

Git en version control

not only for software engineers

Dawid Zalewski & Ronald Tangelder

September 2, 2021

Intro

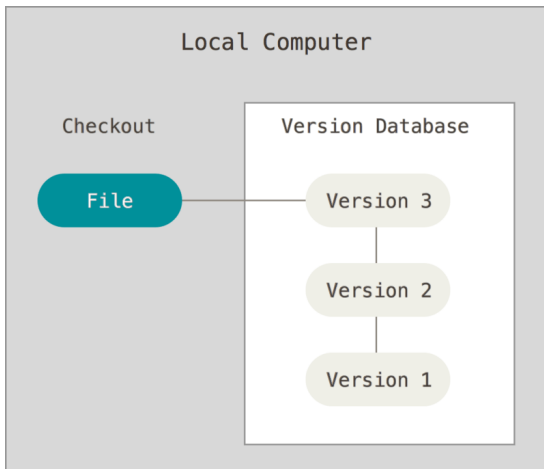
Version Control Systems

Version control is a system in which changes to a file or group of files are tracked over time so that later a specific version can be retrieved.

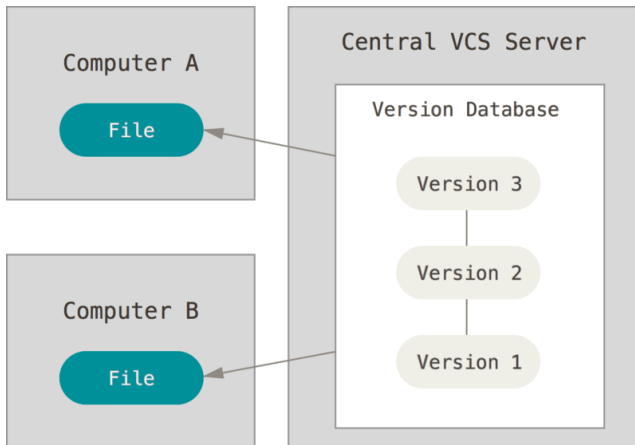
Why version control?

- Retrieve previous versions of files or the whole project
- View changes between two points in time
- Tracking who changed what

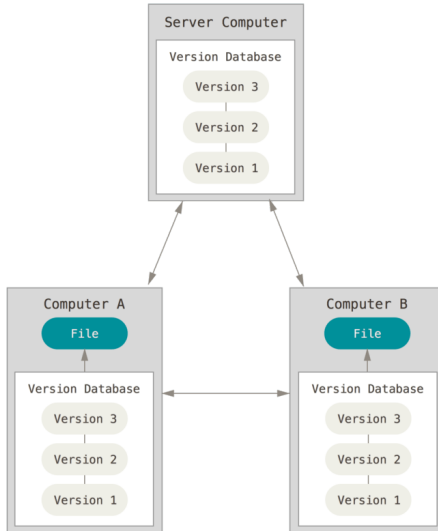
Local version control



Centralized version control



Distributed version control



What is git

Git is a *distributed* version control system

- Each team member can work independently
- Almost all operations are local
- Everyone has a local copy of the database (repository)

Before we start

- Git client: <http://git-scm.com/download/>
 - Get the right version for your system
 - Including *Git Bash* (only Windows)
- GUI tool: <http://git-scm.com/downloads/guis>
 - For example *Github Desktop*

Info

We will (try) not to use GUI tools.

Getting started - a few settings

```
$ git config --global user.name "Dawid Zalewski"
$ git config --global user.email "d.r.zalewski@saxion.nl"
```

Getting started - text editor

You can also change the default text editor:

For Notepad++:

```
$ git config --global core.editor  
"'notepad++.exe' -multiInst -nosession"
```

For Visual Studio Code:

```
$ git config --global core.editor "code --wait"
```

Getting started - check the settings

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

Local version control

Initialize a repository

```
$ mkdir my_project
```

```
$ cd my_project/
```

```
$ git init
```

```
Initialized empty Git repository in ...
```

```
$ git status
```

```
On branch master
```

```
Initial commit
```

```
nothing to commit (create/copy files  
and use "git add" to track)
```

Three states

There are three states that files can be in:

- 'modified'
- 'staged' (prepared for a commit)
- 'committed'

Three states

Modified

A file has been modified but not yet committed to the database

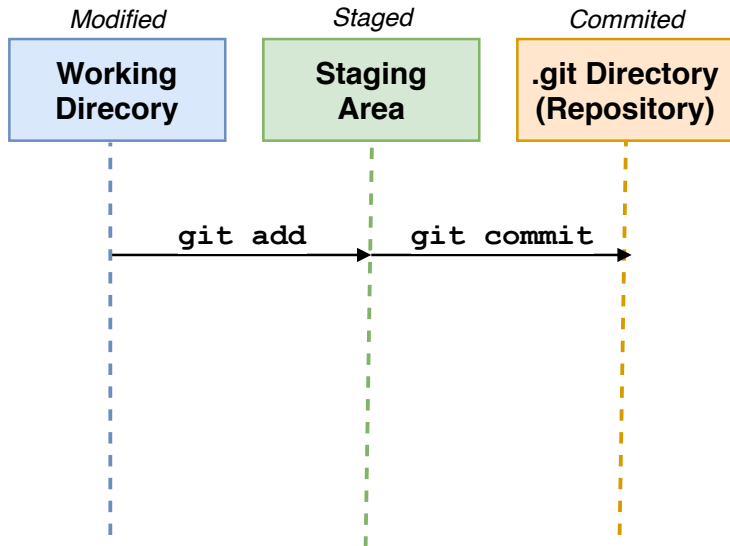
Staged

A modified file is included in the next commit

Committed

All data is safe in the local database

Git workflow & environment



Environment

Working directory

The directory structure and files in which you make changes

Staging Area

An *Intermediate station* between the **working directory** and the **repository**

Allows for selective *committing* of changes

Repository

Collection (backup database) of all commits, branches, tags, ...

First commit

Create a new file (readme.txt) in the `my_project` folder

For VS Code users:

```
$ code readme.txt
```

- Type in some text and save.

Check

```
$ ls -l
```

```
total 1
```

```
-rw-r--r-- 1 zalda 197609 7 Apr 11 12:55 readme.txt
```

Git status

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

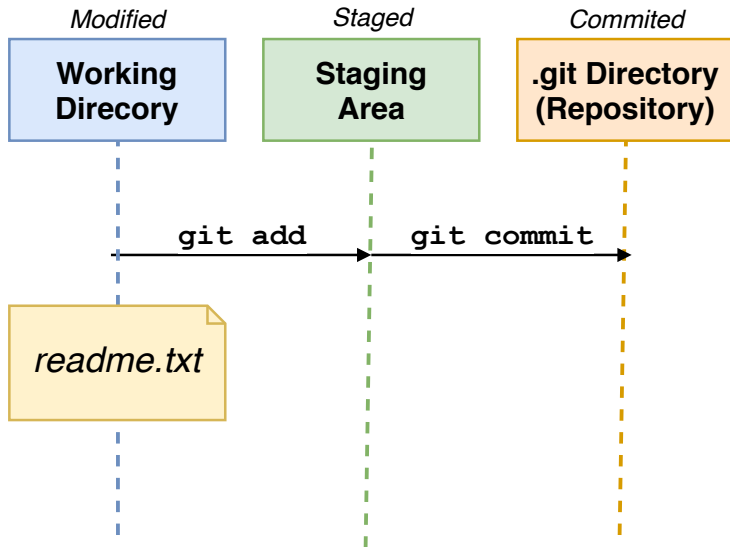
```
(use "git add <file>..." to include in what  
will be committed)
```

```
    readme.txt
```

```
nothing added to commit but untracked files present
```

```
(use "git add" to track)
```

Where is `readme.txt`



To the *Staging Area*: `git add`

```
$ git add readme.txt
```

```
$ git status
```

```
On branch master
```

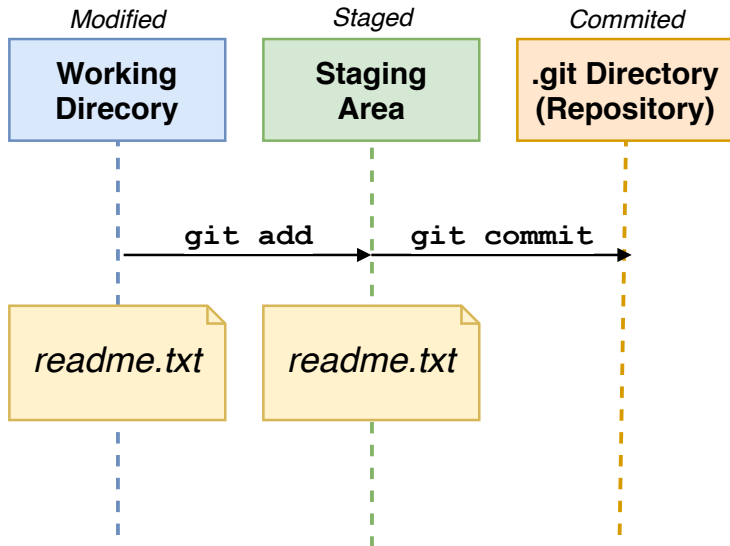
```
Initial commit
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   readme.txt
```

Where is `readme.txt`

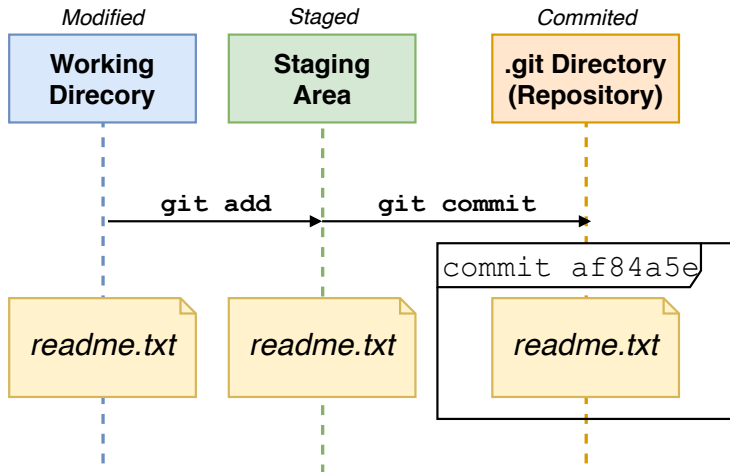


Adding a commit: `git commit`

```
$ git commit -m "readme.txt added"
[master (root-commit) af84a5e] readme.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt

$ git status
On branch master
nothing to commit, working tree clean
```

Where is `readme.txt`



One more check

```
$ git log
commit af84a5e... (HEAD -> master)
Author: Dawid Zalewski <d.r.zalewski@saxion.nl>
Date:   Wed Aug 11 14:56:28 2021 +0200
```

```
    readme.txt added
```

The original file remains in the folder.

By the way, it's in the *Staging Area* as well.

Let's modify something

Open readme.txt and add some text / modify something.

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what  
   will be committed)
```

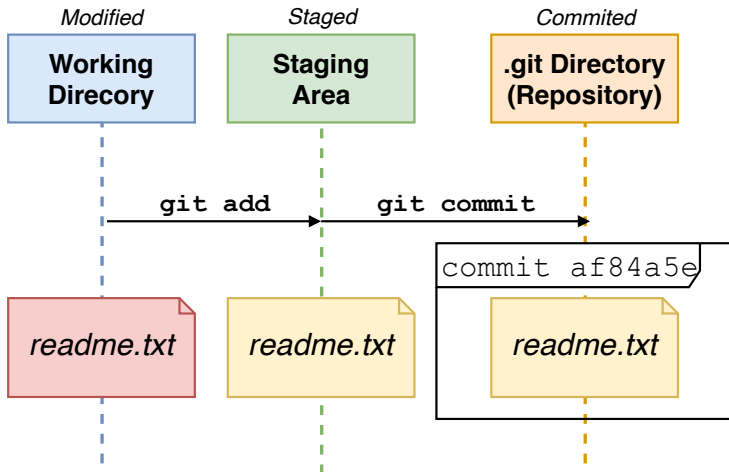
```
  (use "git checkout -- <file>..." to  
   discard changes in working directory)
```

```
        modified:   readme.txt
```

```
no changes added to commit
```

```
  (use "git add" and/or "git commit -a")
```

Git recognizes the changes



Back to the Staging Area

```
$ git add .
```

```
$ git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

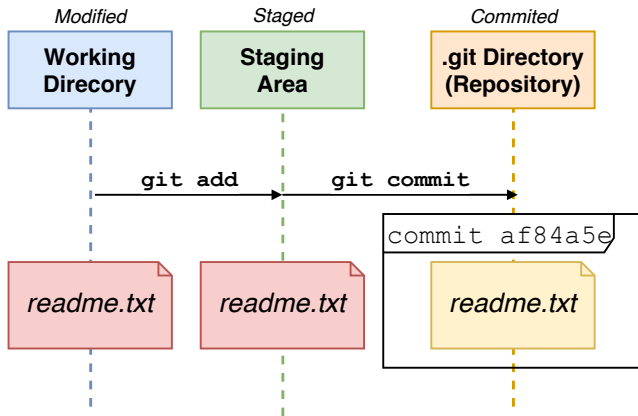
modified: readme.txt

```
git add .
```

```
$ git add .
```

- . is de current directory (incl. sub-dirs)
- You can also add files individually (by mentioning their names)
- **Please note!** Works only *within* the directory containing repository

Staging area after git add .



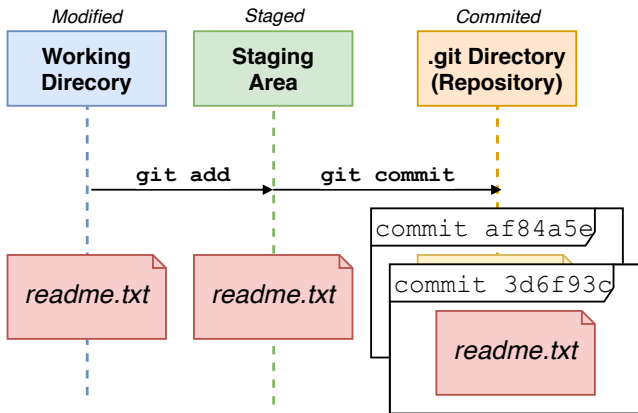
Note that `readme.txt` is still in the directory.

git only acknowledges that it is ready for committing as well.

Committing changes

```
$ git commit -m "added a line to readme.txt"
[master 3d6f93c] added a line to readme.txt
1 file changed, 1 insertions(+), 0 deletion(-)
```

The situation after the commit



We have now commits!

One more file (try it yourself)

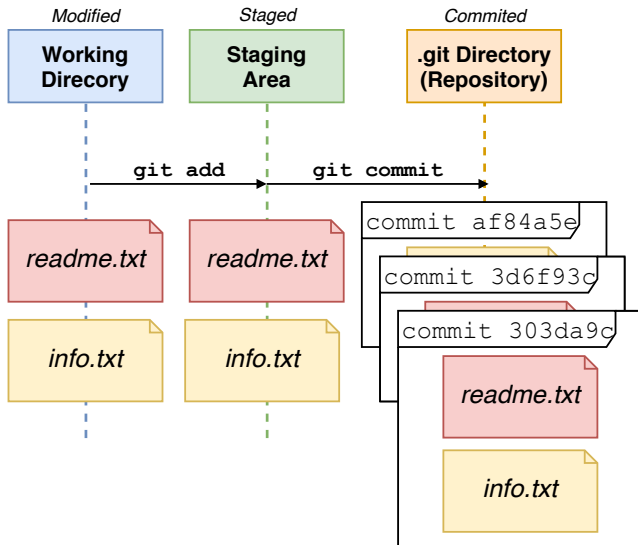
Create a new file (e.g. info.txt) and:

- Add it to the *Staging Area*.
- Commit it.

One more file (code)

```
$ touch info.txt
$ code info.txt
$ git add .
$ git commit -m "info.txt added"
$ git log --oneline
```

We now have two files



Git time machine

Undoing local changes

There are changes in a file (e.g. `info.txt`) that you want to undo.

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..."
```

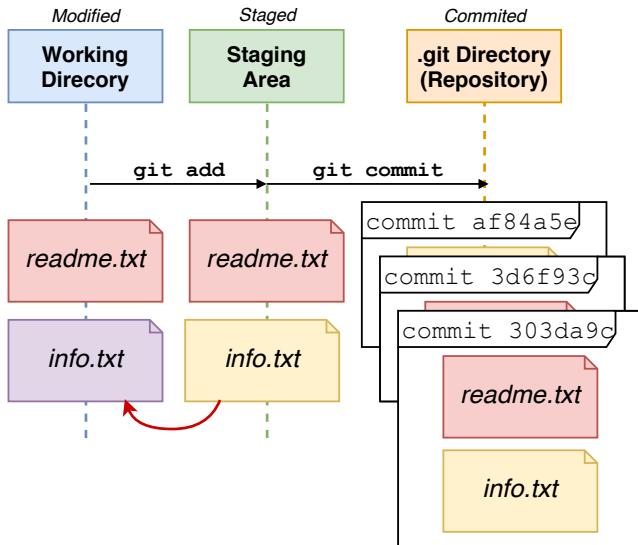
```
   to update what will be committed)
```

```
  (use "git checkout -- <file>..."
```

```
   to discard changes in working directory)
```

```
    modified:   info.txt
```

Undoing local changes



Undoing local changes

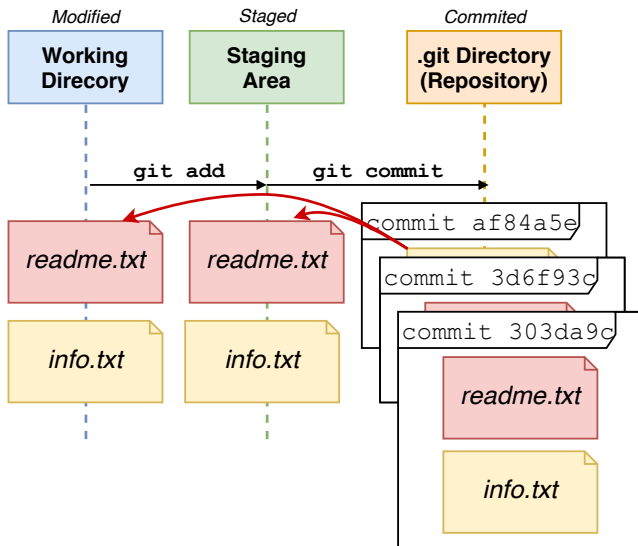
```
$ git checkout -- info.txt
```

```
$ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

Restoring a file to a previous revision



Restoring a file to a previous revision

```
$ git log --oneline
303da9c (HEAD -> master) info.txt added
3d6f93c readme updated
af84a5e readme.txt added

$ git checkout af84a5e -- readme.txt
```

The situation after checkout

```
$ git status
```

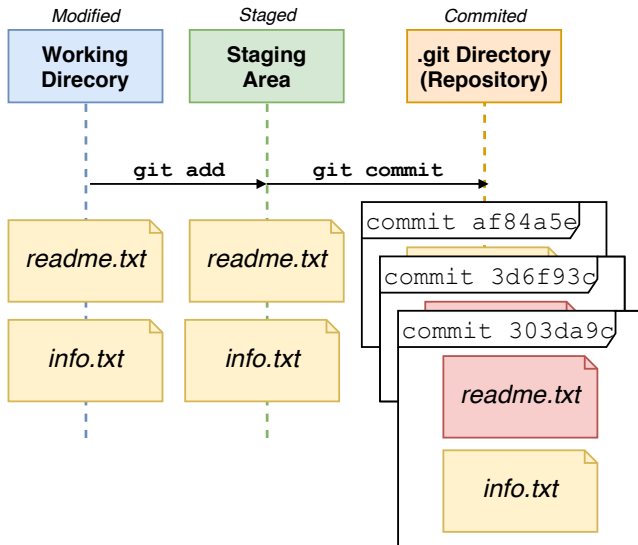
```
On branch master
```

```
Changes to be committed:
```

```
    (use "git reset HEAD <file>..." to unstage)
```

```
        modified:   readme.txt
```

The situation after checkout



The situation after checkout: what now?

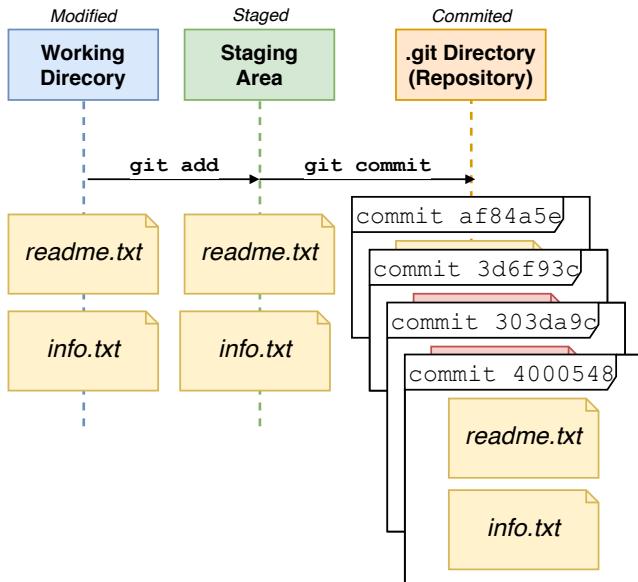
Or: make new changes,

followed by `git add .` & `git commit`

Or: commit directly

```
$ git commit -m "readme.txt back to original revision"
[master 4000548] readme.txt back to original revision
1 file changed, 1 deletion(-)
```

The situation after checkout and commit

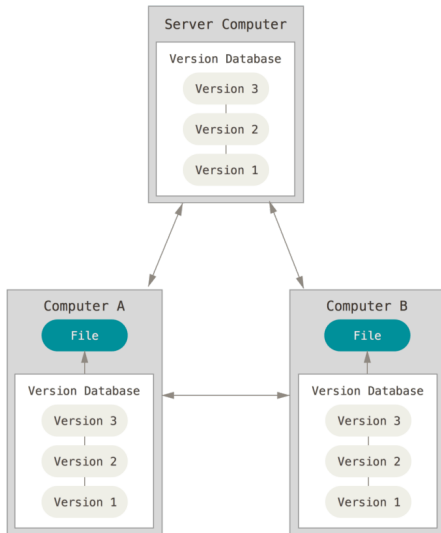


Just checking...

```
$ git log --oneline
4000548 (HEAD -> master) readme.txt back to original revision
303da9c info.txt added
3d6f93c readme updated
af84a5e readme.txt added
```


Git remote

Why remote

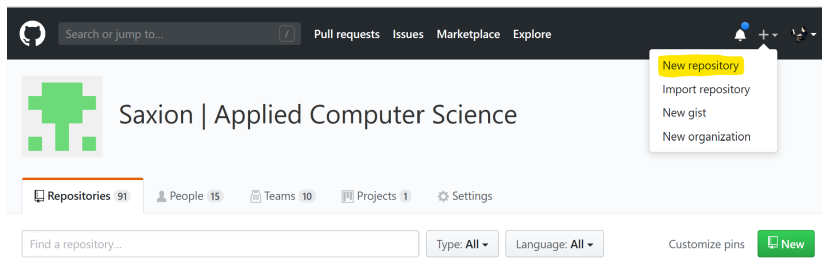


GitHub

<https://github.com/>

- (Still) most popular git-hosting service
- Free for everyone (with restrictions)
- Free for education (without restrictions)
- GitHub Classroom (continued)

Creating a new repository on GitHub



Entering details

Create a new repository

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

[Import a repository.](#)

Owner

Repository name *



SaxionACS ▾

/

git_workshop



Great repository names are short and memorable. Need inspiration? How about **potential-invention**?

Description (optional)

This is a git workshop using markdown and beamer



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Public vs. Private

Public

Anyone can see and copy (but not modify) the files.



Only the owner and team members can modify the files.

Private

Only the owner and team members can see, copy or modify the files.

Linking local with remote

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# git_workshop" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SaxionACS/git_workshop.git
git push -u origin master
```



Linking an existing (local) repo with a remote

```
$ git remote add origin  
    https://github.com/SaxionACS/git_workshop.git  
$ git push -u origin master
```

Replace the url in the command with your repo url!

Alternative: cloning the remote to a new folder

```
$ mkdir my_project
$ cd my_project
$ git clone
  https://github.com/SaxionACS/git_workshop.git
```

Has it worked?

SaxionACS / git_workshop

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

This is a git workshop using markdown and beamer [Edit](#)

[Manage topics](#)

4 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

Commit	Message	Time
zaldawid	readme.txt naar originele revisie	Latest commit 4000548 18 hours ago
info.txt	info.txt toegevoegt	19 hours ago
readme.txt	readme.txt naar originele revisie	18 hours ago

readme.txt

readme.

Has it worked?

```
$ git remote -v
```

```
origin
```

```
https://github.com/SaxionACS/git_workshop.git (fetch)
```

```
origin
```

```
https://github.com/SaxionACS/git_workshop.git (push)
```

What's next?

It depends:

- Solo user
- Team user

Remote for a solo user

```
$ git pull
[work on project]
$ git status
$ git add .
$ git commit -m "...
$ git push
```

- pull: Changes remote -> local
- push: Changes local -> remote

Remote for a team: collaborators

SaxionACS / git_workshop

Unwatch ▾

1

★ Star

0

🍴 Fork

0

<> Code

🕒 Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📊 Insights

⚙ Settings

Options

Collaborators & teams

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Alerts

Moderation

Interaction limits

Teams

+ Create new team

No teams have been given access to this repository yet. Use the form below to add a team.

Add a team ▾

Collaborators

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

Remote for a team: initialization

Everyone must point their local repo to the same remote.

One person takes care of the initialization of the remote repository.

Then:

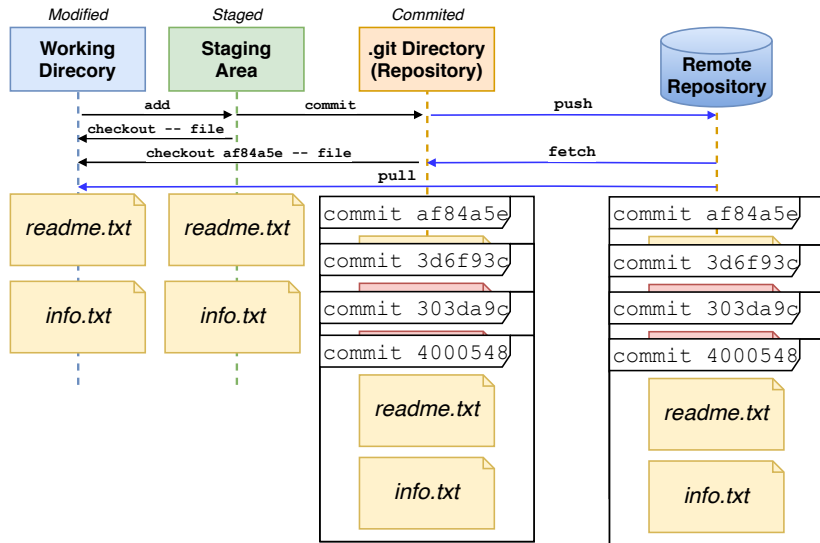
```
$ git clone  
https://github.com/SaxionACS/git_workshop.git
```

Remote for a team: workflow

```
$ git pull --rebase
[work on files]
$ git status
$ git add .
$ git commit -m "... "
$ git pull --rebase
$ git push
```

--rebase enables “simple” commit history

Working with remote



GitHub for education

GitHub for students

- Free pro accounts
- Some extra freebees
- Check <https://education.github.com/pack> for more info