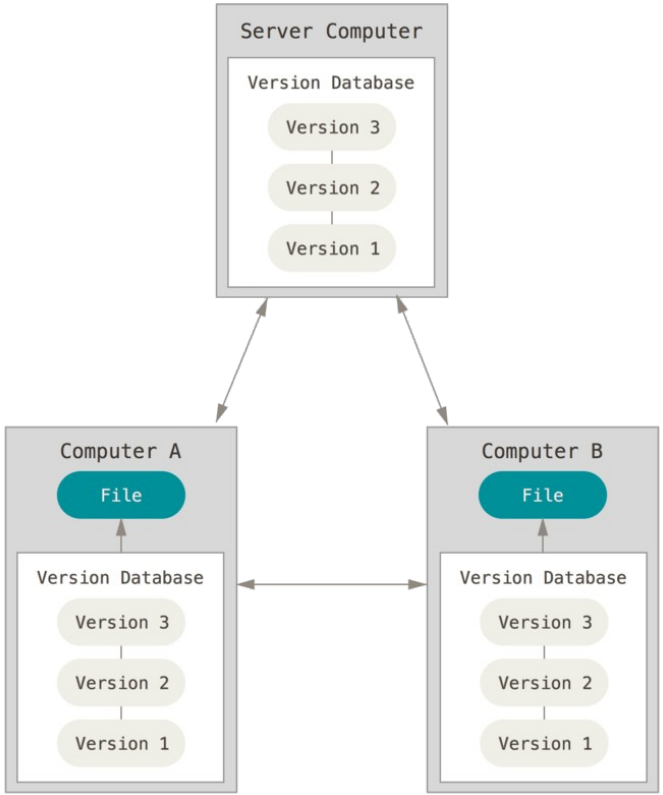
# Git introduction

**Jonas Jakobsen**
**AAU Satlab**

# What is Git - VCS?

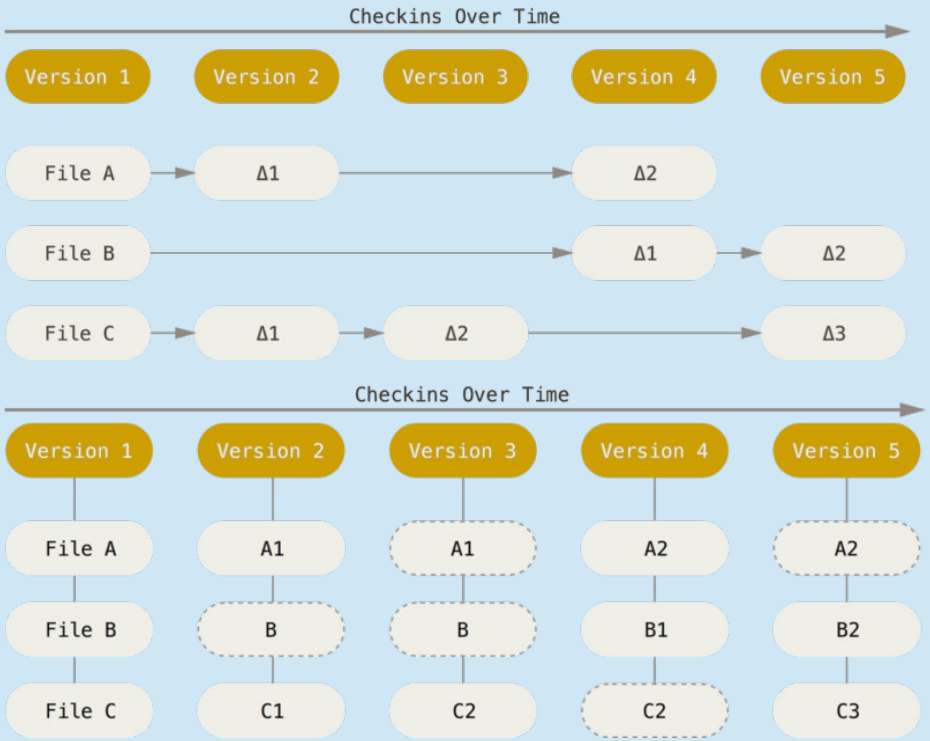
- VCS Version Control System
- (LVCS) Local VCS
- (CVCS) Centralized VCS
- (DVCS) Distributed VCS




©git-scm.com - DVCS

# What is Git?

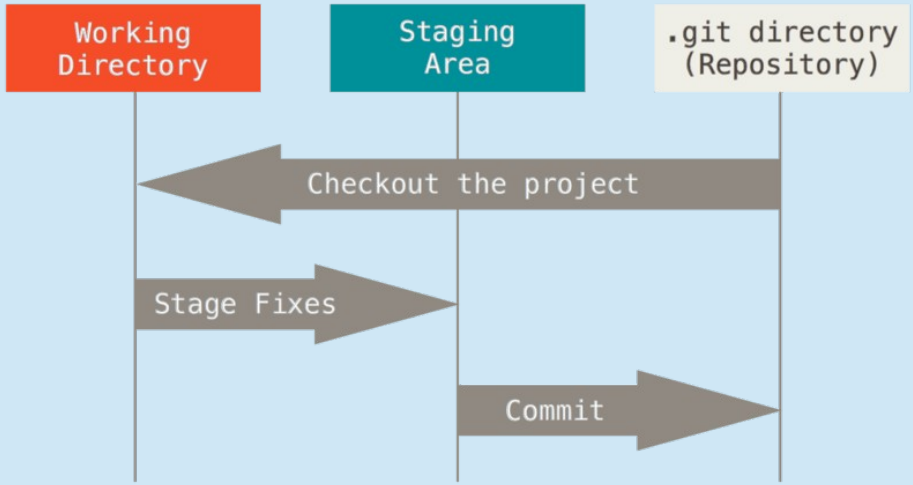
- Delta-based – Track file changes over time.
- Snapshot-based
  - Like taking a picture of the current files.
  - Git is snapshot based.




©git-scm.com – Delta based (Top) and Snapshot based (bottom)

# What is Git?

- Three states
  - Modified
  - Staged
  - Comitted



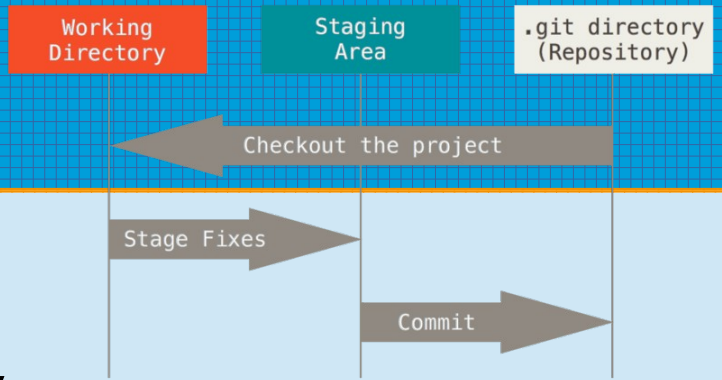

©git-scm.com – Working Tree, staging area and Git directory

# Gits history

- Created for the Linux Kernel
- Goals for Git:
  - Speed
  - Simple Design
  - Good for non-linear development
  - Handle large projects

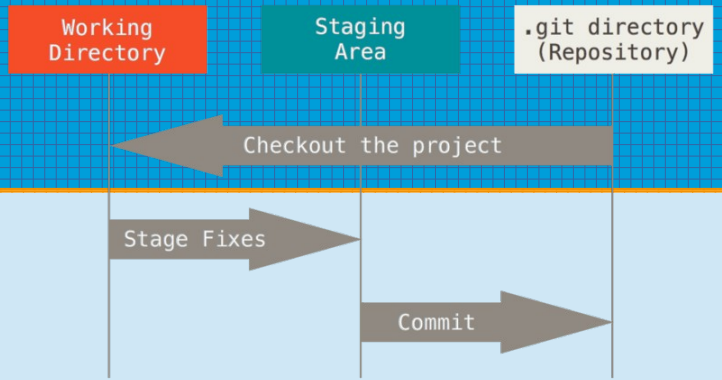
©Britannica – Linus Torvalds

# Common commands




- git init - Initialize
- git clone - Clone existing repository
- git add - Add files to staging area
- git commit - Commit to repositoty
- git push - Upload to remote
- git pull - Pull from remote
- git checkout - Switch between branches
- git branch - List branches
- git status - Show status og staging area

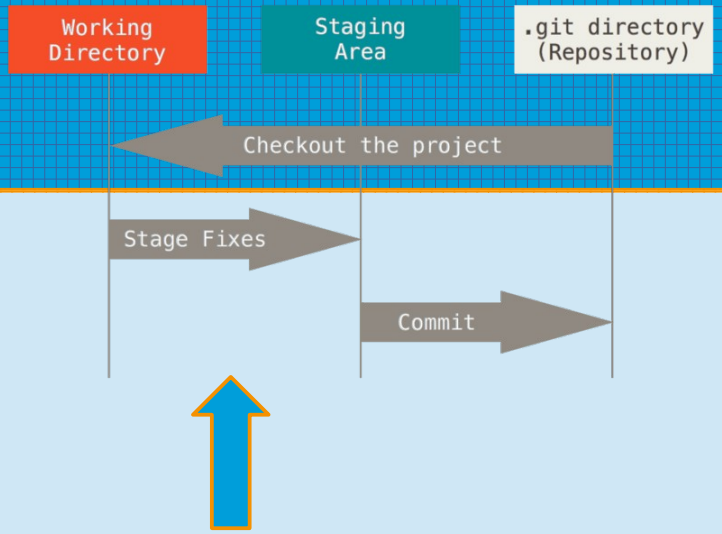
# Getting started




- Create a repository
- Local: `git init {path}`
- Remote: `git clone {source} [path]`

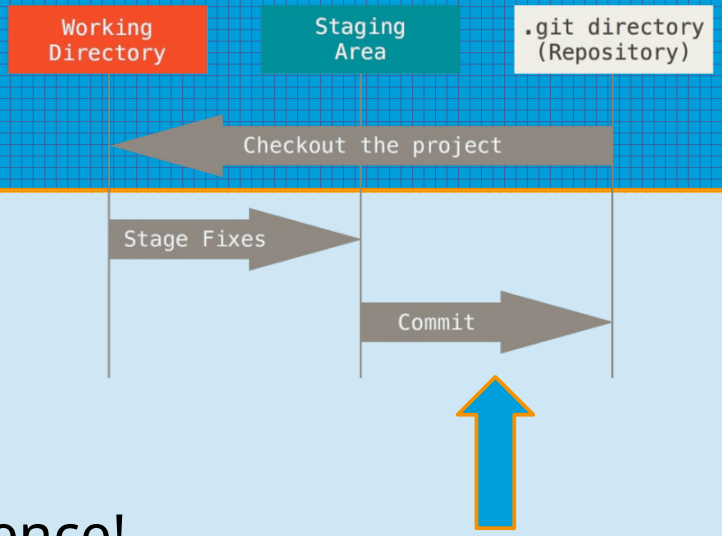
Notation: [optional] {required}

# Getting started




- Add files:
- git add {path}
- Don’t add everything!!!!
  - eg. `git add * / .`

# Getting started




- Save staged files
- git commit [-m „Commit message“]
  - Be descriptive, this is your primary reference!

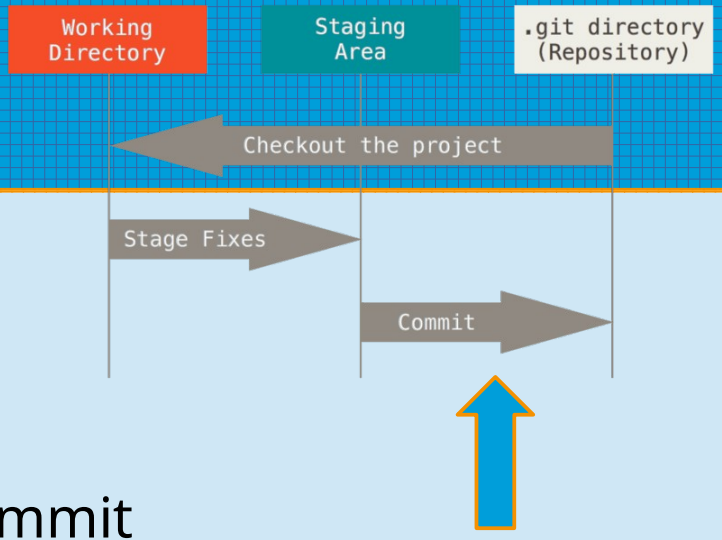
```
Author: Jonas Jakobsen
Date:   Sun Jan 28 20:45:48 2024 +0100

    A descriptive commit message
```

Jonasj2001 A descriptive commit message 9dab8eb · 3 minutes ago 1 Commits

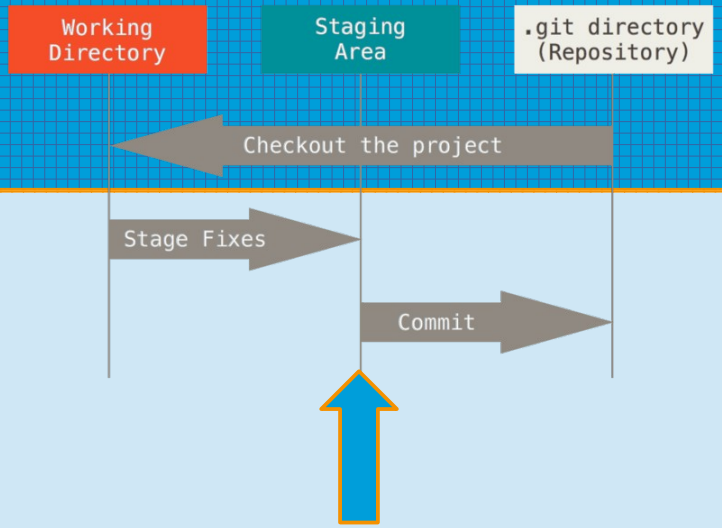
pres.odp A descriptive commit message 3 minutes ago

# Getting started




- Committed to early?
  - Do the forgotten operations
  - git commit --amend → Overwrites old commit

# Getting started




- Keep track of files
- git status [path]

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   pres.odp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .~lock.pres.odp#
        newfile

no changes added to commit (use "git add" and/or "git commit -a")
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   newfile
        modified:   pres.odp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .~lock.pres.odp#
```

# Exercise 1:

- Install Git (git-scm.com/downloads)
- Generate and add a SSH key to your Github
- Create a new repository
- Clone the repository and add a file.
- Commit and push to Github.

Generate SSH-key:

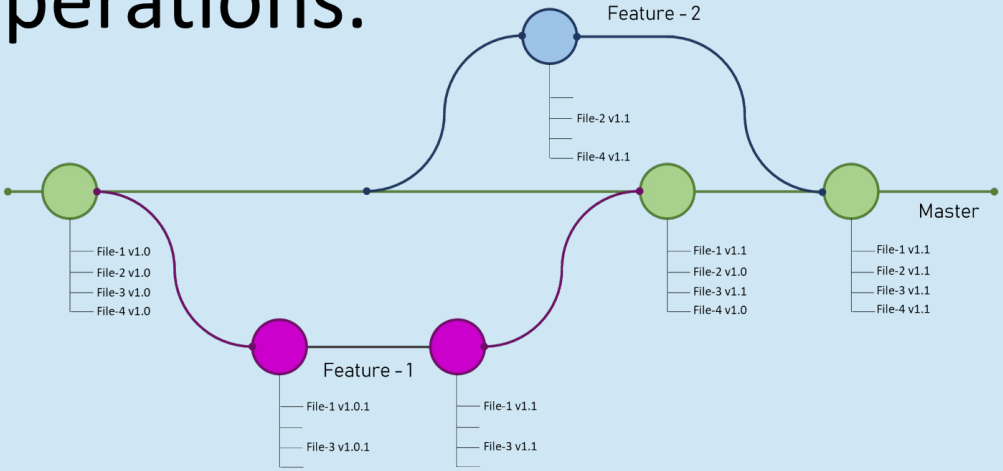
```
ssh-keygen -t ed25519 -C „your_email@example.com“
```

Add to Github:

https://github.com/settings/keys

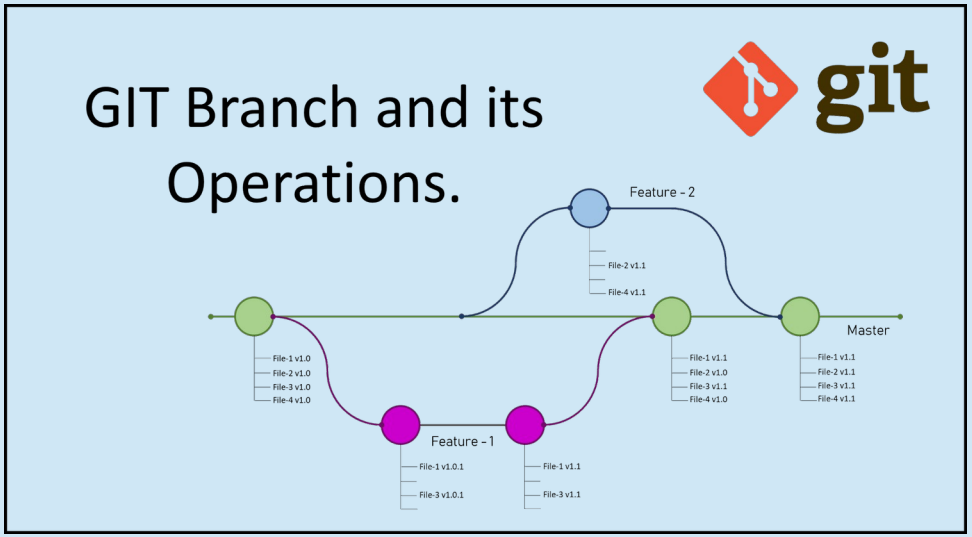
# Branching

GIT Branch and its Operations.

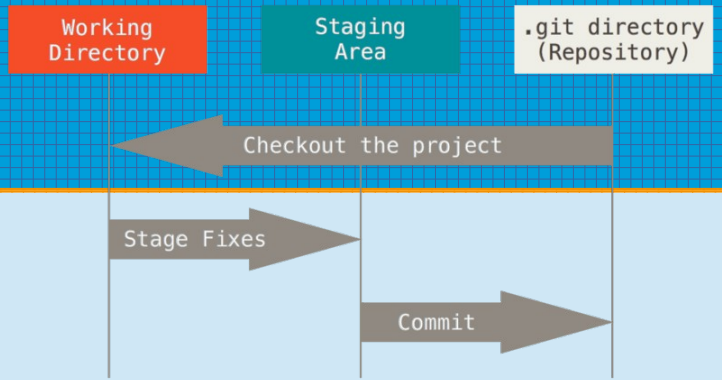


# Branching

- Create new branch:
  - git checkout -b branchname
- Switch branch
  - git checkout {branchname}



# Branching




- Error when pushing!
  - Remember Distributed VCS
- git push -u origin HEAD
  - HEAD (Reference to last commit)

```
fatal: The current branch newfeature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin newfeature

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

# Merging

- Merge to current branch
  - git merge {branch} [-m message]
- Remove branch:
  - Local: git branch -d {local_branch}
  - Remote: git push origin -d {remote_branch}

# Pull-Request

- Demo

# Exercise 2:

- Create a new branch
- Check out your new branch
- Add a new file
- Commit and Push to origin
  - See your new branch online.
- Merge your branch to main, or create and close a PR
- Delete branch(-es)

# Merge Conflicts

Here are lines that are either unchanged from the common
ancestor, or cleanly resolved because only one side changed,
or cleanly resolved because both sides changed the same way.
<<<<<<< yours:sample.txt
Conflict resolution is hard;
let's go shopping.
=======
Git makes conflict resolution easy.
>>>>>>> theirs:sample.txt
And here is another line that is cleanly resolved or unmodified.

| | |
|---|---|
| <<<<<<< | Your changes located below. |
| ======= | Marks conflict. |
| >>>>>>> | Remote changes located above. |

# Merge Conflicts

- Git is good at merging files!
- Conflicts happens if the same line is edited.

- Tools to help with source control:
  - VS code
  - Git kraken

# Exercise 3:

- Split into groups of two.
- Edit the same file in two different places.
  - What happens when you push and pull?
- Repeat by editing the file in the same place.
  - What happens?
- Resolve any potential merge request.

# How to avoid messages like this?

- We use `.gitignore` files
- Why?
  - Tells git not to track specific files
  - Eg. `Target` folder in Rust projects.

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        1 Sun 28 Jan 20:29:37 CET 2024
        10 Sun 28 Jan 20:29:46 CET 2024
        11 Sun 28 Jan 20:29:47 CET 2024
        12 Sun 28 Jan 20:29:48 CET 2024
        13 Sun 28 Jan 20:29:49 CET 2024
        14 Sun 28 Jan 20:29:50 CET 2024
        15 Sun 28 Jan 20:29:51 CET 2024
        16 Sun 28 Jan 20:29:52 CET 2024
        17 Sun 28 Jan 20:29:53 CET 2024
        18 Sun 28 Jan 20:29:54 CET 2024
        19 Sun 28 Jan 20:29:55 CET 2024
        2 Sun 28 Jan 20:29:38 CET 2024
        20 Sun 28 Jan 20:29:56 CET 2024
        21 Sun 28 Jan 20:29:57 CET 2024
        22 Sun 28 Jan 20:29:58 CET 2024
        23 Sun 28 Jan 20:29:59 CET 2024
        24 Sun 28 Jan 20:30:00 CET 2024
        25 Sun 28 Jan 20:30:01 CET 2024
        26 Sun 28 Jan 20:30:02 CET 2024
        27 Sun 28 Jan 20:30:03 CET 2024
        28 Sun 28 Jan 20:30:04 CET 2024
        29 Sun 28 Jan 20:30:05 CET 2024
        3 Sun 28 Jan 20:29:39 CET 2024
        30 Sun 28 Jan 20:30:06 CET 2024
        4 Sun 28 Jan 20:29:40 CET 2024
        5 Sun 28 Jan 20:29:41 CET 2024
        6 Sun 28 Jan 20:29:42 CET 2024
        7 Sun 28 Jan 20:29:43 CET 2024
        8 Sun 28 Jan 20:29:44 CET 2024
        9 Sun 28 Jan 20:29:45 CET 2024
        generator.sh

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

# .gitignore

- Adding *2024 to .gitignore
  Transforms the output to:

```
On branch main

No commits yet

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

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

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        1 Sun 28 Jan 20:29:37 CET 2024
        10 Sun 28 Jan 20:29:46 CET 2024
        11 Sun 28 Jan 20:29:47 CET 2024
        12 Sun 28 Jan 20:29:48 CET 2024
        13 Sun 28 Jan 20:29:49 CET 2024
        14 Sun 28 Jan 20:29:50 CET 2024
        15 Sun 28 Jan 20:29:51 CET 2024
        16 Sun 28 Jan 20:29:52 CET 2024
        17 Sun 28 Jan 20:29:53 CET 2024
        18 Sun 28 Jan 20:29:54 CET 2024
        19 Sun 28 Jan 20:29:55 CET 2024
        2 Sun 28 Jan 20:29:38 CET 2024
        20 Sun 28 Jan 20:29:56 CET 2024
        21 Sun 28 Jan 20:29:57 CET 2024
        22 Sun 28 Jan 20:29:58 CET 2024
        23 Sun 28 Jan 20:29:59 CET 2024
        24 Sun 28 Jan 20:30:00 CET 2024
        25 Sun 28 Jan 20:30:01 CET 2024
        26 Sun 28 Jan 20:30:02 CET 2024
        27 Sun 28 Jan 20:30:03 CET 2024
        28 Sun 28 Jan 20:30:04 CET 2024
        29 Sun 28 Jan 20:30:05 CET 2024
        3 Sun 28 Jan 20:29:39 CET 2024
        30 Sun 28 Jan 20:30:06 CET 2024
        4 Sun 28 Jan 20:29:40 CET 2024
        5 Sun 28 Jan 20:29:41 CET 2024
        6 Sun 28 Jan 20:29:42 CET 2024
        7 Sun 28 Jan 20:29:43 CET 2024
        8 Sun 28 Jan 20:29:44 CET 2024
        9 Sun 28 Jan 20:29:45 CET 2024
        generator.sh

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

# Other useful commands:

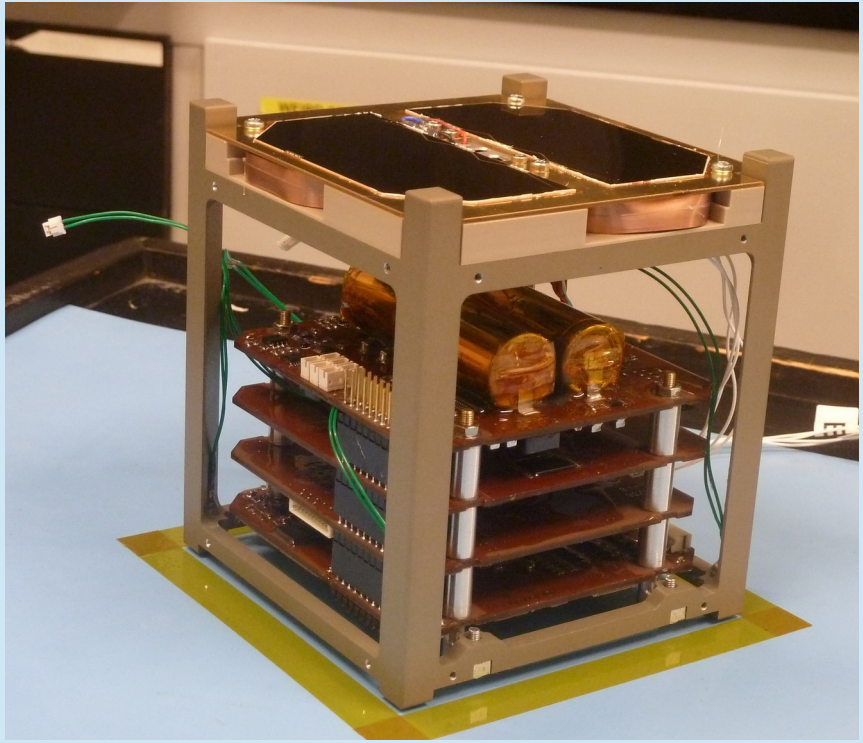
- git reset HEAD → Unstage staged files to HEAD.
- git restore {file} → Remove modifications from file.
  - Can also be used to unstage files:

    ```
    git restore --staged {file}
    ```
- git stash → Temporarily save current progress.

# Who/What - Are AAU Satlab?

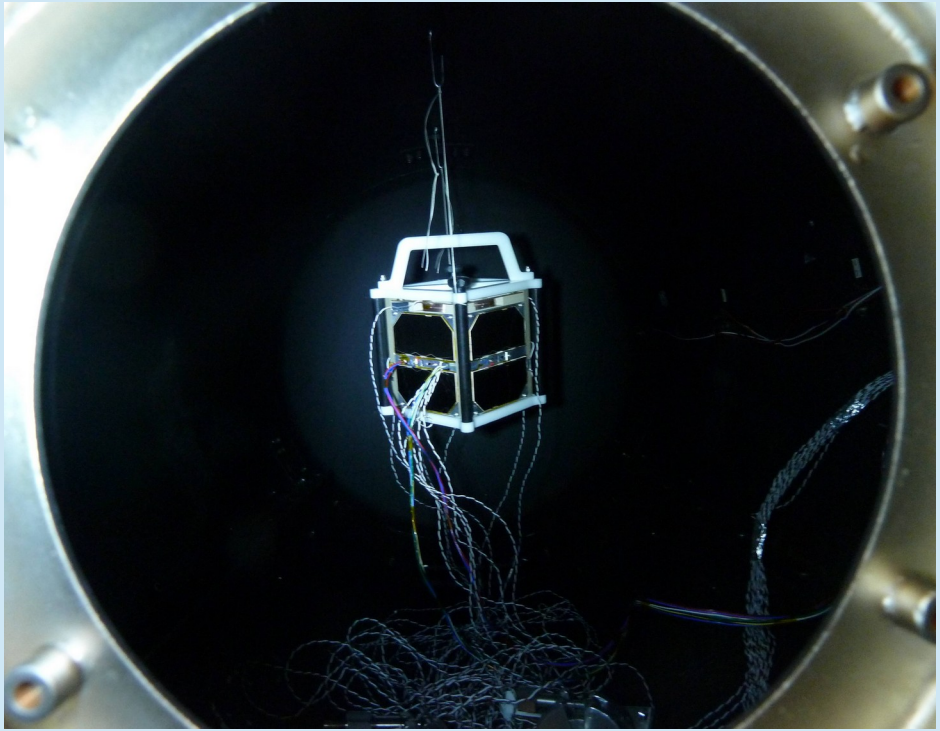
# Who/What - Are AAU Satlab?

- Designed and built at AAU!
- Tasks suiting every interest.
  - Antennas
  - BMS/Power supply
  - Radio
  - Control systems
  - Mission Control / Ground station
  - Payload(s) and more!



# Who/What - Are AAU Satlab?

- Long-Term project!
  - Target launch 2026
- Be a part of the whole process!
  Design → Launch → Contact
- Gain relevant experience to work in the space sector!



# AAUSAT Github.

- Current structure.
- Old structure of AAUSAT3 / AAUSAT 4 repositories.