

# Sustainable Software Development in an Academic Setting

4th International Symposium on Research and Education of Computational Science (RECS)  
University of Tokyo, October 1<sup>st</sup>, 2019

Hartwig Anzt, Terry Cojean, Thomas Grützmacher, Pratik Nayak, Mike Tsai  
Steinbuch Centre for Computing (SCC)



These slides are available under: <https://hartwiganzt.github.io/slides/SustainableSoftwareDevelopment.pdf>

# What we cover today

- *Versioning systems*
- *Git workflow*
- *Git hosting sites*
- *Continuous Integration (CI)*
- *GitLab runners*
- *Automated Testing*
- *Unit Testing with Googletest*
- *Software Documentation with Doxygen*

*To interactively participate in this course, you need a GitLab account.*

- *Please create an account (choose your name carefully!)*
- *Please log in*
- *<https://gitlab.com/>*



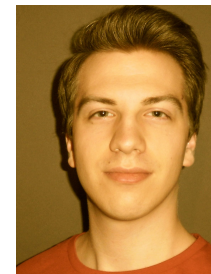
For complete interactive participation (on Linux Systems):

```
apt-get install git cmake g++ gcovr
```

*Interactive webinar with the help of*



Terry Cojean



Thomas  
Grützmacher



Pratik Nayak



Tobias Ribizel



Mike Tsai

# What is a version control system?

---

Version control system for tracking changes in computer files.

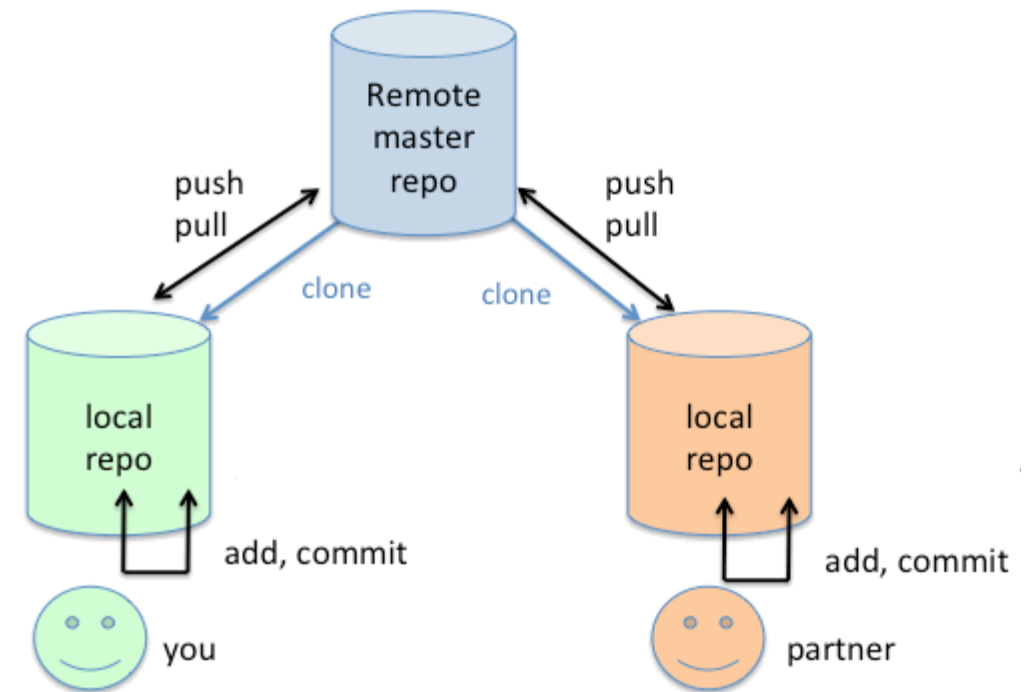
# What is a version control system?

Version control system for tracking changes in computer files.



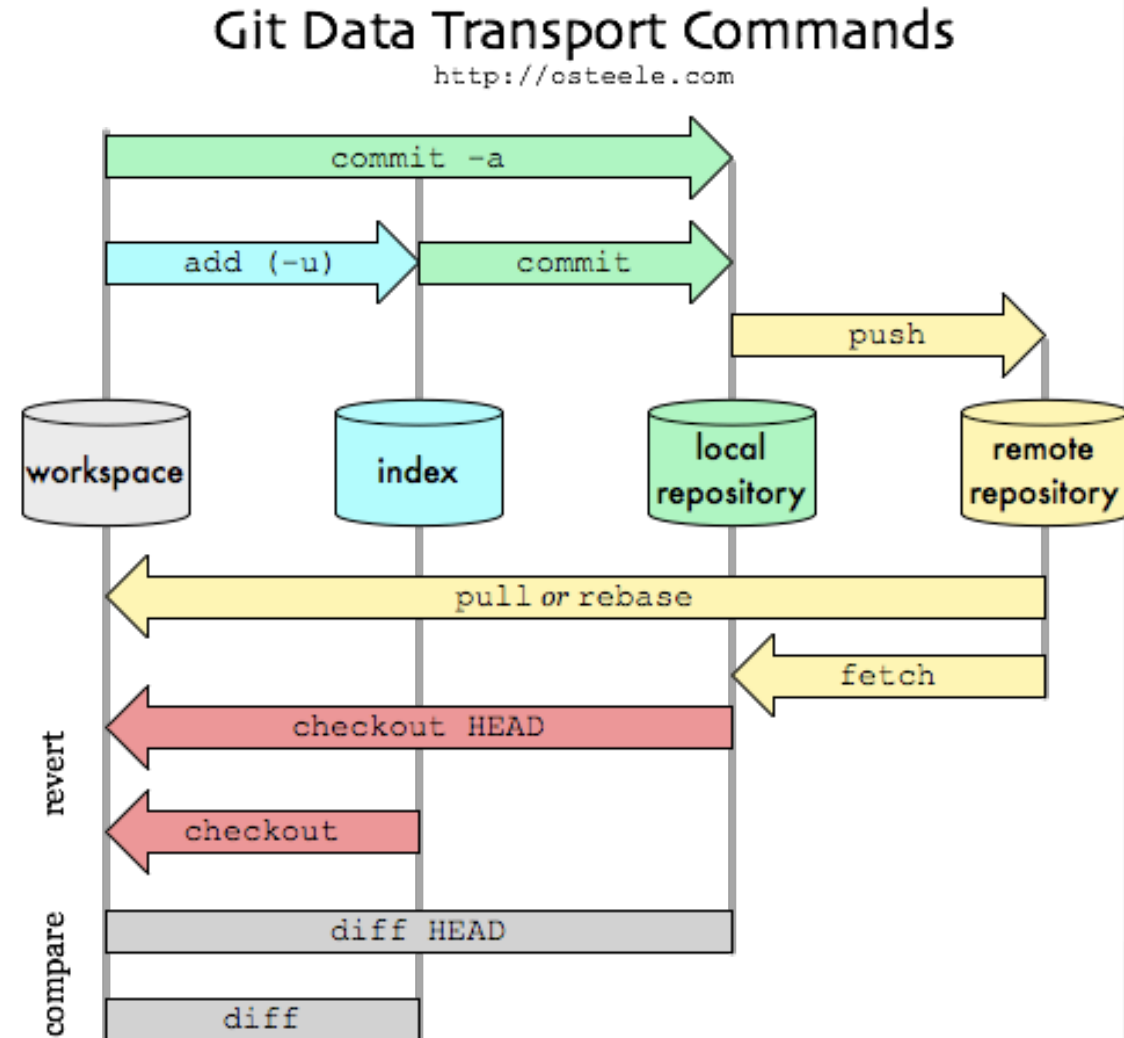
- Distributed version control
- Coordinates work between multiple developers
- Who made what changes and when
- Revert back at any time
- Local and remote repos

- ✓ Keeps track of code history
- ✓ Takes “snapshots” of your files
- ✓ You decide when to take a snapshot by making a “commit”
- ✓ You can visit any snapshot at any time



# Git Versioning System

- `git init` // Initialize local git repository
- `git add *files` // Add file(s) to snapshot
- `git status` // Check changes not yet in the snapshot
- `git commit *files` // Take snapshot (commit changes)
- `git commit -m 'put a comment on this commit' *files`



Git Cheat Sheet: <https://www.git-tower.com/blog/git-cheat-sheet/>

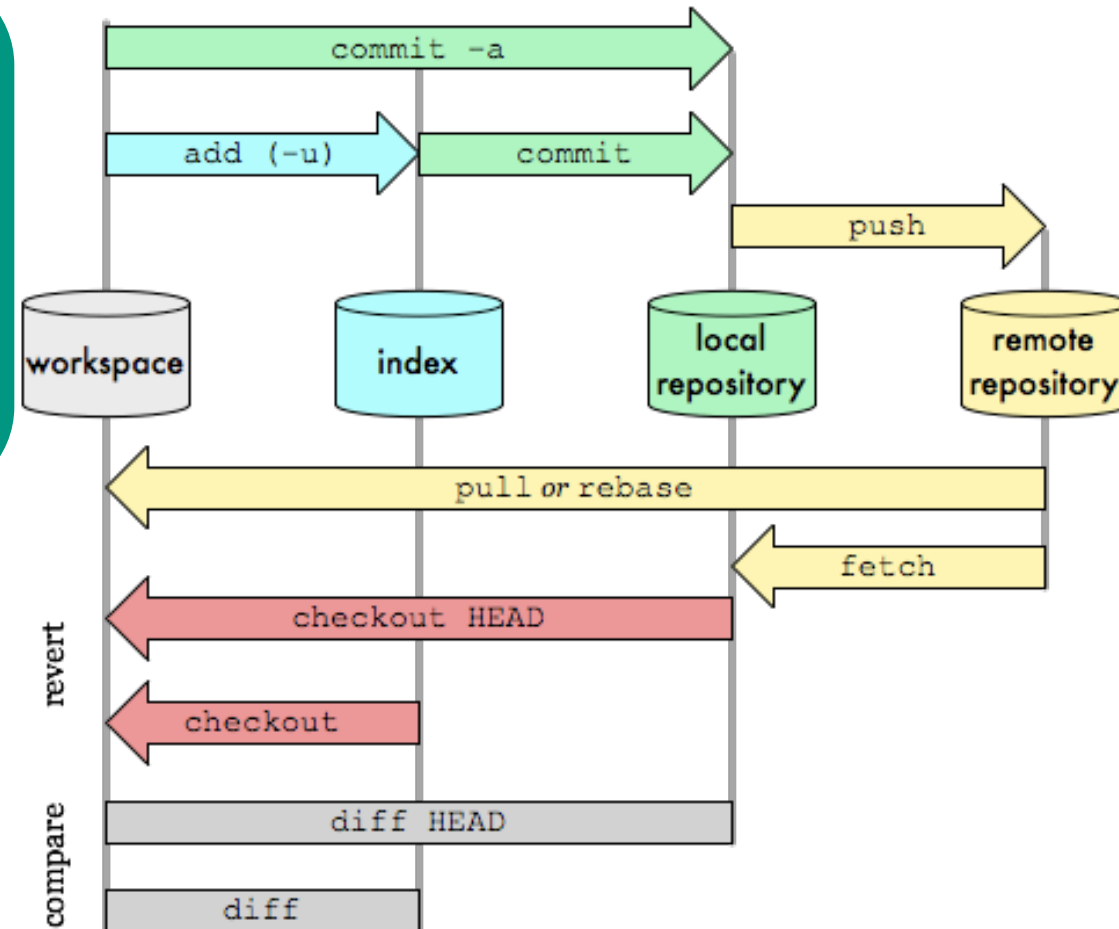
# Git Versioning System

- `git init` // Initialize local git repository
- `git add *files` // Add file(s) to snapshot
- `git status` // Check changes not yet in the snapshot
- `git commit *files` // Take snapshot (commit changes)
- `git commit -m 'put a comment on this commit' *files`

Local Repository

## Git Data Transport Commands

<http://osteele.com>



Git Cheat Sheet: <https://www.git-tower.com/blog/git-cheat-sheet/>

# Git Versioning System

- `git init` // Initialize local git repository
- `git add *files` // Add file(s) to snapshot
- `git status` // Check changes not yet in the snapshot
- `git commit *files` // Take snapshot (commit changes)
- `git commit -m 'put a comment on this commit' *files`

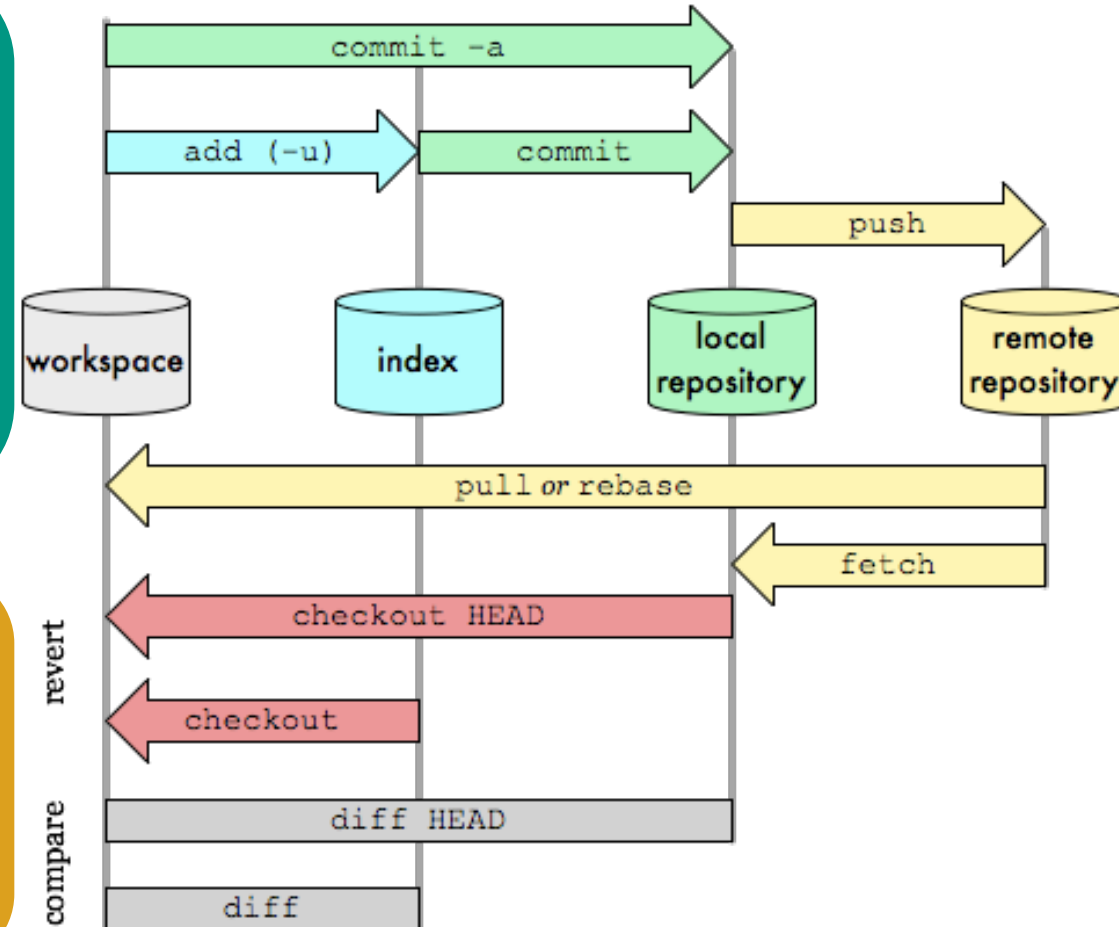
Local Repository

- `git push` // Push local snapshots to remote repo
- `git pull` // Get latest snapshot from remote repo
- `git clone *path/to/repo` // Clone an existing remote repository

Remote Repository

## Git Data Transport Commands

<http://osteele.com>

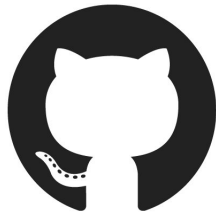


Git Cheat Sheet: <https://www.git-tower.com/blog/git-cheat-sheet/>

# Git Hosting Sites

They offer the environment for the remote repository.

- GitHub
- GitLab
- Bitbucket
- ...



*We may just choose GitLab for this course*

- *Please create an account (choose your name carefully!)*
- *Please log in*
- *<https://gitlab.com/>*

[https://en.wikipedia.org/wiki/Comparison\\_of\\_source-code-hosting\\_facilities](https://en.wikipedia.org/wiki/Comparison_of_source-code-hosting_facilities)

| Name         | Code review         | Bug tracking           | Web hosting         | Wiki | Translation system | Shell server | Mailing List | Forum              | Personal branch | Private branch      | Announce | Build system  | Team    | Release Binaries    | Self-hosting  |
|--------------|---------------------|------------------------|---------------------|------|--------------------|--------------|--------------|--------------------|-----------------|---------------------|----------|---|---------|---------------------|---|
| Assembla     | Yes <sup>[20]</sup> | Yes                    | Yes                 | Yes  | Yes                | No           | No           | No                 | Yes             | Yes <sup>[21]</sup> | Yes      | Yes   | Yes     | Unknown             | No  |
| Bitbucket    | Yes <sup>[22]</sup> | Yes <sup>[a]</sup>     | Yes <sup>[23]</sup> | Yes  | No                 | No           | No           | No                 | Yes             | Yes <sup>[b]</sup>  | No       | Yes <sup>[24]</sup>   | Yes     | No <sup>[25]</sup>  | Commercially (BitBucket Server formerly Stash) <sup>[c]</sup> |
| Buddy        | Yes                 | Yes                    | No                  | No   | No                 | No           | Yes          | Yes                | Yes             | Yes                 | Yes      | Yes <sup>[d]</sup>  | Yes     | Yes                 | Yes   |
| CloudForge   | Unknown             | Yes                    | Yes                 | Yes  | No                 | No           | No           | No                 | Unknown         | Unknown             | Unknown  | Unknown   | Unknown | Unknown             | No  |
| GitHub       | Yes <sup>[26]</sup> | Yes <sup>[27][e]</sup> | Yes <sup>[28]</sup> | Yes  | No                 | No           | No           | No                 | Yes             | Yes                 | Yes      | 3rd-party (e.g. Travis CI, Appveyor and others) <sup>[29]</sup> | Yes     | Yes                 | Commercially (GitHub Enterprise)                              |
| GitLab       | Yes <sup>[30]</sup> | Yes                    | Yes <sup>[31]</sup> | Yes  | No                 | No           | No           | No                 | Yes             | Yes                 | Yes      | Yes <sup>[32]</sup>   | Yes     | Yes <sup>[33]</sup> | Yes <sup>[f]</sup>  |
| GNU Savannah | Yes <sup>[34]</sup> | Yes                    | Yes                 | No   | No                 | Yes          | Yes          | No <sup>[35]</sup> | No              | No                  | Yes      | No  | Yes     | Unknown             | Yes   |
|              |                     |                        |                     |      |                    |              |              |                    |                 |                     |          | Yes with  |         |                     |   |



# Git Hands-On

1. We create an Account at GitLab and log in.
2. I create a project on GitLab ( \git@gitlab.com:hanzt/recs )
3. I add a first source file and make it a public repository
4. You all clone or the project:
5. You add your name to the local version of contributors.txt
6. You check your changes:

```
git diff
```

7. You commit your local changes:

```
git commit -m 'add my name' contributors.txt
```

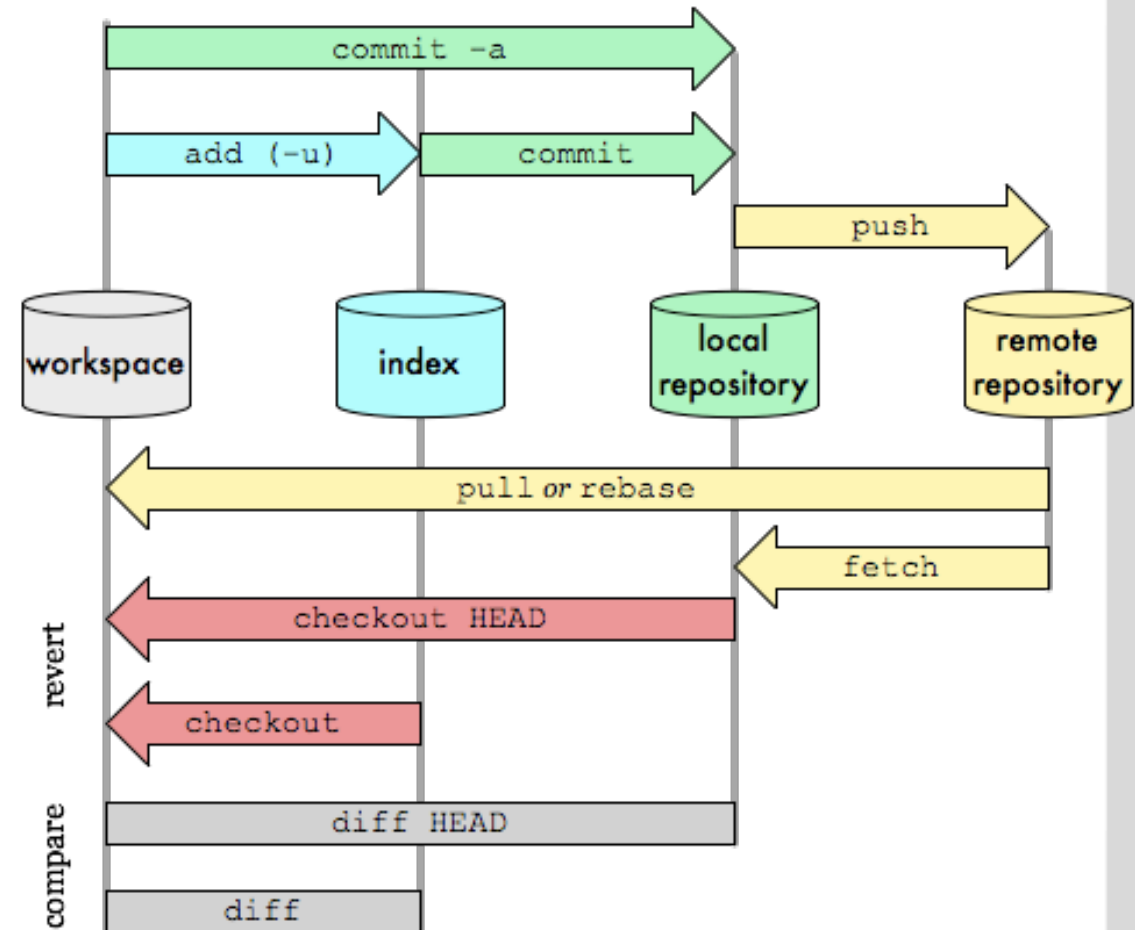
8. You push your local changes to the remote repository:

```
git push origin master
```

9. You fix "merge conflicts" ....

## Git Data Transport Commands

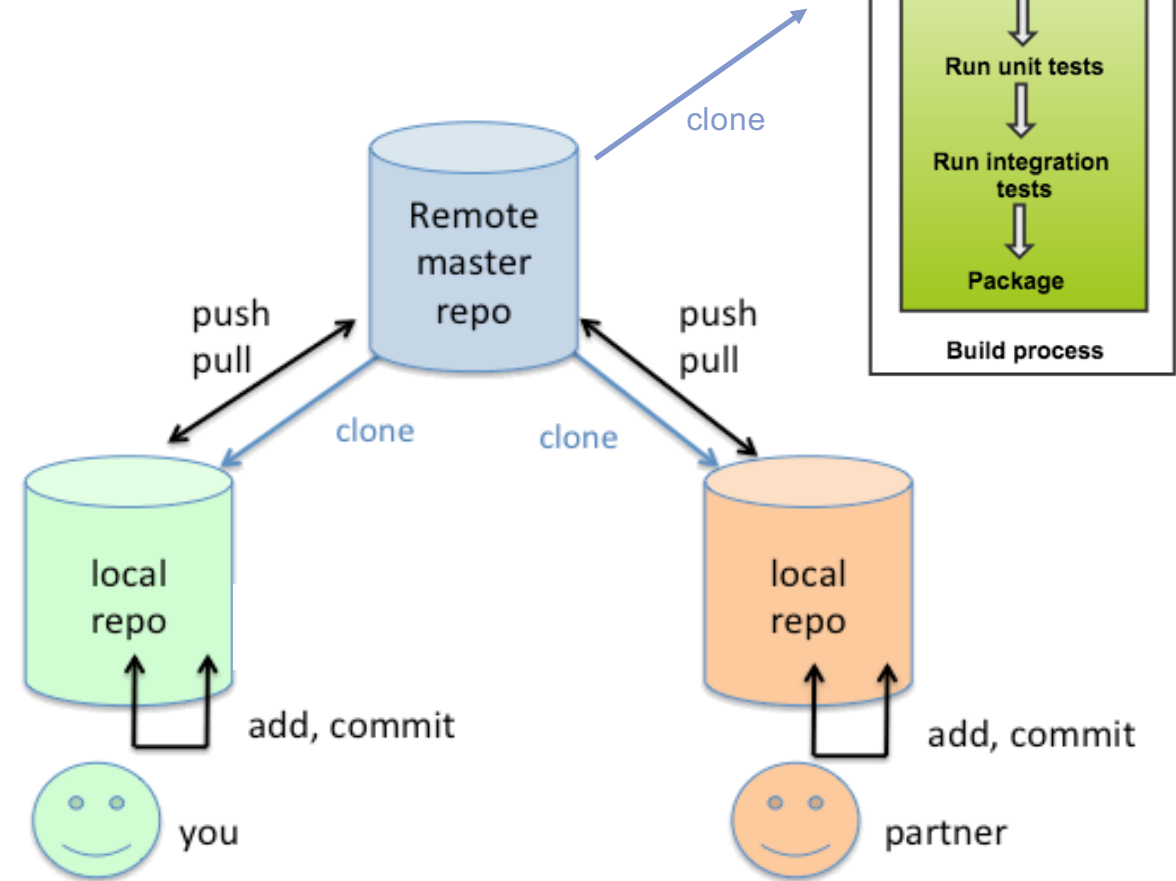
<http://osteele.com>



# Continuous Integration (CI)

Sometimes, someone introduces a bug that breaks the code...

- *How do you find out the code is broken?*
- *How do we find out who which code integration introduced the bug?*
- *How can we make sure everything works at any point in time?*



# Continuous Integration (CI)

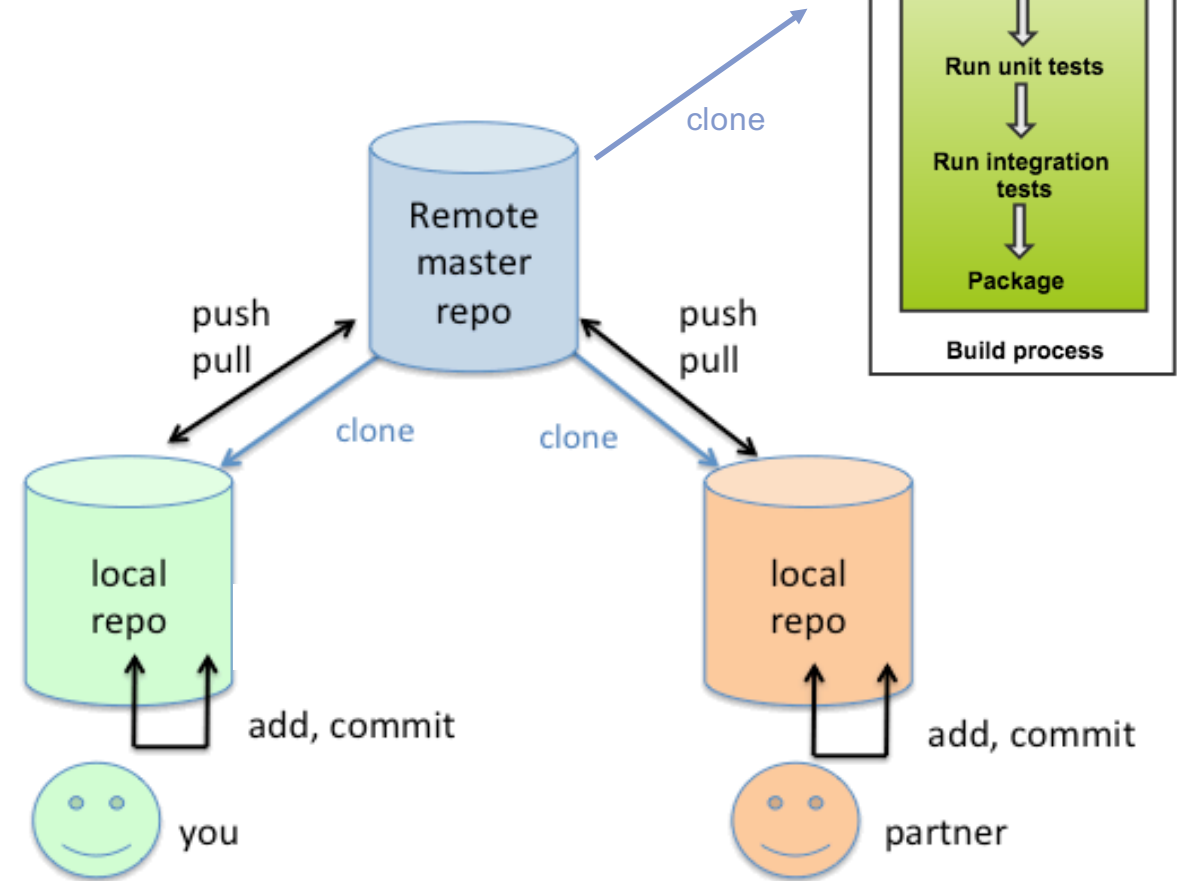
Sometimes, someone introduces a bug that breaks the code...

- *How do you find out the code is broken?*
- *How do we find out who which code integration introduced the bug?*
- *How can we make sure everything works at any point in time?*

*We need a mechanism that constantly checks the functionality of the master "branch".*

## Continuous Integration

- Sets up a pre-defined environment;
- Clones the remote repository on a server;
- Tries to compile and run all pre-defined tests;
- Reports the outcome;



# Continuous Integration on GitLab

- GitLab supports GitLab runners as CI feature.
- They can be configured via adding the file `.gitlab-ci.yml`  
<https://docs.gitlab.com/ee/ci/yaml/>
- GitLab runner are set up via web interface.

```
image: ubuntu:14.04

job:
  script:
    - apt-get update
    - apt-get install -y git cmake g++ gcovr
    - git submodule update --init --recursive
    - mkdir build
    - cd build
    - cmake ..
    - make -j
    - gcovr -r ../src/
```

Load image of OS

Update OS

Install packages needed:  
git cmake g++ gcovr

Create directory


Call CMake

Build library

gcov checks unit test coverage

Hartwig Anzt > hekksagon\_test > Pipelines > #51657453

✓ passed

Pipeline #51657453 triggered 24 minutes ago by  Hartwig Anzt

example `.gitlab-ci.yml`

# Software Testing

Automated testing:

The practice of writing code to test the code,  
and then run those tests in an automated fashion.

Production Code



Test Code

# Software Testing

Automated testing:

The practice of writing code to test the code,  
and then run those tests in an automated fashion.

Production Code



Test Code

**Many. Easy-to-write. Execute Fast.**

**Unit tests:**

Check the functionality and validity of each  
building block without its external dependencies.

- Track down bugs

# Software Testing

Automated testing:

The practice of writing code to test the code,  
and then run those tests in an automated fashion.

Production Code



Test Code

**Fewer. Dependencies.**

## Integration tests:

Check the applications functionality with its external dependencies.

- Give more confidence in complex application.
- Tests units/classes as whole.

**Many. Easy-to-write. Execute Fast.**

## Unit tests:

Check the functionality and validity of each  
building block without its external dependencies.

- Track down bugs

# Software Testing

Automated testing:

The practice of writing code to test the code, and then run those tests in an automated fashion.

Production Code



Test Code

**Few.  
Complex.**

## End-to-End tests:

Check the applications functionality with its external dependencies and user input.

- Test complete workflow and application interaction.
- Very slow and brittle.

**Fewer. Dependencies.**

## Integration tests:

Check the applications functionality with its external dependencies.

- Give more confidence in complex application.
- Tests units/classes as whole.

**Many. Easy-to-write. Execute Fast.**

## Unit tests:

Check the functionality and validity of each building block without its external dependencies.

- Track down bugs



# Unit tests with Googletest



- Framework to facilitate unit testing.
- <https://github.com/google/googletest>

*Let's do an example.*

Much of this material is taken from **Nikolaos Pothitos**

<http://cgi.di.uoa.gr/~pothitos/>

<https://github.com/pothitos/gtest-demo-gitlab>

```
[=====] Running 1 test from 1 test case.
[-----] Global test environment set-up.
[-----] 1 test from AddTest
[ RUN    ] AddTest.TwoAndTwo
test2.cc:6: Failure
           Expected: Add(2, 2)
           Which is: 4
To be equal to: 5
[  FAILED ] AddTest.TwoAndTwo (0 ms)
[-----] 1 test from AddTest (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (1 ms total)
[ PASSED ] 0 tests.
[  FAILED ] 1 test, listed below:
[  FAILED ] AddTest.TwoAndTwo

1 FAILED TEST
```

# Software Documentation

*Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D.*

- Automated code documentation tool.
- Converts comments in source code into documentation.
- HTML-oriented.
- <http://www.doxygen.nl/>
- <https://github.com/doxygen/doxygen.git>

## My Project

|            |             |               |        |
|------------|-------------|---------------|--------|
| Main Page  | Classes     | Files         | Search |
| Class List | Class Index | Class Members |        |
| Foo        | Foo         |               |        |

Public Types | List of all members

### Foo::Foo Class Reference

#### Public Types

enum **Foo**  
**Foo** enum, possible ways to foo.

The documentation for this class was generated from the following file:

- **Foo.h**

Generated on Thu Dec 6 2012 15:38:12 for My Project by  1.8.2



Example:

<https://www.dealii.org/current/doxygen/deal.II/classTorusManifold.html>

# References & Further Reading



<https://bssw.io/>

- <https://www.git-tower.com/blog/git-cheat-sheet/>
- <https://github.com/google/googletest>
- <https://github.com/pothitos/gtest-demo-gitlab>
- <https://docs.gitlab.com/ee/ci/yaml/>
- <http://www.doxygen.nl/>

## Better Scientific Software (BSSw)

Scientific software has emerged as an essential discipline in its own right. Because computational models, computer architectures, and scientific software projects have become extremely complex, the Computational Science & Engineering (CSE) community now has a unique opportunity—and an implicit mandate—to address pressing challenges in scientific software productivity, quality, and sustainability.

GET ORIENTED

Communities Overview

Site Overview

Intro to CSE

Intro to HPC

These slides are available under: <https://hartwiganzt.github.io/slides/SustainableSoftwareDevelopment.pdf>