Scientific Computing Using Python

Development Practices and Tools

14th of June, 2023

Thomas Arildsen
tari@its.aau.dk

CLAAUDIA
Aalborg University

Version Control
Agenda

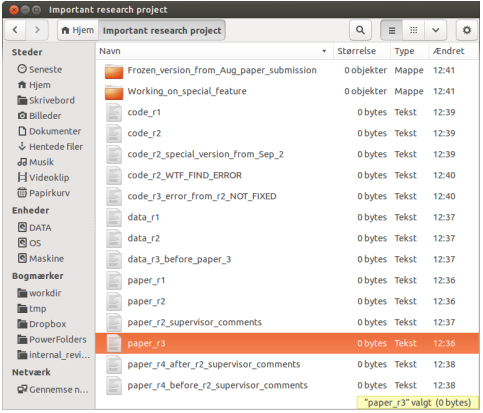
Introduction

Introduction
What is version control?

- Version Control Systems let you keep track of different versions of files.
- Very often used for software development where the files are individual source files of the software.
- The files can be any kind in general. Also documents, images etc. etc.

Introduction
Why version control?

You probably all know this scenario?



Introduction
Why version control?

Managing different file versions manually can be frustrating, confusing, and begging for mistakes!

- Version control formalises the process of managing different versions of files and takes care of the practicalities for you.
- Basically makes it possible (and easy!) for you to figure out "who did what when?"
- For example: when your code suddenly starts failing and you simply cannot figure out why, you can easily view the differences between your current and your last known "good" version.
- (Can) serve the secondary purpose of providing backup too.

Introduction
An example

An example of version control that you may not think so much about on a day-to-day basis:

- Dropbox – automatic storage of earlier versions of a file.

Advantages:

- Works behind the scenes – you do not have to think about it at all (until you need an older version of a file).

Drawbacks:

- Stores every single little change.
- No user-definable meta-data.
- Difficult to find your way around a long history of file versions.

Introduction
Intended learning outcomes

After today's course you will have knowledge about:

- What version control is.
- Benefits you get from using version control.
- Know the main differences between different types of version control.

You will be able to:

- Create repositories.
- Track versions of your software, documents etc. in a version control system.

Version Control Types

Version Control Types

Basic Version Control Concepts

Version control is usually centered around the concept of a *repository*.

- This is where all the versions of your files are kept.
- Think of it as a bank vault where you deposit your precious files.

Version Control Types
Basic Version Control Concepts

Version control is usually centered around the concept of a *repository*.

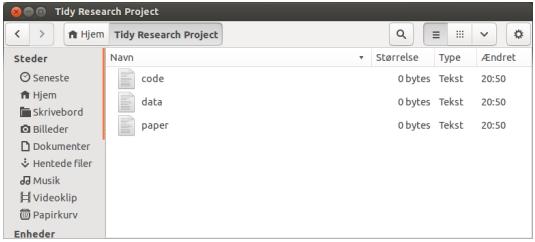
- This is where all the versions of your files are kept.
- Think of it as a bank vault where you deposit your precious files.

When changing and working on your files, you do this in a *working copy* of the files.

- You do not work on the files in the repository directly – you do that in the working copy.
- You tell the VCS to get the files from the repository for you when you need to work on them.
- . . . and you tell it to store them back there when you want to store a new version of them.

Version Control Types
In a Nutshell

Basically, version control lets you keep your working copy like this. Everything else is "under the hood".



Centralised Version Control

Centralised Version Control

Centralised Version Control

Centralised Version Control represents the "old days" of version control.

- There is *one* central repository – hence centralised.
- You check files directly out of / into this repository.
- Several users can work on each their own working copy of the repository.
- They will have to coordinate their changes whenever they check something back into the repository.

Centralised Version Control
Subversion

Subversion[1] is an example of a centralised version control system.

- For a great reference on Subversion, see [CFP11] (freely available online).

[1] `http://subversion.apache.org`

Centralised Version Control

Individual Use

- Great for individual use, especially if you simply keep the repository on your local machine.
- That, however, somewhat defies the secondary purpose of serving as a backup too.

Centralised Version Control
Collaborative Use

- Useful for collaborative use too.
- Some drawbacks, though:
  - You need connection to the central repository every time you want to commit something.
  - Makes it difficult to work offline.
  - Makes committing and updating slower.

Distributed Version Control

Distributed Version Control

Distributed Version Control

Distributed version control represents the newest generation of version control systems.

- There is *not* one central repository – hence distributed.
- There are multiple copies of the repository which may be more or less synchronised.
- Several users can work on working copies of each their own copy of the repository.
- No need to coordinate with others when checking something into your own version of the repository.
- This is not done until you eventually synchronise your copy of the repository with someone else's.

Distributed Version Control
Git etc.

A very popular example of distributed version control in especially the open source community is Git[2].

- Very flexible, endlessly configurable, some would say a bit too much.
- In my opinion, one detail making Git better than Subversion is its multi-stage commit process with a "staging area" that gives a better overview of what you are about to commit.
- For a great reference on Git, see [Cha] (freely available online).

Alternative:

Mercurial Quite similar to Git but less complicated and flexible. Not as widely used as Git `http://mercurial.selenic.com/`.

[2] `http://git-scm.com/`

Distributed Version Control
A Word on Workflows

Workflow

Workflows are something you often hear about in relation to distributed version control.

- A workflow is a convention on how users should use the repository.
- Not directly something enforced by the VCS itself.
- Rather, it is a policy adopted by its users.
- It helps make sense of what to use different branches for etc.
- We will not have time to cover the details here, but you should know that different workflow models exist and you might want to look further into them.
- Overview of some popular workflow models here: `https://www.atlassian.com/git/tutorials/comparing-workflows`.

Distributed Version Control
Individual Use

- Not much advantage over centralised systems such as Subversion.
- If you keep your repository online, distributed systems offer faster repository access to your local repository, relying only on slower network connection when pushing/pulling against remote copy.

Distributed Version Control
Collaborative Use

- Better suited for collaborative use due to the possibility for more independent workflows of the individual users.
- If you really want one centralised "main" repository, this is only enforced by user policy in distributed systems, whereas by construction in centralised systems.

Example and Exercise

Example and Exercise

Example/Exercise

- I am going to walk you through basic version control with Git on-screen.
- Alongside, the exercise is for you to:
  - create a repository of your own and add something to it.
  - clone it and edit files in the new copy.
  - commit changes.
  - push changes to the original repository.

Online Repositories

Online Repositories
Where to Host Your Repository

- In addition to keeping your repository on your own machine or your university's or company's network, there are also several options for hosting your repository on the Internet.
- The available services are very popular with open source projects where the code is openly accessible. This is also relevant for researchers for code accompanying publications for the sake of reproducible research.
- There are also possibilities of private repositories for the files you do not want to share with everyone.

Online Repositories
GitHib

GitHub is probably the most popular repository hosting service.

- As the name suggests, this is based on Git.
- Unlimited *public* repositories.
- Private repositories (paid) available as well; free for students; academics can apply to get private repositories for free for a limited time.

Online Repositories
AAU ITS Git

AAU hosts our own Git platform AAU Git more or less similar to GitHub.

- Based on the Git platform software Gitea.
- Controlled by AAU.
- Unlimited public *and* private repositories.
- Only for AAU users.

Online Repositories
NeiC Git

The Nordic e-Infrastructure Collaboration (NeIC) hosts a Git platform also similar to GitHub.

- Based on the Git platform software Gitlab.
- Controlled by Nordic universities through NeIC.
- Unlimited public *and* private repositories.
- For researchers at Nordic universities.

Online Repositories
Citable Code Repositories

What if you deposit your repository publically online, can you make it easier to cite in publications?

- Actually, yes.
- Convenient ways to deposit and assign a DOI to a particular version of your repository:

  Zenodo (publicly funded) provides a service for automatically importing a repository from GitHub, see: `https://guides.github.com/activities/citable-code/`

  figshare (commercial service provider) lets you link to a version of your repository from GitHub (for example) as a "code" publication and assigns a DOI to it.

Summary

Summary

In today's lecture we have seen:

- what version control systems are.
- the main differences between centralised and distributed version control.
- an example of creating and working with a Git repository.
- options for hosting version control repositories online.
- options for depositing your code publicly with a DOI for consistent citation.

References

[CFP11] Ben Collins-Sussman, Brian W. Fitzpatrick, and Michael Pilato. *Version Control with Subversion*. 2011. URL: `http://svnbook.red-bean.com/` (visited on 05/20/2014).

[Cha] Scott Chacon. *Pro Git*. URL: `http://git-scm.com/book` (visited on 05/21/2014).