSH, and GitHub CLI. The HTTPS URL is shown as <https://github.com/GemmaTuron/event-test.git>. Below the modal, the repository's contents are listed, including files like LICENSE and README.md.

GemmaTuron / event-test Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

Clone

HTTPS SSH GitHub CLI

<https://github.com/GemmaTuron/event-test.git>

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

Test Repo for workshop ends

Readme 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

GemmaTuron Initial commit

LICENSE Initial commit

README.md Initial commit

README.md

event-test

Test Repo for workshop ends



Clone a GitHub Repo



```
#Clone the repository in your Desktop
(base) PS C:\Users\Gemma> cd Desktop
(base) PS C:\Users\Gemma\Desktop> git clone https://github.com/GemmaTuron/event-test.git
Cloning into 'event-test'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
(base) PS C:\Users\Gemma\Desktop>

#Let's check what happened
(base) PS C:\Users\Gemma\Desktop> cd event-test
(base) PS C:\Users\Gemma\Desktop\event-test> ls

Directory: C:\Users\Gemma\Desktop\event-test

Mode                LastWriteTime         Length Name
----                -----          -----  --
-a---        21/09/2022     16:37           1088 LICENSE
-a---        21/09/2022     16:37            43 README.md
```



Make changes to local repo



```
#We first add a file to the repository
(base) PS C:\Users\Gemma\Desktop\event-test> notepad file1.txt
(base) PS C:\Users\Gemma\Desktop\event-test> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file1.txt

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

(base) PS C:\Users\Gemma\Desktop\event-test> git add file1.txt
(base) PS C:\Users\Gemma\Desktop\event-test> git commit -m "adding file1"
[main b333f13] adding file1
 1 file changed, 1 insertion(+)
  create mode 100644 file1.txt
(base) PS C:\Users\Gemma\Desktop\event-test> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```



Push changes to remote



```
(base) PS C:\Users\Gemma\Desktop\event-test> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 157.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/GemmaTuron/event-test.git
  bcc9514..b333f13 main -> main
(base) PS C:\Users\Gemma\Desktop\event-test>
```

When working with a remote Git folder, in GitHub or another online server, we need to push the changes using the `git push`

Go online and check what happened!



A small exercise:

Go to your online repository, and add a file directly using the Add File button.

Explore the `git pull` command to get a copy of this file on your local repository as well!