Git version control

Git is a tool to manage sourcecode

Never lose your coding progress again

Enable native scrolling

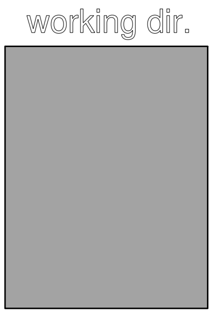
An empty folder

Windows

Go to your programs overview and start Git Bash

Everywhere else

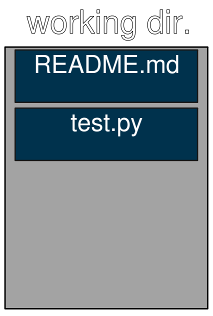
Open a terminal window



`cd` into the directory where you save your python projects

Type `mkdir test` to create a new empty folder called `test`

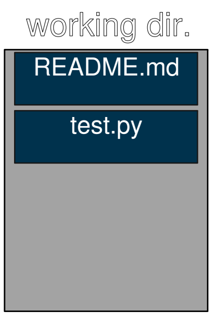
Type `cd test` to go to the new folder



Open your text editor and create the files `README.md` and `test.py` with some dummy content like `hello world` or `test`

Save the files to the folder you created on the previous slide

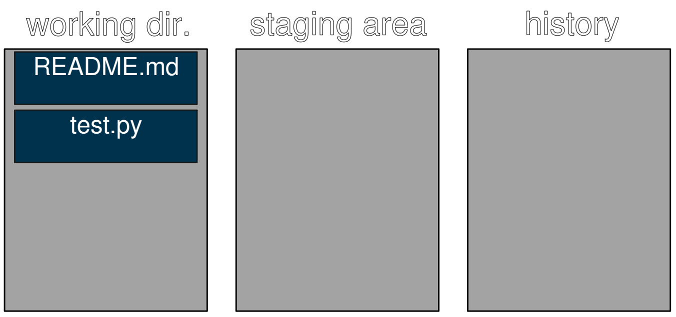
Type `ls` in the terminal to verify that you saved the files correctly



The folder in which you saved the files is called the

working directory

we now want to manage this directory using git

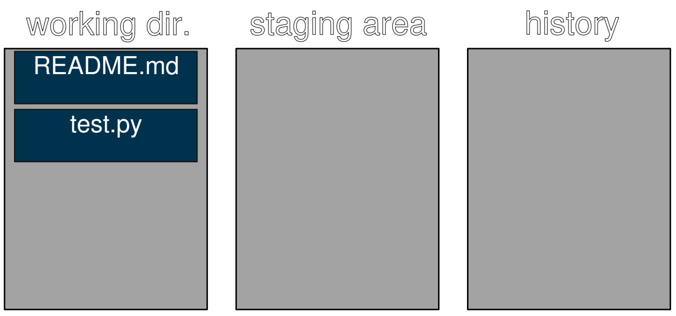


To make the current directory a git repository type `git init` on the terminal

`git init` creates a hidden `.git` folder in the working directory to store all its data

Type `ls -a` to see the hidden `.git` directory

Type `ls .git` to see the content of the `.git` directory



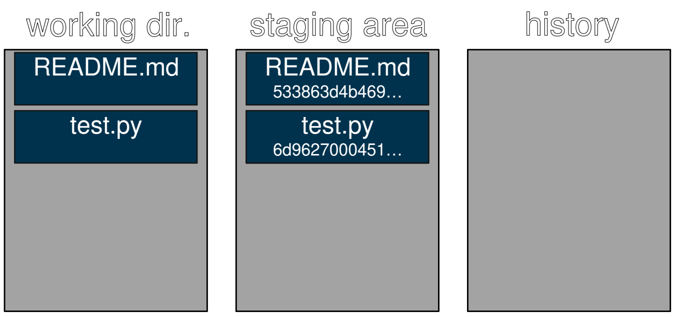
We want git to keep track of our `README.md` and `test.py` files

Git does not perform any automatic synchronizations in the background like e.g. Dropbox does

Instead we have to manually inform git about changes we made to the files in the working directory

```
$ git status
On branch master

Initial commit

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

  README.md
  test.py

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

Use the `git status` command to find the files in the working directory that git does not yet track



On the previous slide git told us to use `git add` to start tracking `README.md` and `test.py`

Type `git add README.md test.py` on the terminal

`git add` creates copies of the files and saves them under the `.git` directory

It also calculates hash values that uniquely identify the contents of the files

Commits

```
$ git status
On branch master

Initial commit

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

	new file:   README.md
	new file:   test.py
```

If we now rerun `git status` it will tell us, that `README.md` and `test.py` are "to be committed"

Commits

A commit in git is a snapshot of the whole project at a given time

As long as you do not actively break your git repository you can always go back to *every previous commit* in your git history

E.g. when you accidentially delete one of your source-files you will only lose the work you did since the last commit

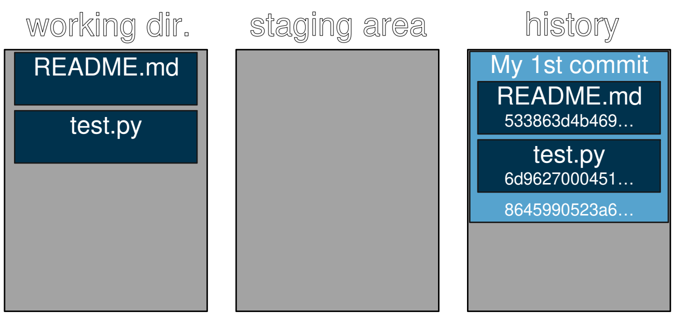
Or when your project stops working you can always go back to a previous version that is known to work and find the breaking change

Commits

In order to create a new commit you:

- `git add` every file that you want to be part of the commit
- use `git status` to check if you included all your changes
- Type `git commit -m "Your description here"`

You should replace "Your description here" by a description of the changes you did since the last commit



In the example we are now about to commit the changes to `README.md` and `test.py`

Type `git commit -m "My 1st commit"` as this is our first commit in this repository

You can now use `git log` to see the history of this repository

Commit #2

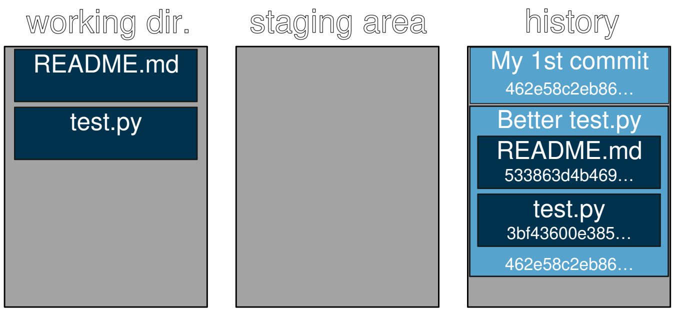
For the next commit we want `test.py` to output `"Hello World!"` when it is executed

Replace the text you put into `text.py` with the following program:

```
print('Hello World!')
```

Create a commit that includes the new `test.py`

Make sure to give it a good description of the changes



The second commit stores:

- A reference to the previous commit
- A description "Better test.py"
- References to the old `README.md` and the new `test.py`

Use `git log` to look at the history of the project

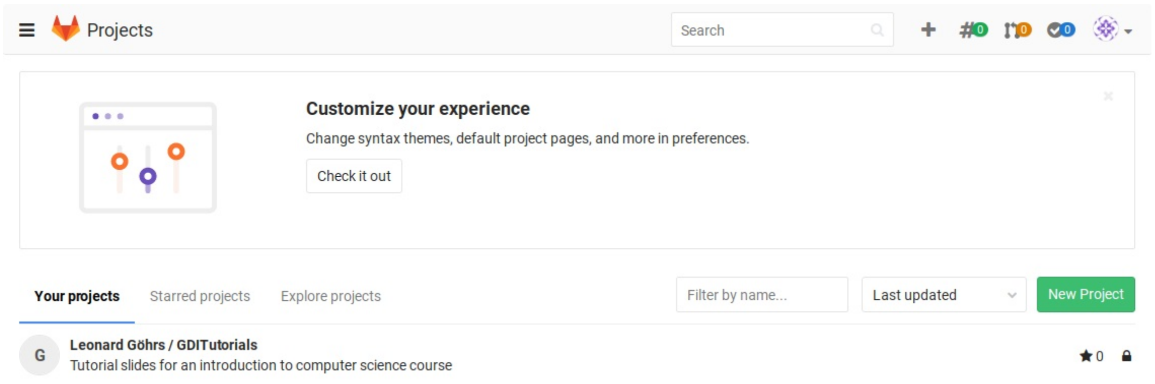
Syncing

Although not losing any local work is already a great feature git can do even more

One of the other mayor features is being able to synchronize repositories with a server

This allows you to simultaneously collaborate on software over the internet

Syncing



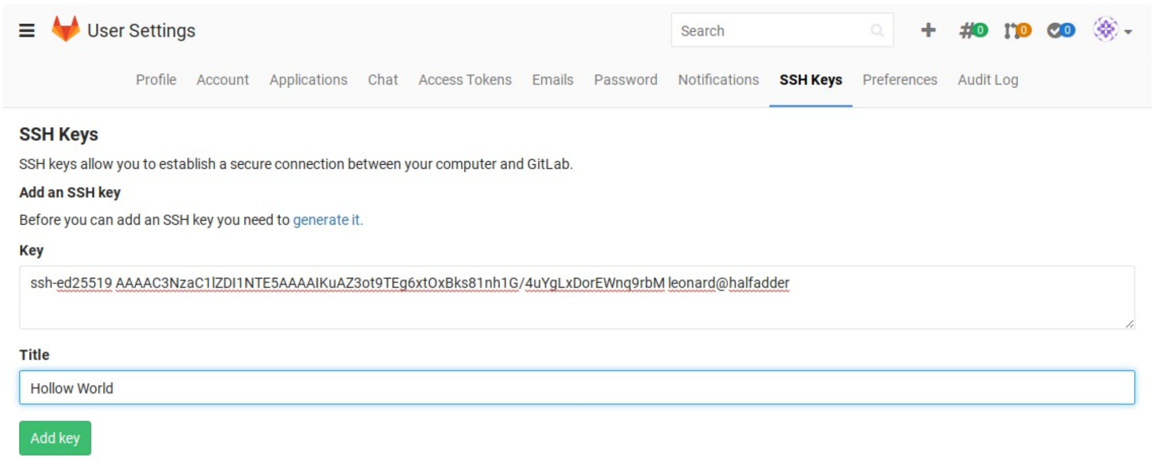
Customize your experience

Change syntax themes, default project pages, and more in preferences.

Leonard Göhrs / GDITutorials
Tutorial slides for an introduction to computer science course

Sign into/create an account on a codesharing site that supports git like Gitlab, Bitbucket or Github

Syncing



SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

Before you can add an SSH key you need to generate it.

Key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKuAZ3ot9TEg6xtOxBks81nh1G/4uYgLxDorEWnq9rbM leonard@halfadder
```

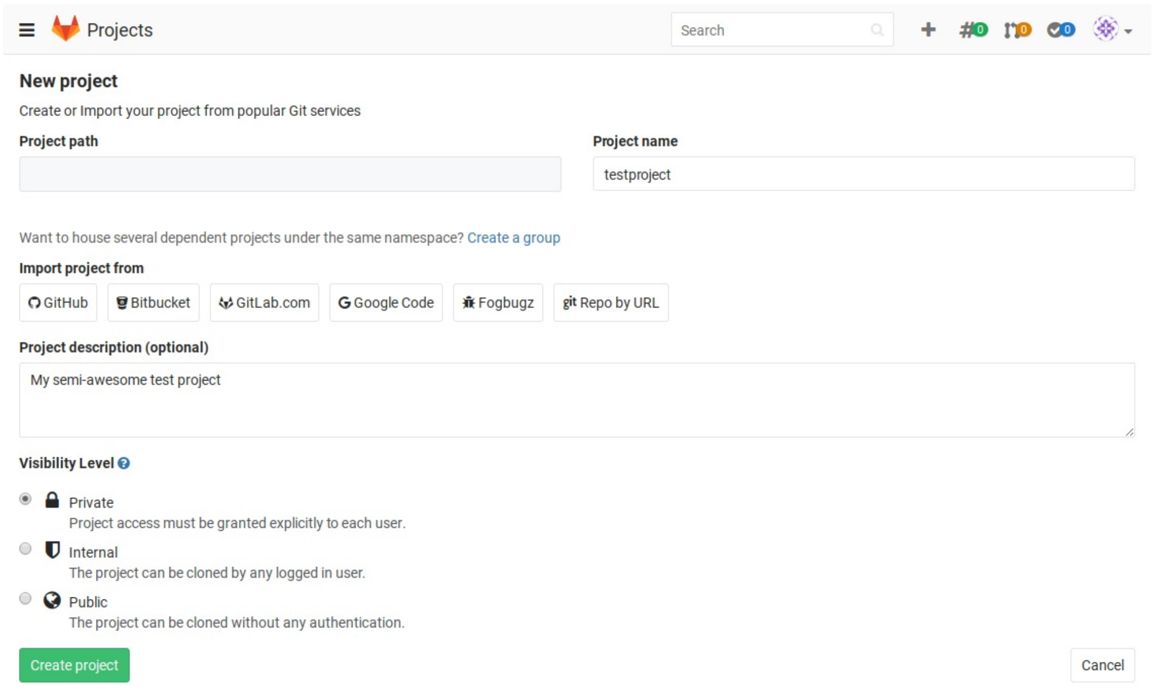
Title

Hollow World

In order to use a git-based codesharing site you have to generate a ssh public key once and configure the site to associate it with your account

Click here for a tutorial on how to generate a public key

Syncing


Projects

New project

Create or Import your project from popular Git services

Project path:

Project name: testproject

Want to house several dependent projects under the same namespace? Create a group

Import project from

Project description (optional): My semi-awesome test project

Visibility Level

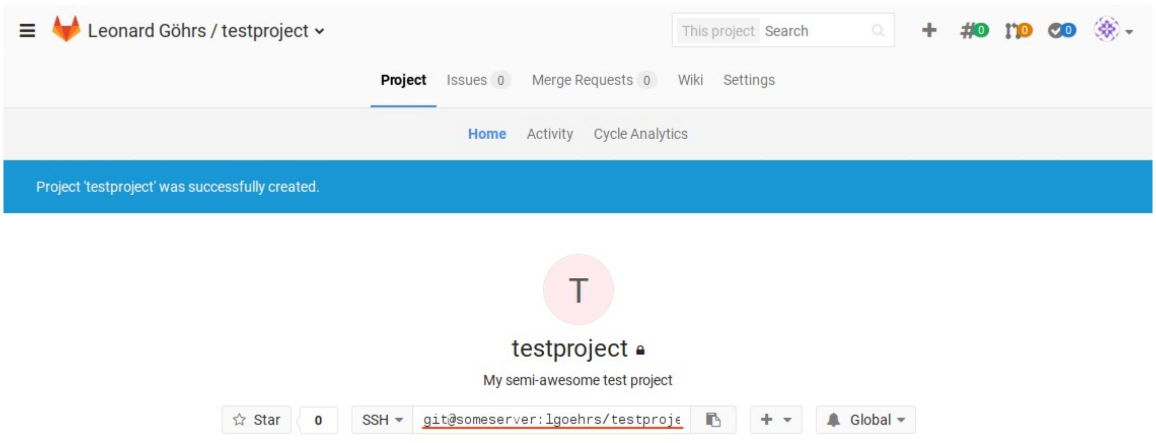
(●) Private
Project access must be granted explicitly to each user.

( ) Internal
The project can be cloned by any logged in user.

( ) Public
The project can be cloned without any authentication.

After configuring your public key you can create a new project …

Syncing


Leonard Göhrs / testproject

Project 'testproject' was successfully created.

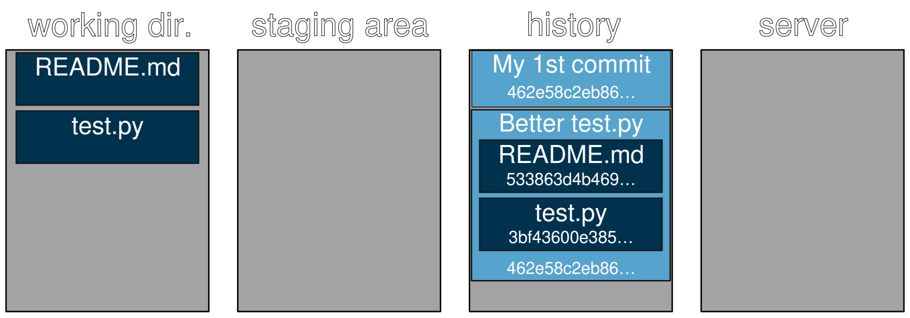
testproject

My semi-awesome test project

```
git@someserver:lgoehrs/testproje
```

… copy the address of the new project …

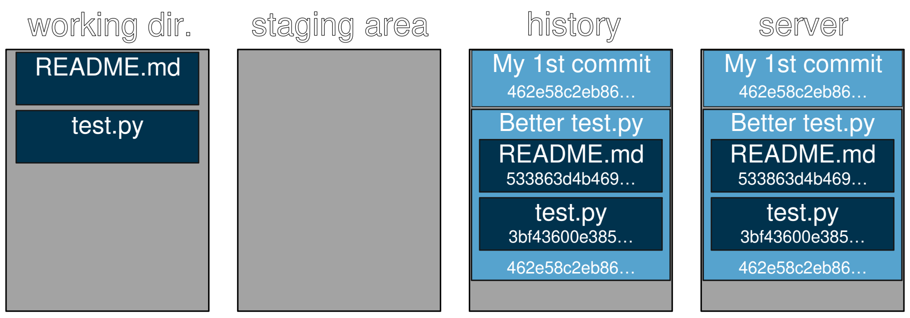
Syncing



... and add it to your git repository as a remote using the command:

```
git remote add origin [address of the project]
```

Syncing

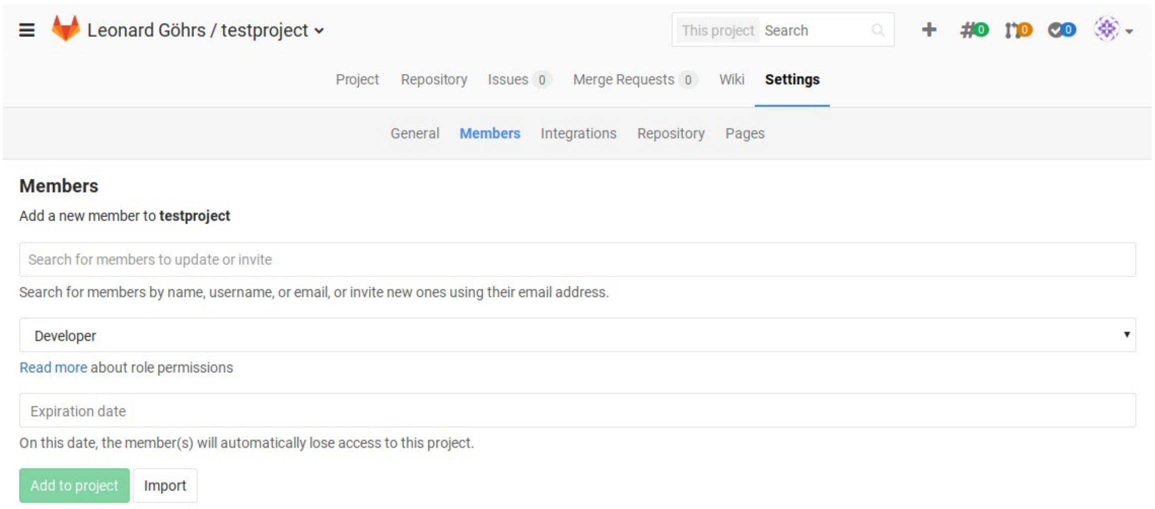


Once again git does not perform any automatic synchronizations.

In order to publish your changes to the server you have to use the `git push` command

To setup this remote as a default type `git push --set-upstream origin master` to push your changes to the server

Collaborating



Members

Add a new member to **testproject**

Search for members to update or invite

Search for members by name, username, or email, or invite new ones using their email address.

Developer

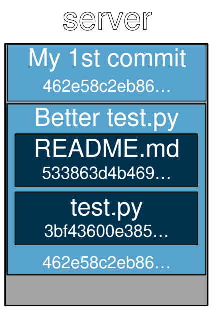
Read more about role permissions

Expiration date

On this date, the member(s) will automatically lose access to this project.

In order to collaborate on a project you have to add another user as a developer to the project

Add a partner to your project



To get a copy of an already existing repository you can use the `git clone` command

Ask your partner for the address she/he used in the `git remote add [address]` command

Type `cd ..` to go one directory up

Type `git clone [address] partner` to get a copy of your partners code into a new `partner` folder

Outlook

Git has a lot more feature than we covered here

We have not covered:

- What to do if you and your partner made changes to the code at the same time
- How to retrieve old commits
- What branches are
- Merging, rebasing, stashing

But as long as you stick to the commands that were demonstrated in this turorial you will not lose progress because of accidentially overwritten/deleted files

CheatSheet - Commands

- `git init` - Initialize a repository in the current directory
- `git remote add` - Tell git that you want to use a server
- `git clone` - Clone a remote repository
- `git status` - Run this command whenever you need advise
- `git add` - Tell git about the newest version of a file
- `git commit` - Create a snapshot of the current state and give it a name
- `git pull` - Get the latest commits from a server
- `git push` - Send the latest commits to a server

Git kaputt

```
$ git push
To remote
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'remote'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Problem: `git push` gets rejected when there are changes on the server you do not have locally

Workaround: `git pull` the changes first

Git kaputt

```
$ git pull
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From remote
   bcdbca8..0a78d9b  master     -> origin/master
Updating bcdbca8..0a78d9b
error: Your local changes to the following files would be overwritten by merge:
    README.md
Please commit your changes or stash them before you merge.
Aborting
```

Problem: `git pull` complains about uncommited files

Workaround: `git add` and `git commit` your local changes and `git pull` again

Git kaputt

```
$ git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Problem: `git pull` complains because you and someone else edited the same file

Workaround: Go through every file git complains about, search for any occurence of `<<<<<<<` and decide which version you want to keep. `git add` and `git commit` the new version.

Git kaputt

Problem: Everything is broken, nothing works, my computer is on fire

Workaround: Clone the repository from the server, forget about your old repository and get a fire extinguisher