Imperial College
London

Introduction to software configuration management

Dr David A. Ham

Department of Computing, Imperial College London
Grantham Institute for Climate Change, Imperial College London
david.ham@imperial.ac.uk

Software configuration management topics

- Version control
- Building on different platforms
- Testing

Version control: getting the source right

- "Nobody knows which version of the code was used to create the plots in the paper, and now the reviewers want more runs!"

Version control: getting the source right

- "Nobody knows which version of the code was used to create the plots in the paper, and now the reviewers want more runs!"
- "Alice has a slope limiter in her version of foo.cpp, and Bob's version has a turbulence model, but I want a slope limiter and a turbulence model."

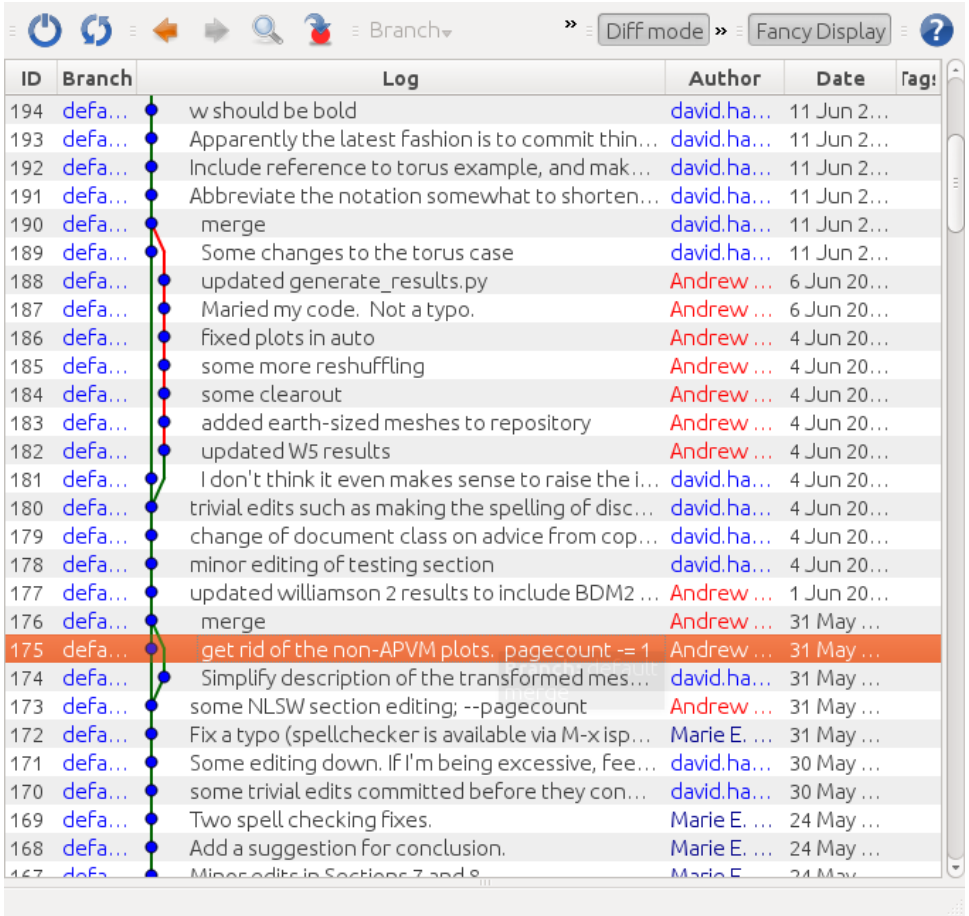
Version control: getting the source right

- "Nobody knows which version of the code was used to create the plots in the paper, and now the reviewers want more runs!"
- "Alice has a slope limiter in her version of foo.cpp, and Bob's version has a turbulence model, but I want a slope limiter and a turbulence model."
- "I think Charlie had that feature in a version on his laptop somewhere, but he's working for a bank now."

Version control: getting the source right

- "Nobody knows which version of the code was used to create the plots in the paper, and now the reviewers want more runs!"
- "Alice has a slope limiter in her version of foo.cpp, and Bob's version has a turbulence model, but I want a slope limiter and a turbulence model."
- "I think Charlie had that feature in a version on his laptop somewhere, but he's working for a bank now."
- "This used to work!"

The core idea behind a version control system



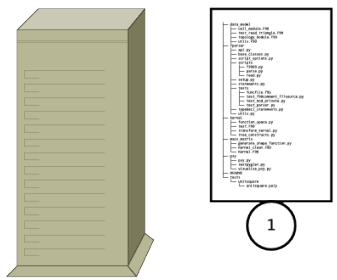
ID	Branch	Log	Author	Date	Tags
194	defa…	w should be bold	david.ha…	11 Jun 2…	
193	defa…	Apparently the latest fashion is to commit thin…	david.ha…	11 Jun 2…	
192	defa…	Include reference to torus example, and mak…	david.ha…	11 Jun 2…	
191	defa…	Abbreviate the notation somewhat to shorten…	david.ha…	11 Jun 2…	
190	defa…	merge	david.ha…	11 Jun 2…	
189	defa…	Some changes to the torus case	david.ha…	11 Jun 2…	
188	defa…	updated generate_results.py	Andrew …	6 Jun 20…	
187	defa…	Maried my code. Not a typo.	Andrew …	6 Jun 20…	
186	defa…	fixed plots in auto	Andrew …	4 Jun 20…	
185	defa…	some more reshuffling	Andrew …	4 Jun 20…	
184	defa…	some clearout	Andrew …	4 Jun 20…	
183	defa…	added earth-sized meshes to repository	Andrew …	4 Jun 20…	
182	defa…	updated W5 results	Andrew …	4 Jun 20…	
181	defa…	I don't think it even makes sense to raise the i…	david.ha…	4 Jun 20…	
180	defa…	trivial edits such as making the spelling of disc…	david.ha…	4 Jun 20…	
179	defa…	change of document class on advice from cop…	david.ha…	4 Jun 20…	
178	defa…	minor editing of testing section	david.ha…	4 Jun 20…	
177	defa…	updated williamson 2 results to include BDM2 …	Andrew …	1 Jun 20…	
176	defa…	merge	Andrew …	31 May …	
175	defa…	get rid of the non-APVM plots. pagecount -= 1	Andrew …	31 May …	
174	defa…	Simplify description of the transformed mes…	david.ha…	31 May …	
173	defa…	some NLSW section editing; --pagecount	Andrew …	31 May …	
172	defa…	Fix a typo (spellchecker is available via M-x isp…	Marie E. …	31 May …	
171	defa…	Some editing down. If I'm being excessive, fee…	david.ha…	30 May …	
170	defa…	some trivial edits committed before they con…	david.ha…	30 May …	
169	defa…	Two spell checking fixes.	Marie E. …	24 May …	
168	defa…	Add a suggestion for conclusion.	Marie E. …	24 May …	

A simple example, several co-authors work on the same paper.

The core ideas behind a version control system

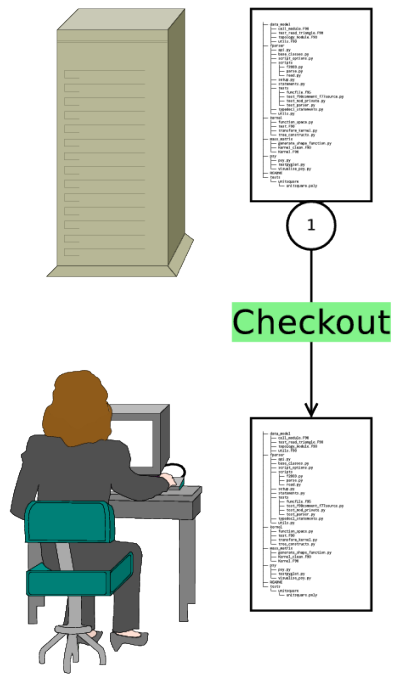
- Keeping each version of your files as they develop is partly automated.
- You can always get back to any version of your files.
- It's very easy to see the differences between different versions of files.
- The conflicts caused by multiple people editing the same files are minimised.

Centralised version control

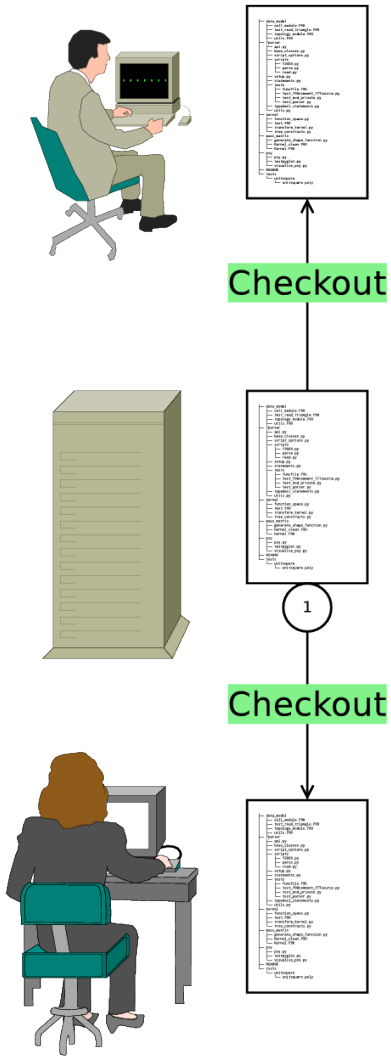


Centralised version control

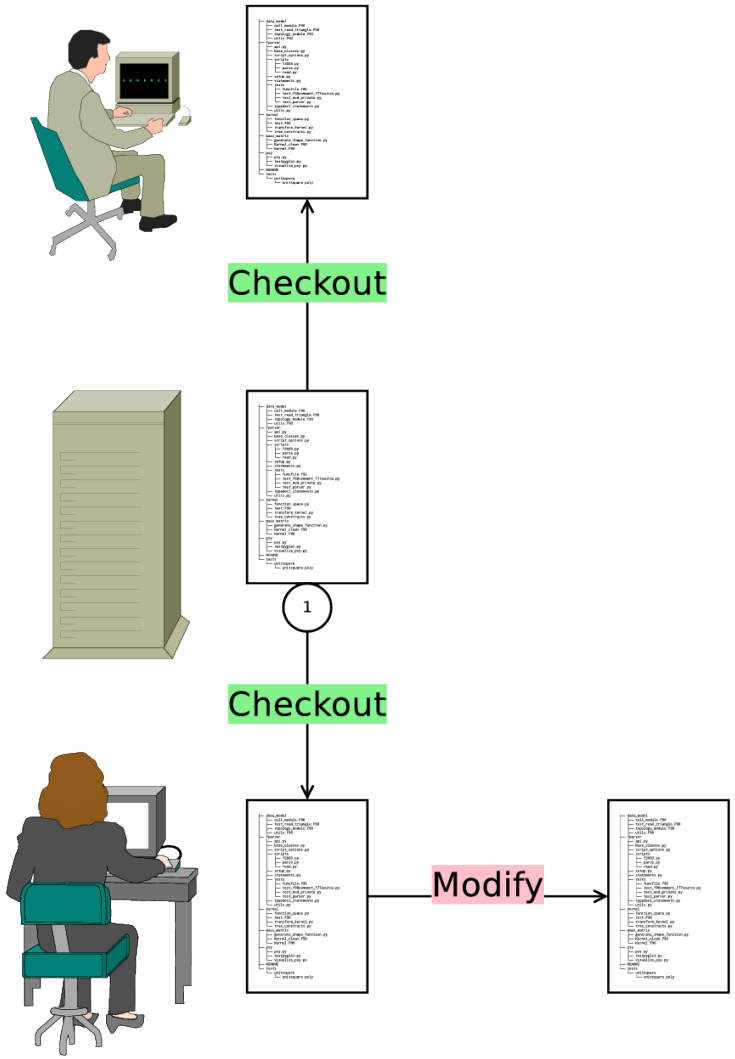
Centralised version control



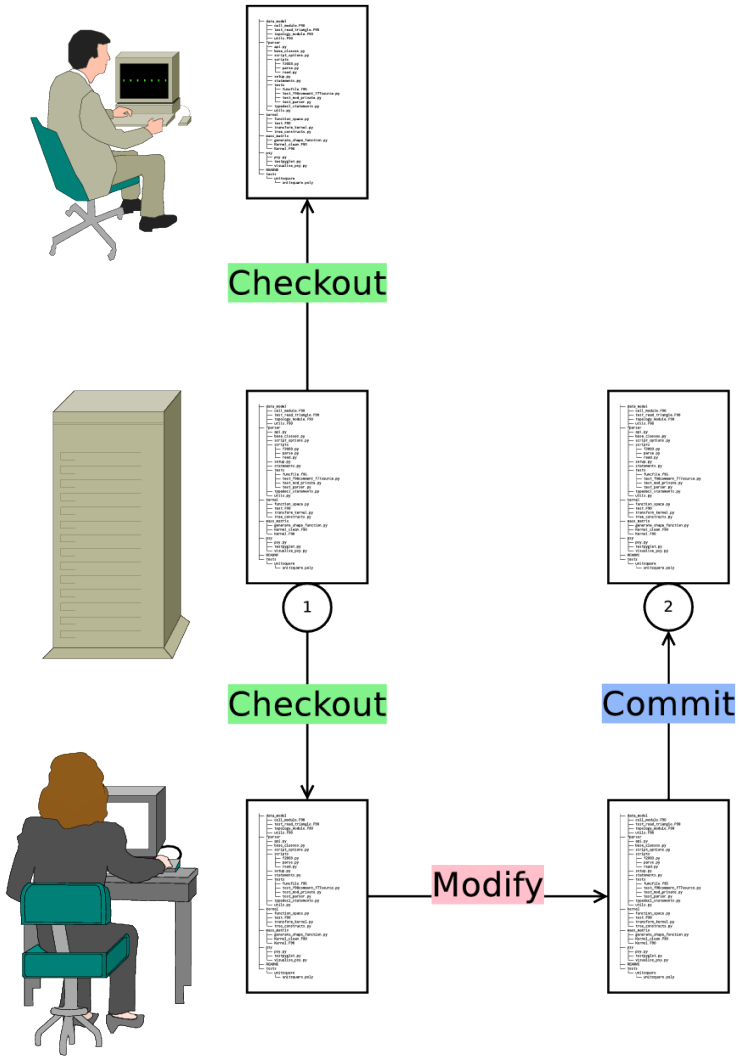
Centralised version control



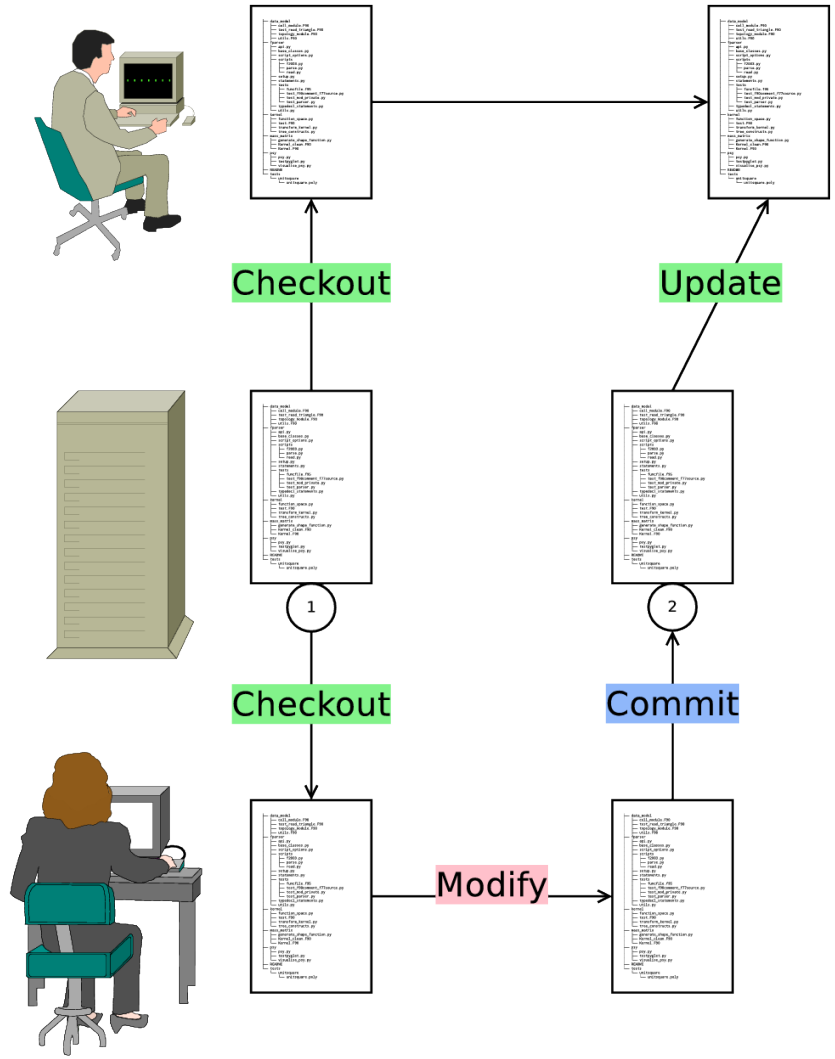
Centralised version control



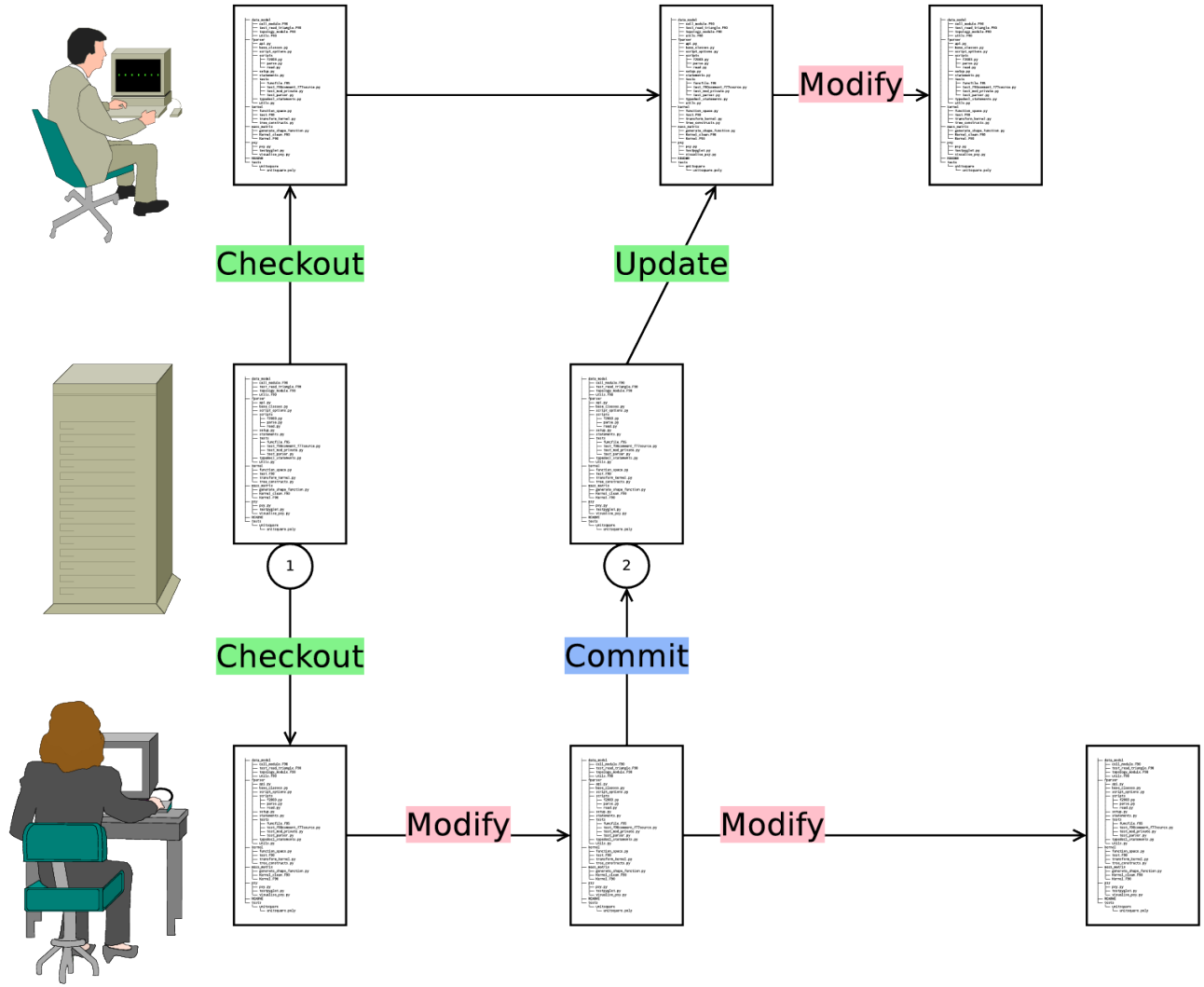
Centralised version control



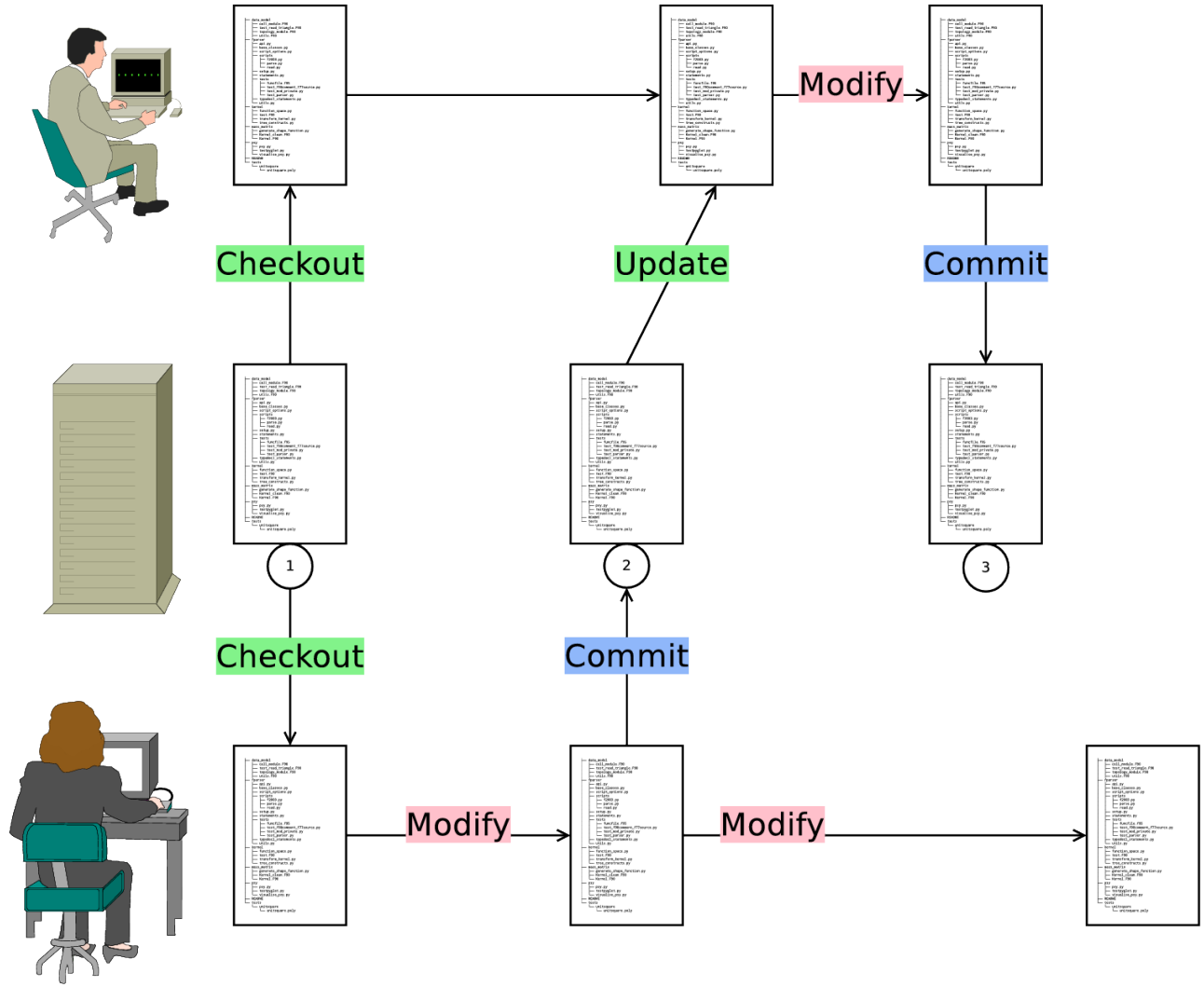
Centralised version control



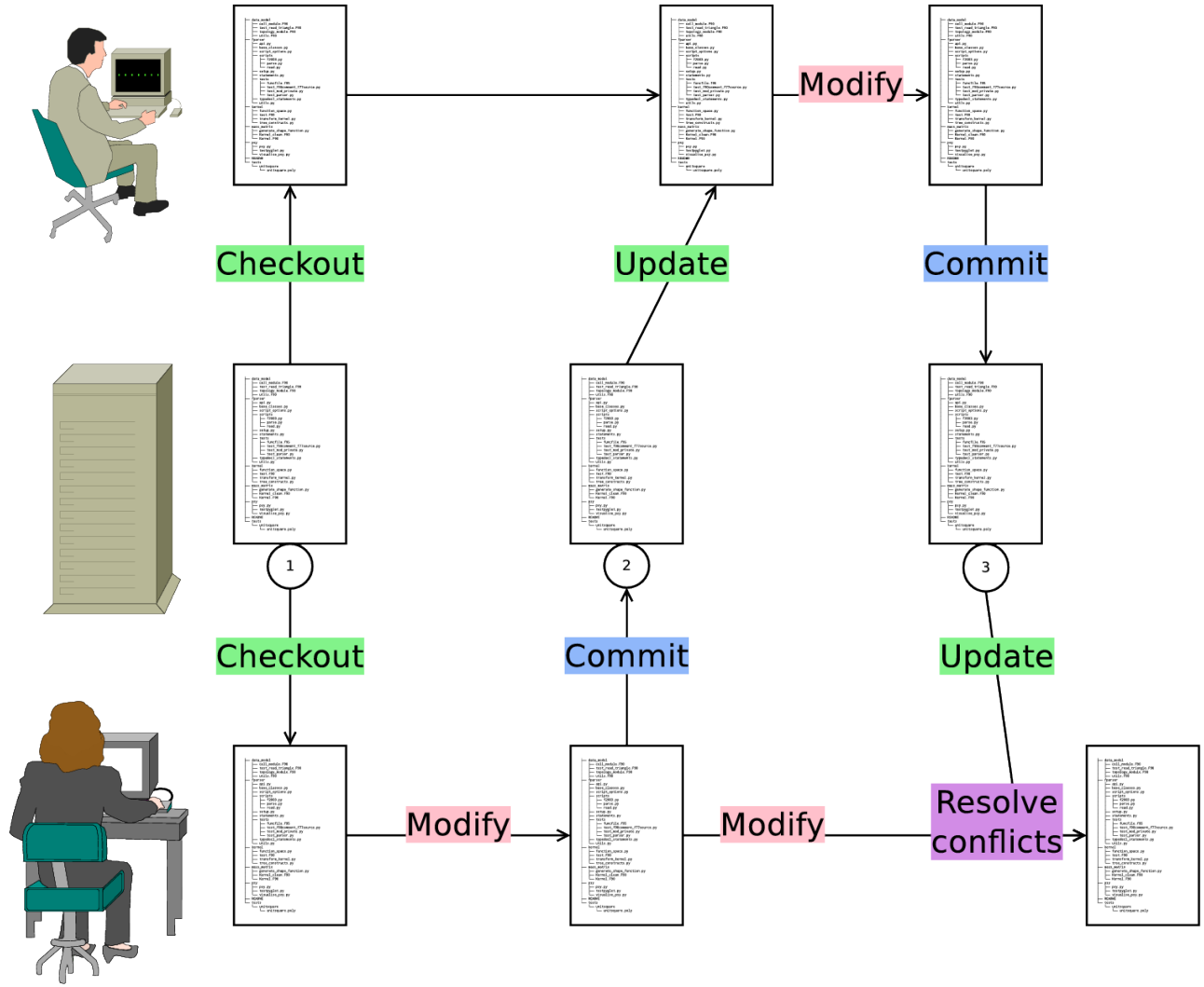
Centralised version control



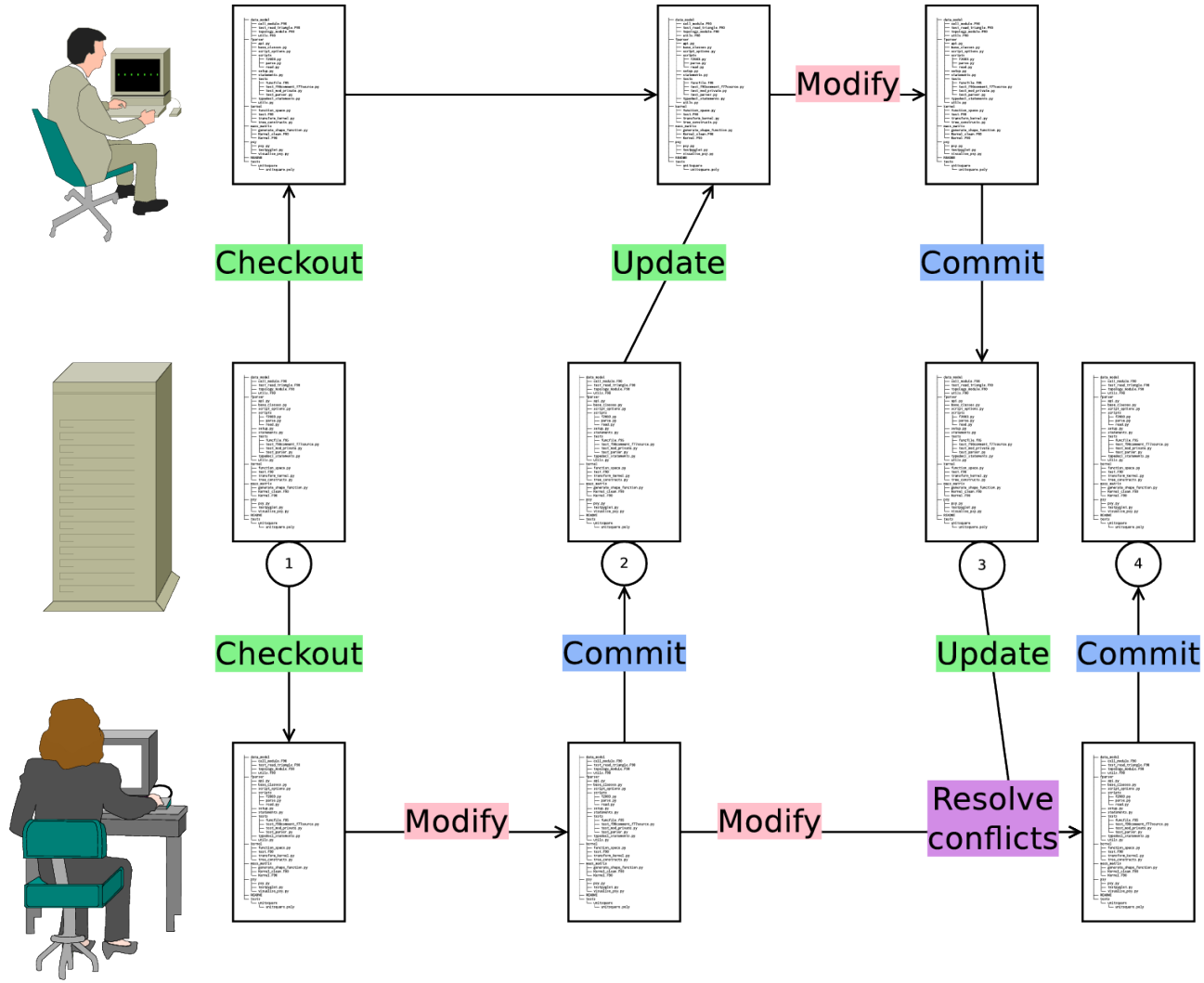
Centralised version control



Centralised version control



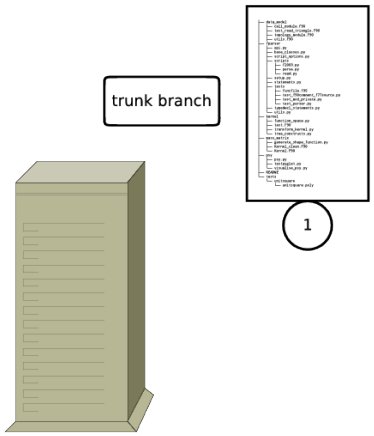
Centralised version control



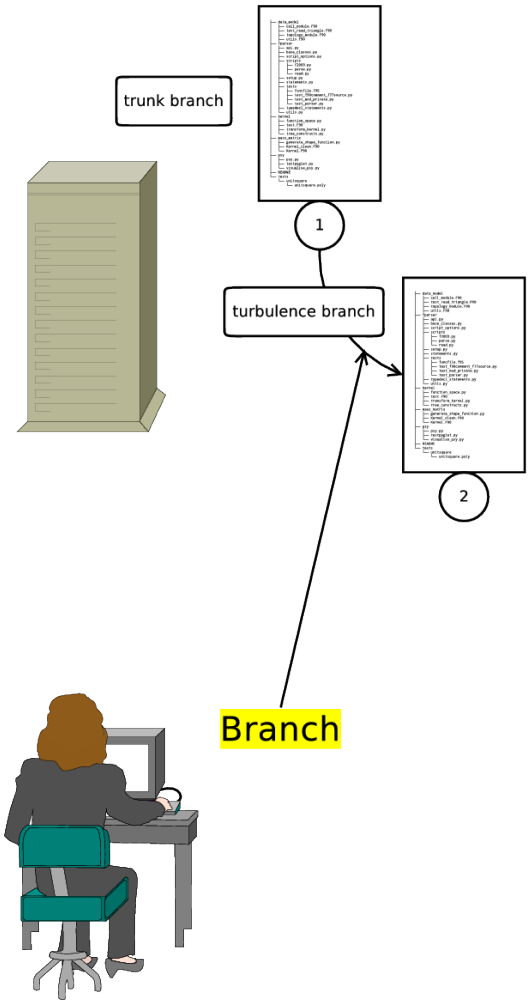
Branching

So far so good, but what if you want to make a more substantial change which should be spread over many commits?

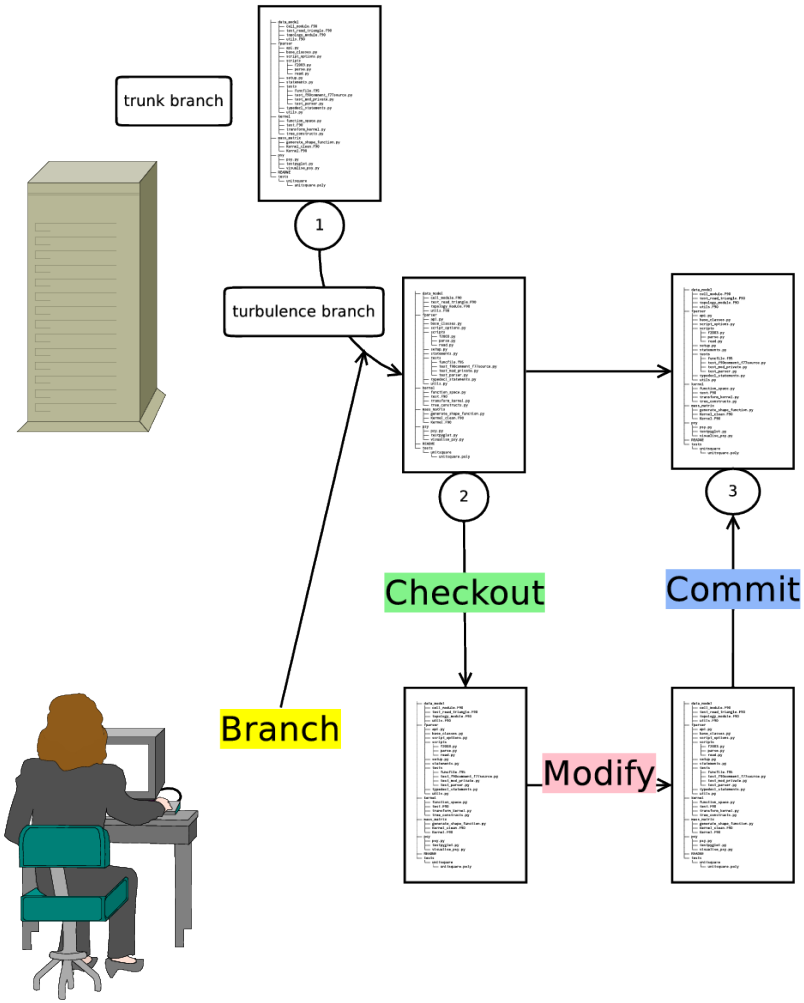
Making a feature branch in a centralised system



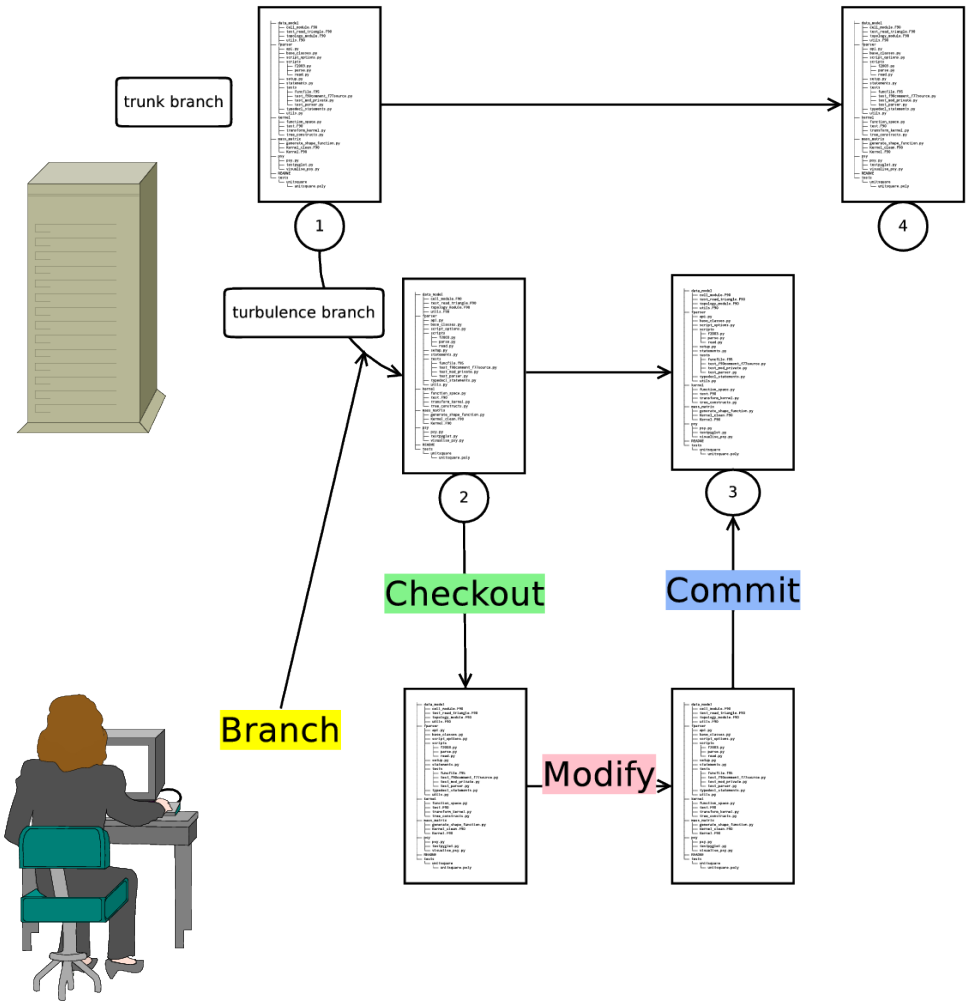
Making a feature branch in a centralised system



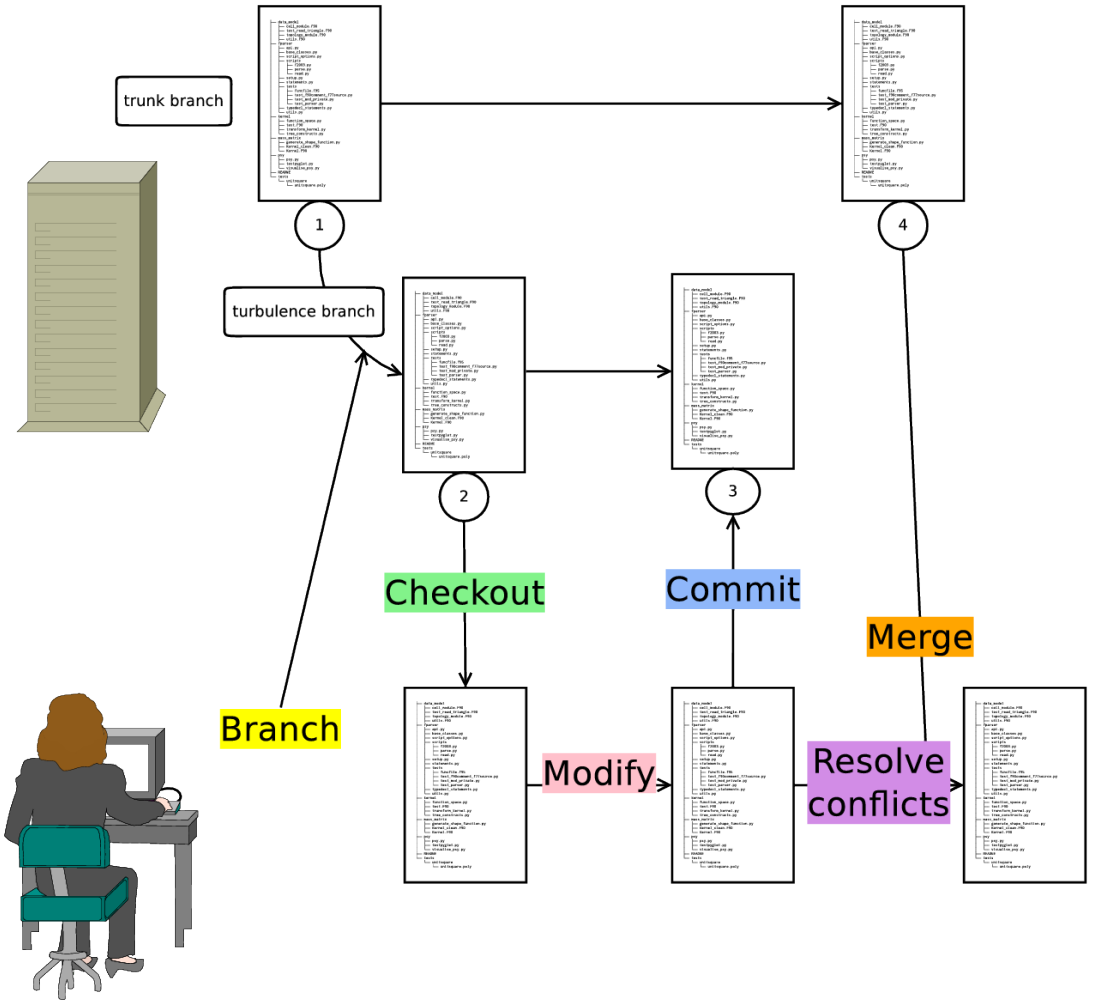
Making a feature branch in a centralised system



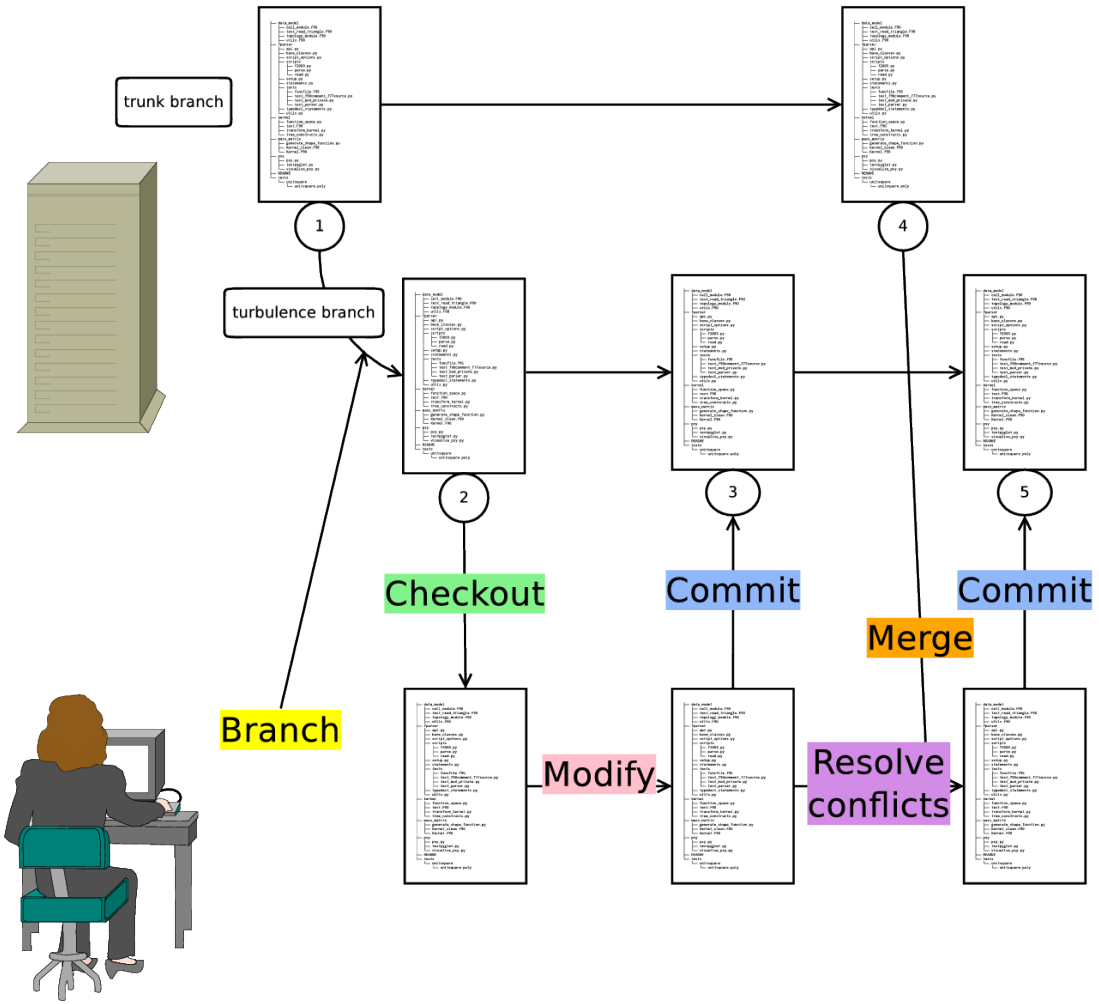
Making a feature branch in a centralised system



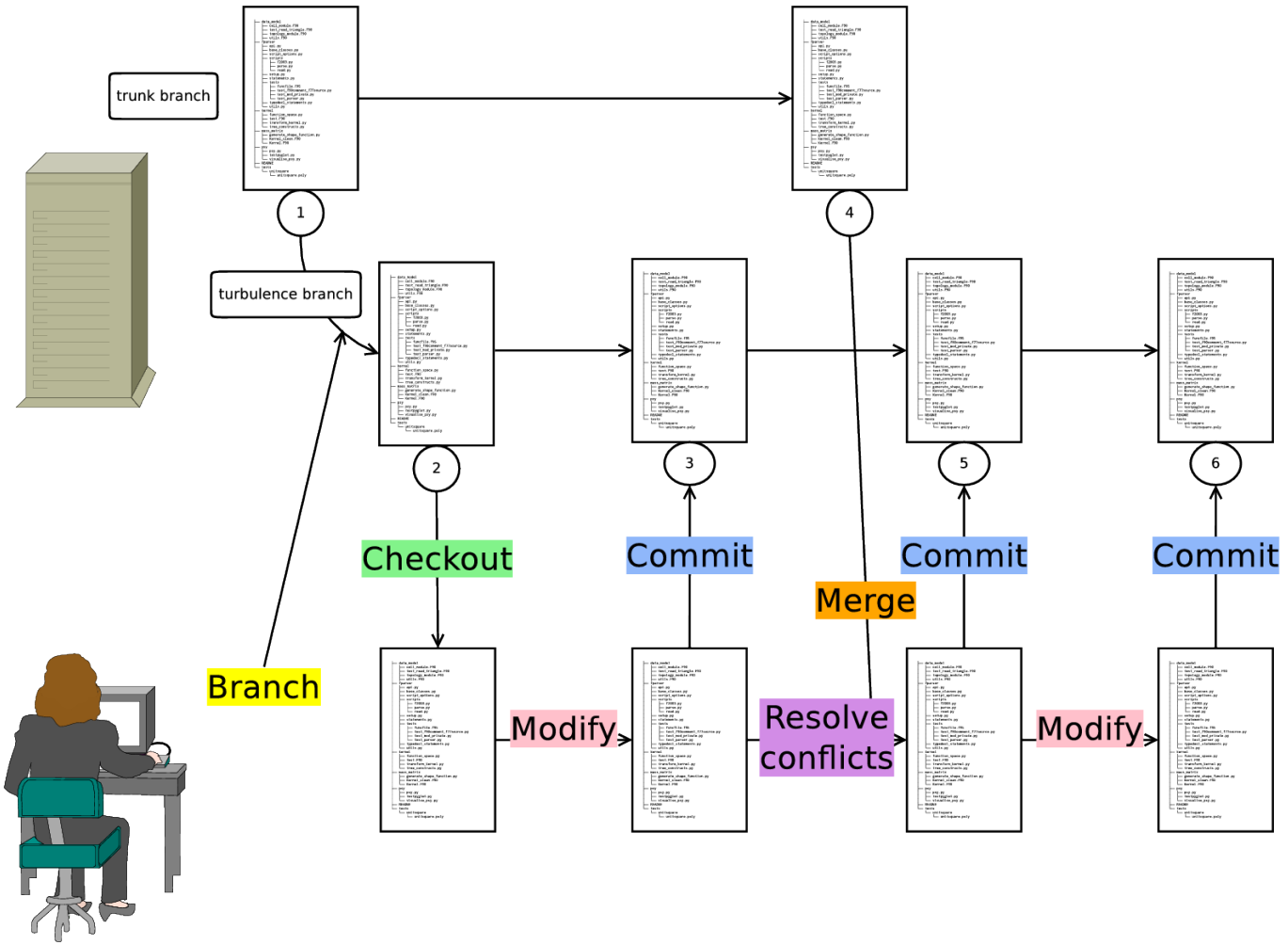
Making a feature branch in a centralised system



Making a feature branch in a centralised system



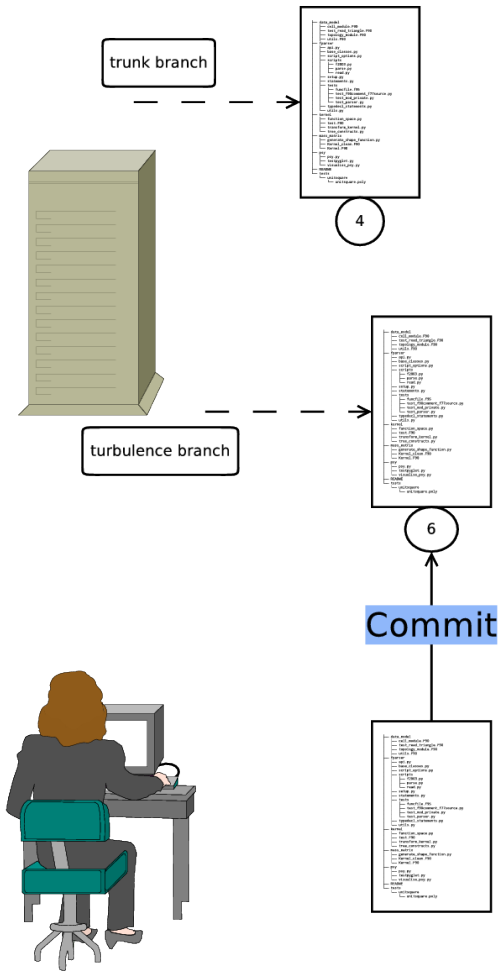
Making a feature branch in a centralised system



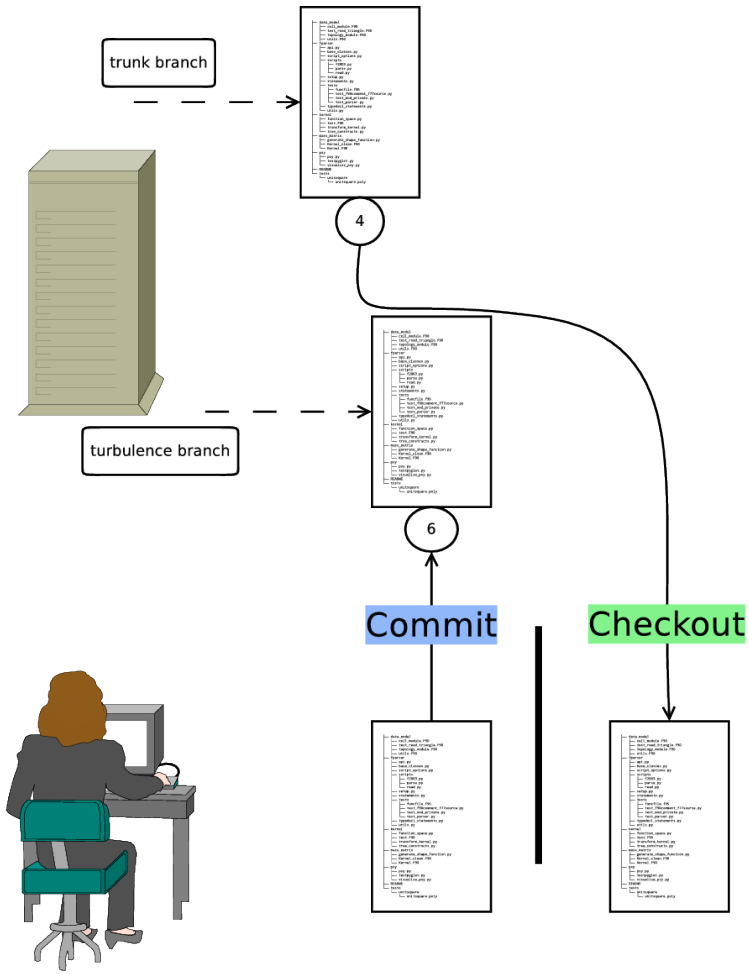
Merging back

So now we know how to work on a feature branch, how do we get our changes back into the trunk?

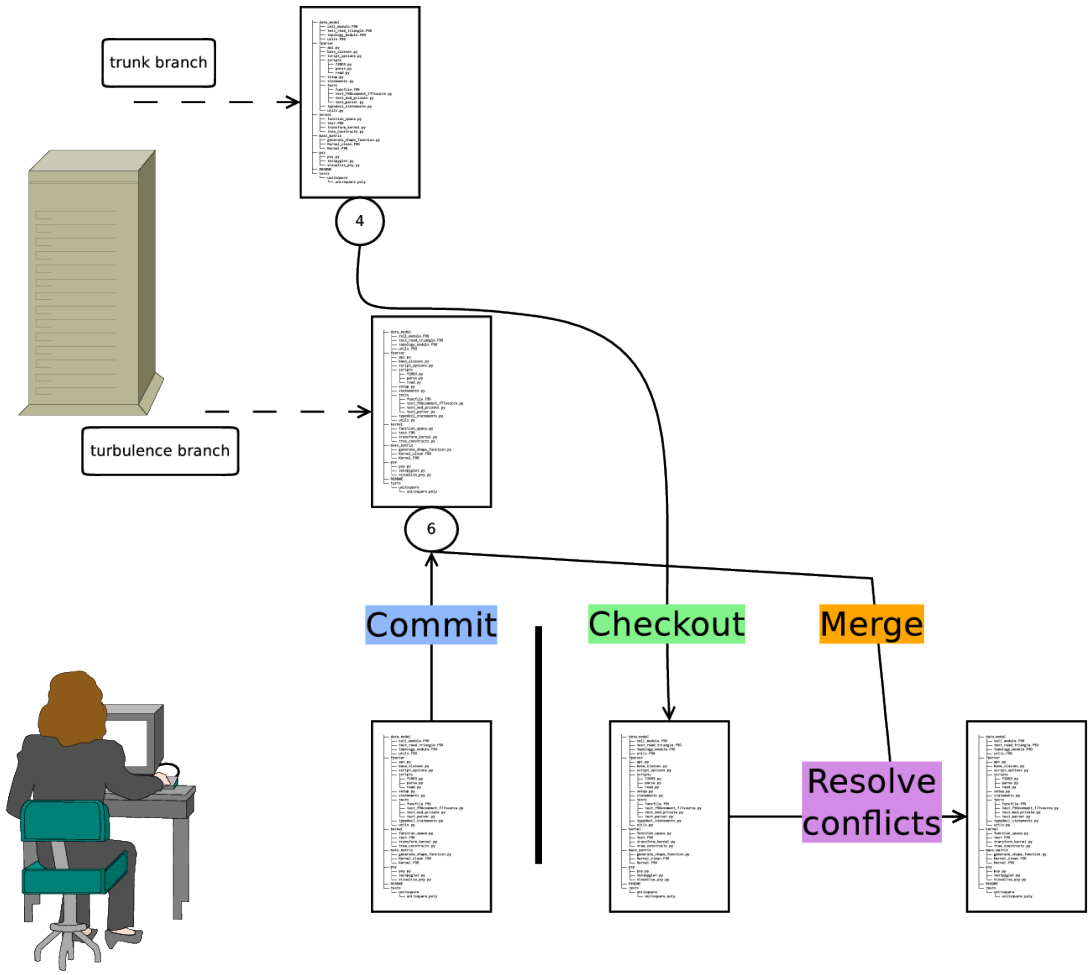
Merging back feature branch in a centralised system



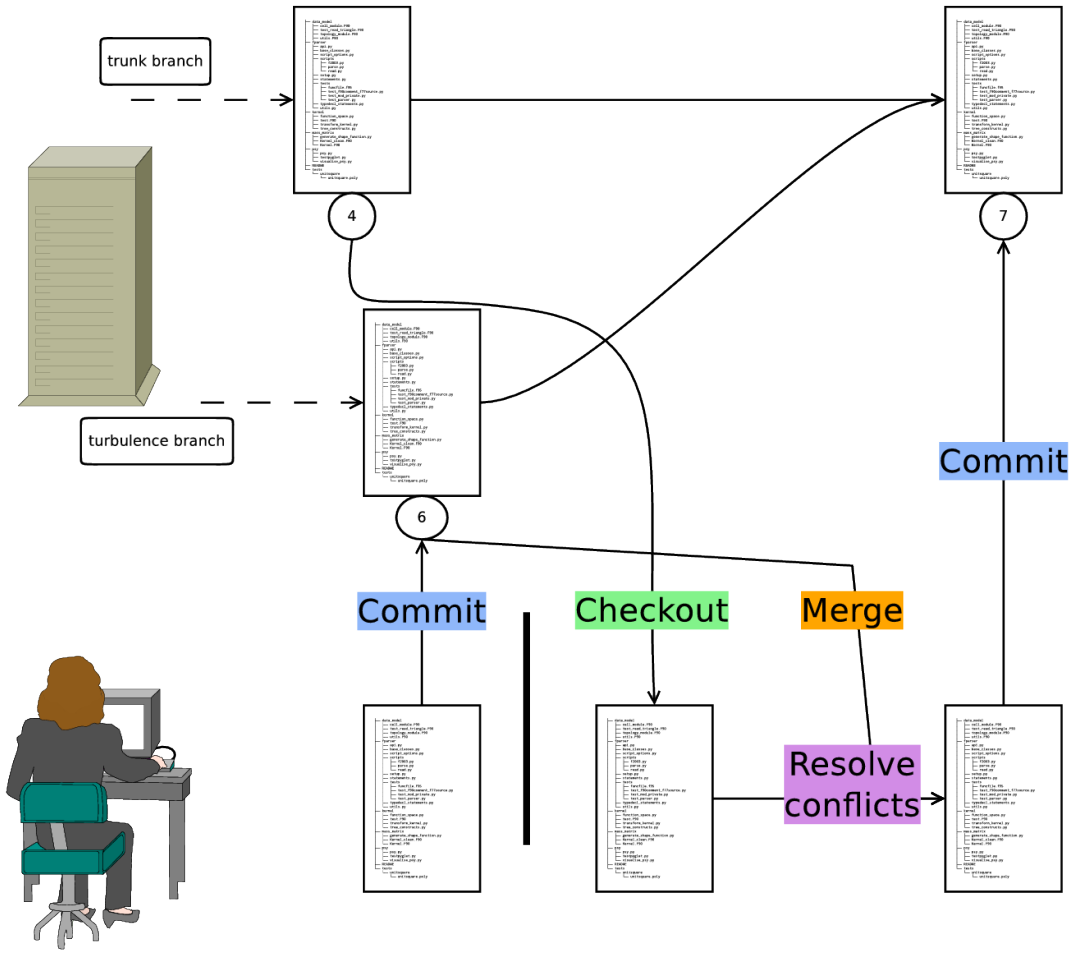
Merging back feature branch in a centralised system



Merging back feature branch in a centralised system



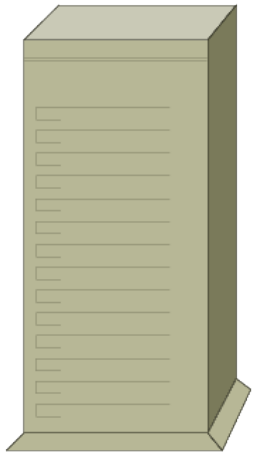
Merging back feature branch in a centralised system

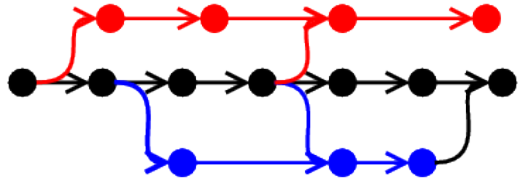


Feature branches are a Good Thing

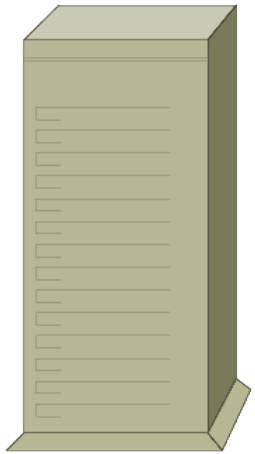
- Keep the trunk in working order.
- Different developers can work independently.
- Encourages frequent commits.
- Facilitates code review (we'll come back to this).

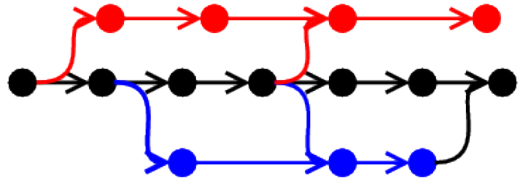
Distributed version control systems



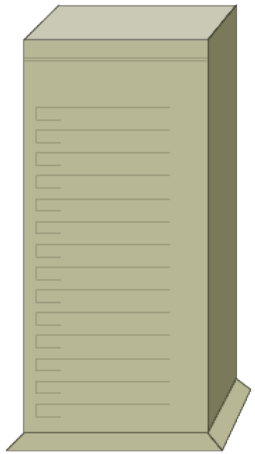


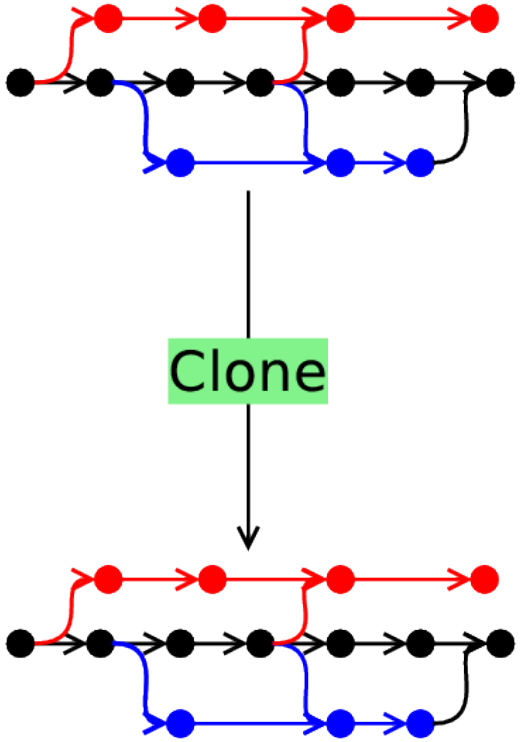
Distributed version control systems



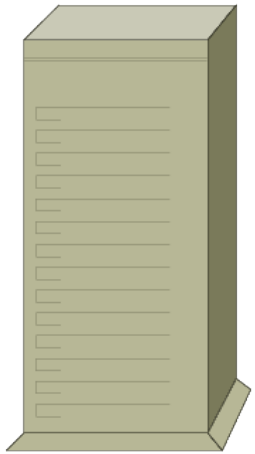


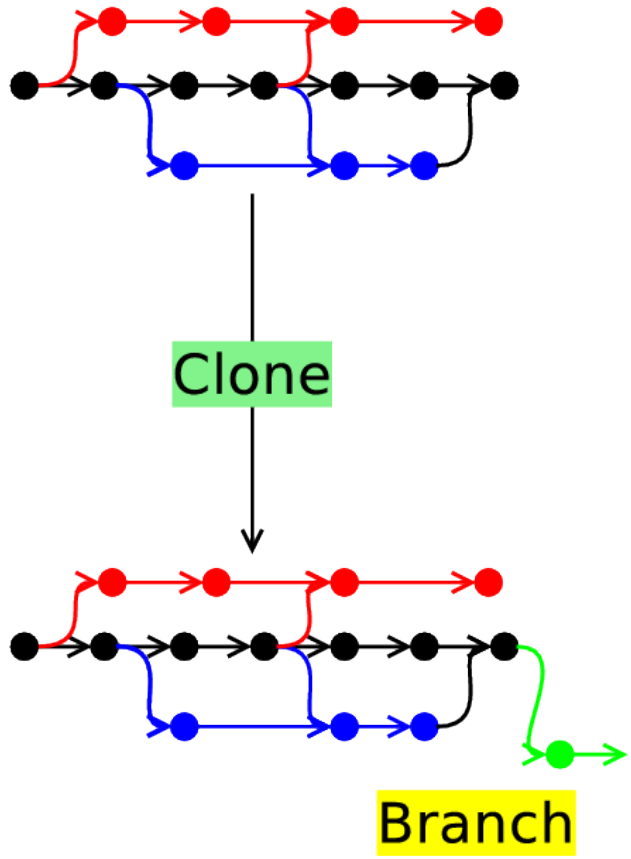
Distributed version control systems



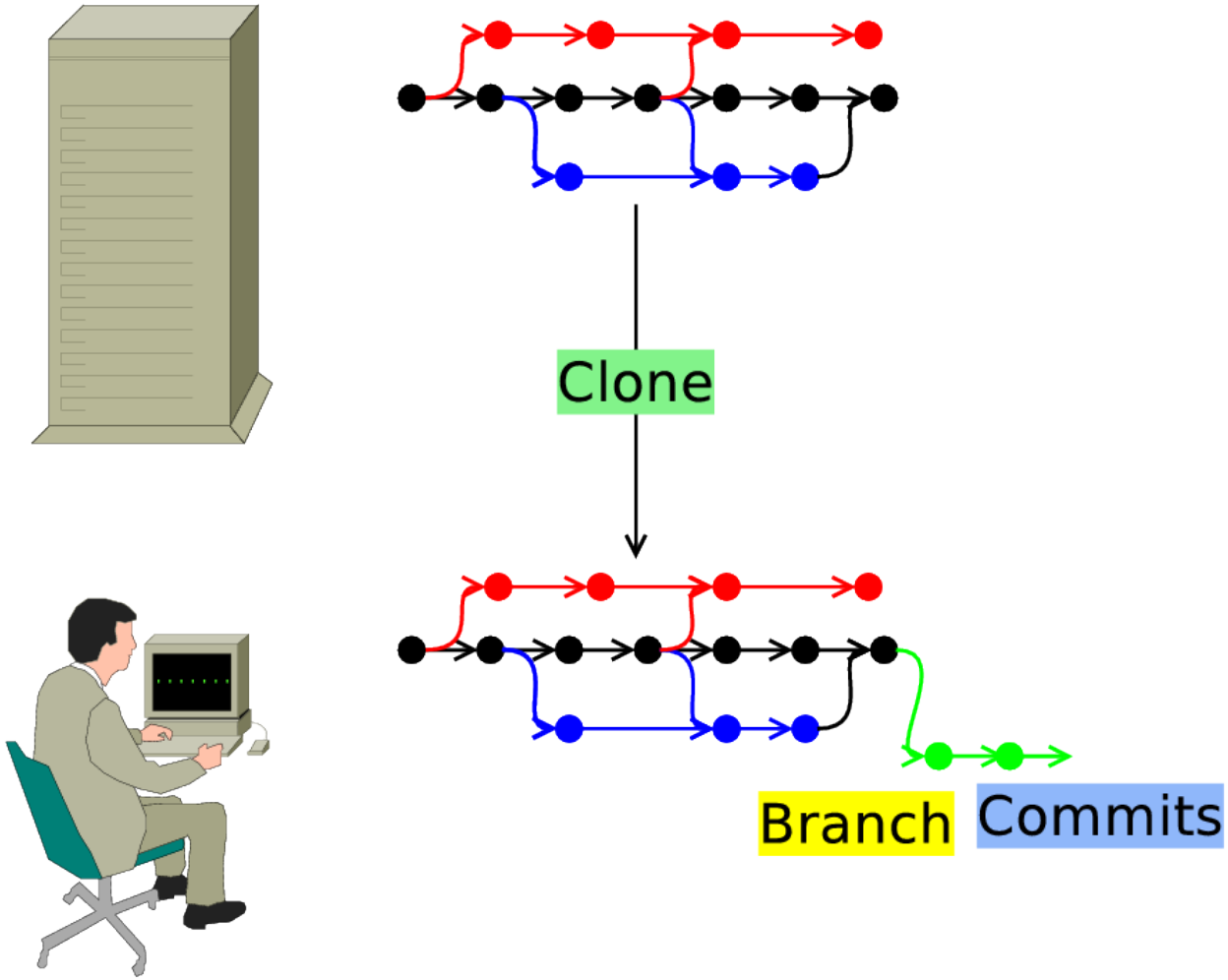


Distributed version control systems

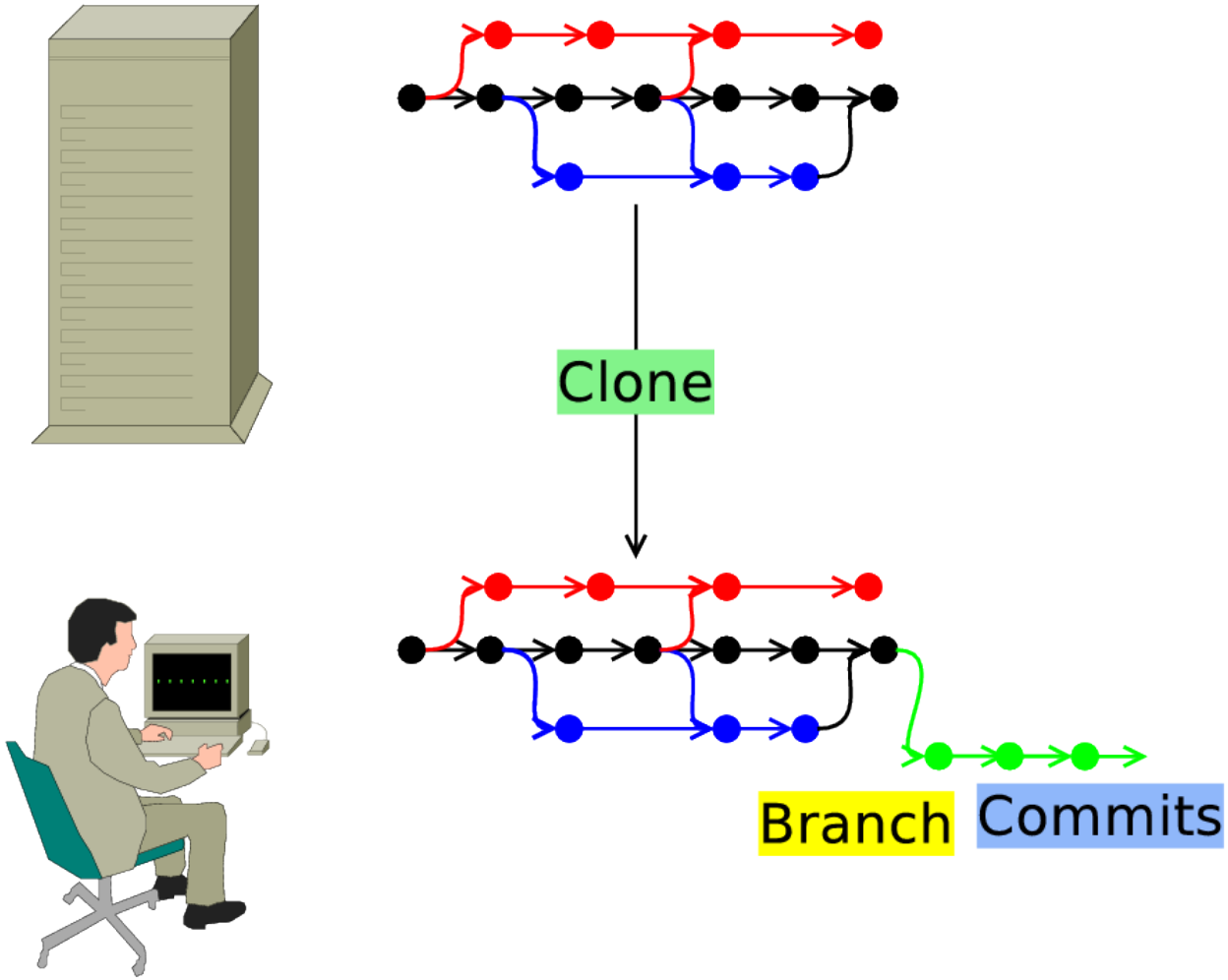




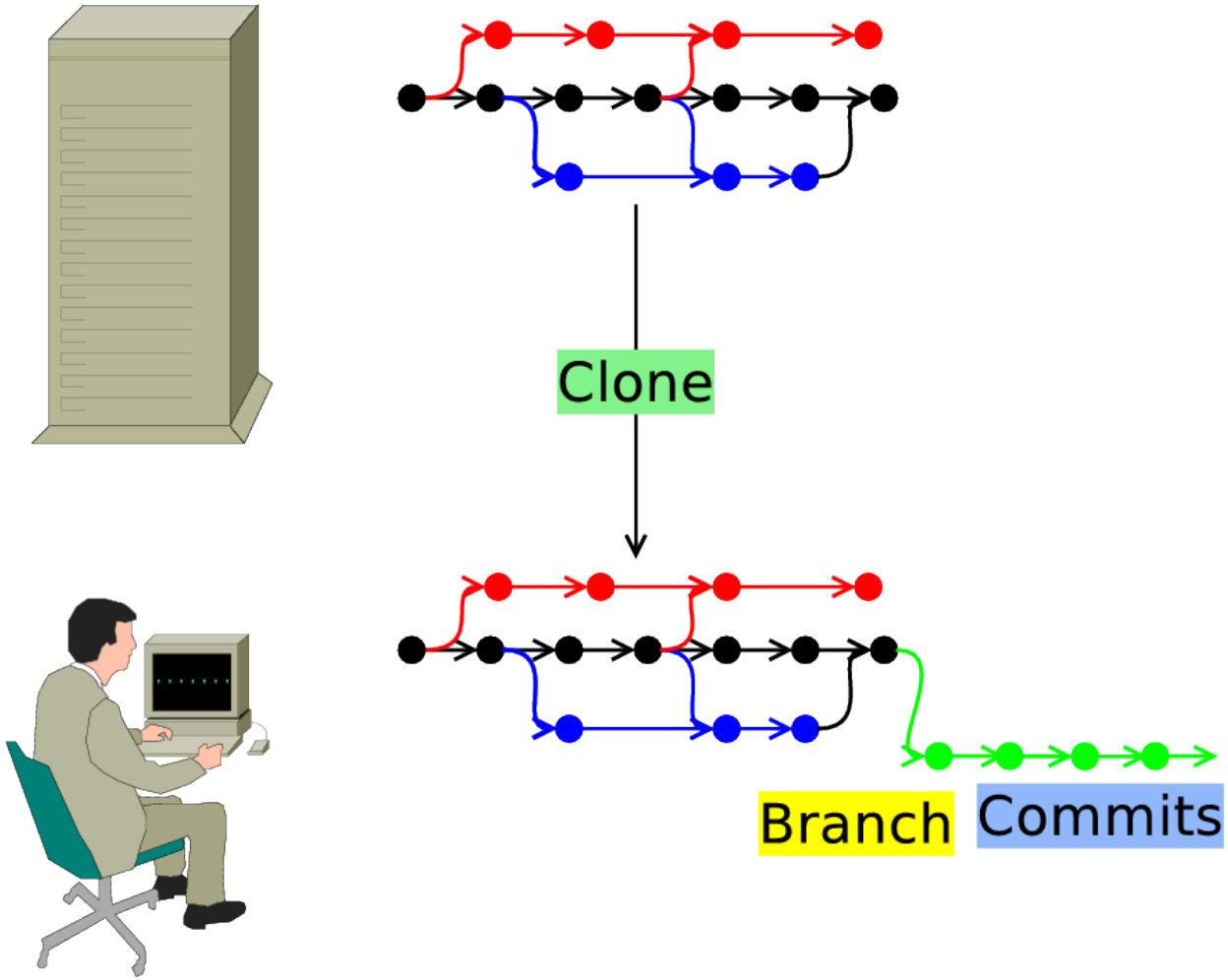
Distributed version control systems



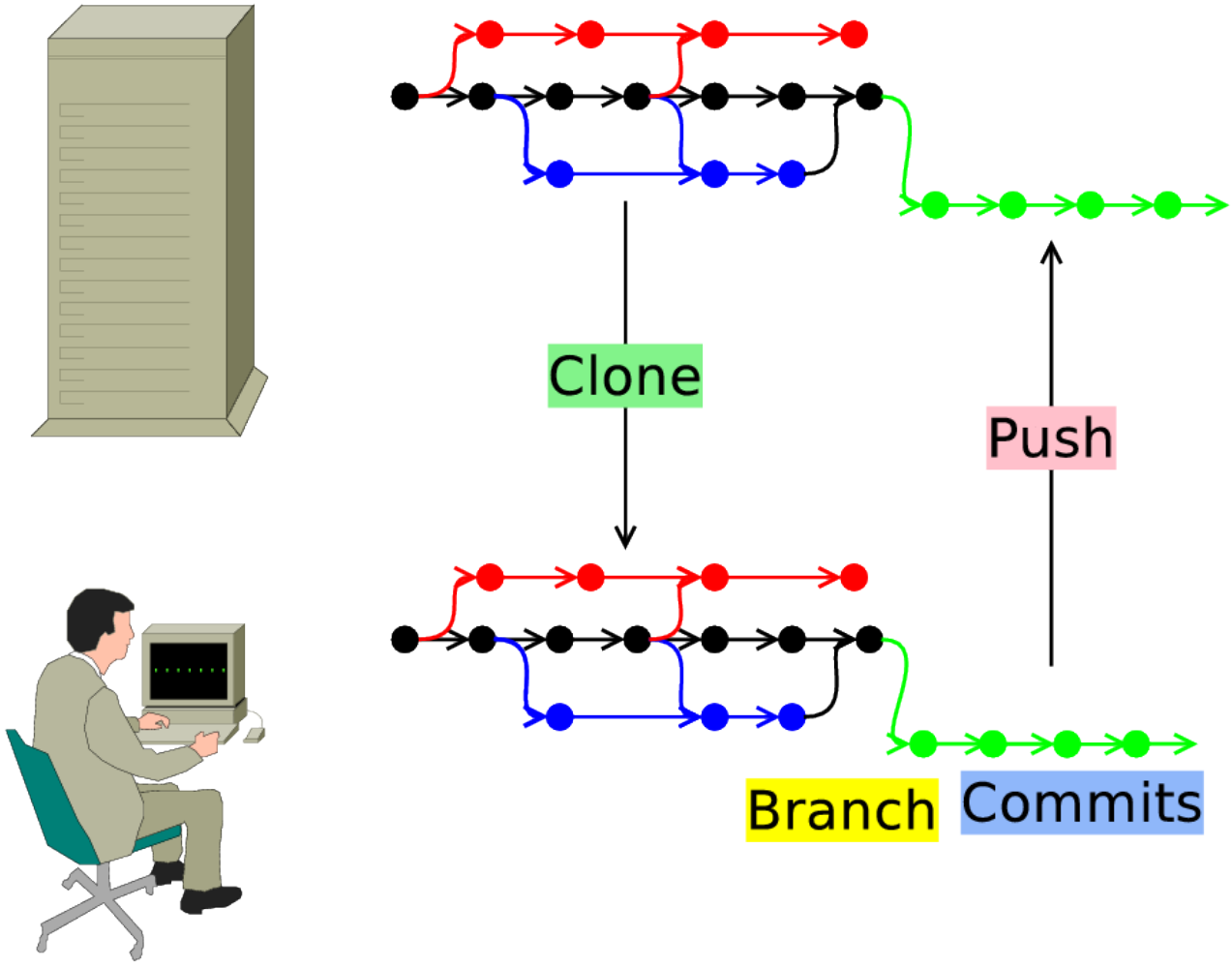
Distributed version control systems



Distributed version control systems



Distributed version control systems



Distributed version control systems

Advantages:

- Most operations are local and so fast. These also work offline.
- More complex workflows involving lots of peers are facilitated.
- Merging is faster and less trouble than with centralised systems.

(Alleged) drawbacks:

- Simple tasks can require more commands than in a centralised system.
- Valuable code is not automatically stored somewhere safe.

Common version control systems

Centralised:

- **CVS** Used to be very common, now rather old-fashioned and hard to work with. Avoid if possible
- **SVN** Very widely used.

Distributed:

- **Git** Probably the most commonly used modern system.
- **Mercurial (hg)** Competitor to git but with syntax more similar to SVN.
- **Bazaar (bzr)** Fairly similar to hg. Extensively used by software related to Ubuntu Linux, but no longer actively developed.

Version control system hosting

Various companies offer the service of hosting version control repositories via web interfaces. These are often free if your project is open source and/or you are affiliated with a university.

The services may also include outher software development tools such as bug trackers, documentation hosting, fora etc.

Github	git
Bitbucket	git and hg
Launchpad	bzr
Sourceforge	git, hg and svn

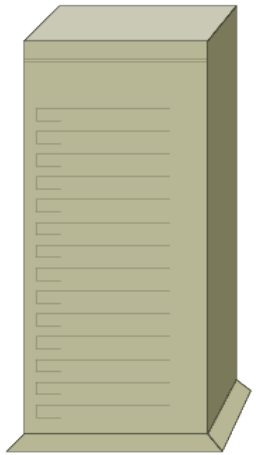
A particularly useful feature of Bitbucket is that it offers unlimited private repositories to university users.

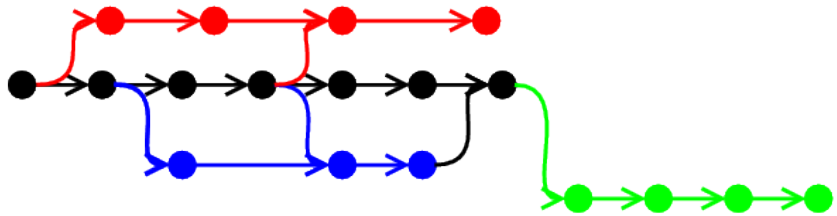
Many departments also offer repository hosting to staff and students.

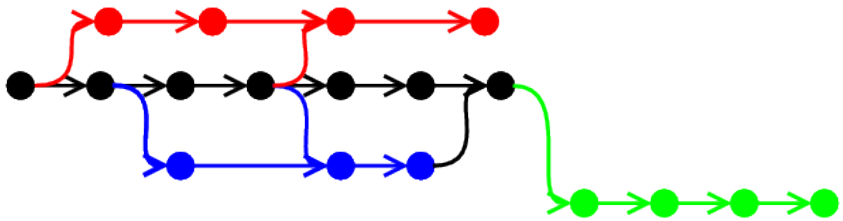
Code review

Having other people read your code is one of the best ways of ensuring quality. Version control hosting sites often support this via a web interface for pull requests or merge requests.

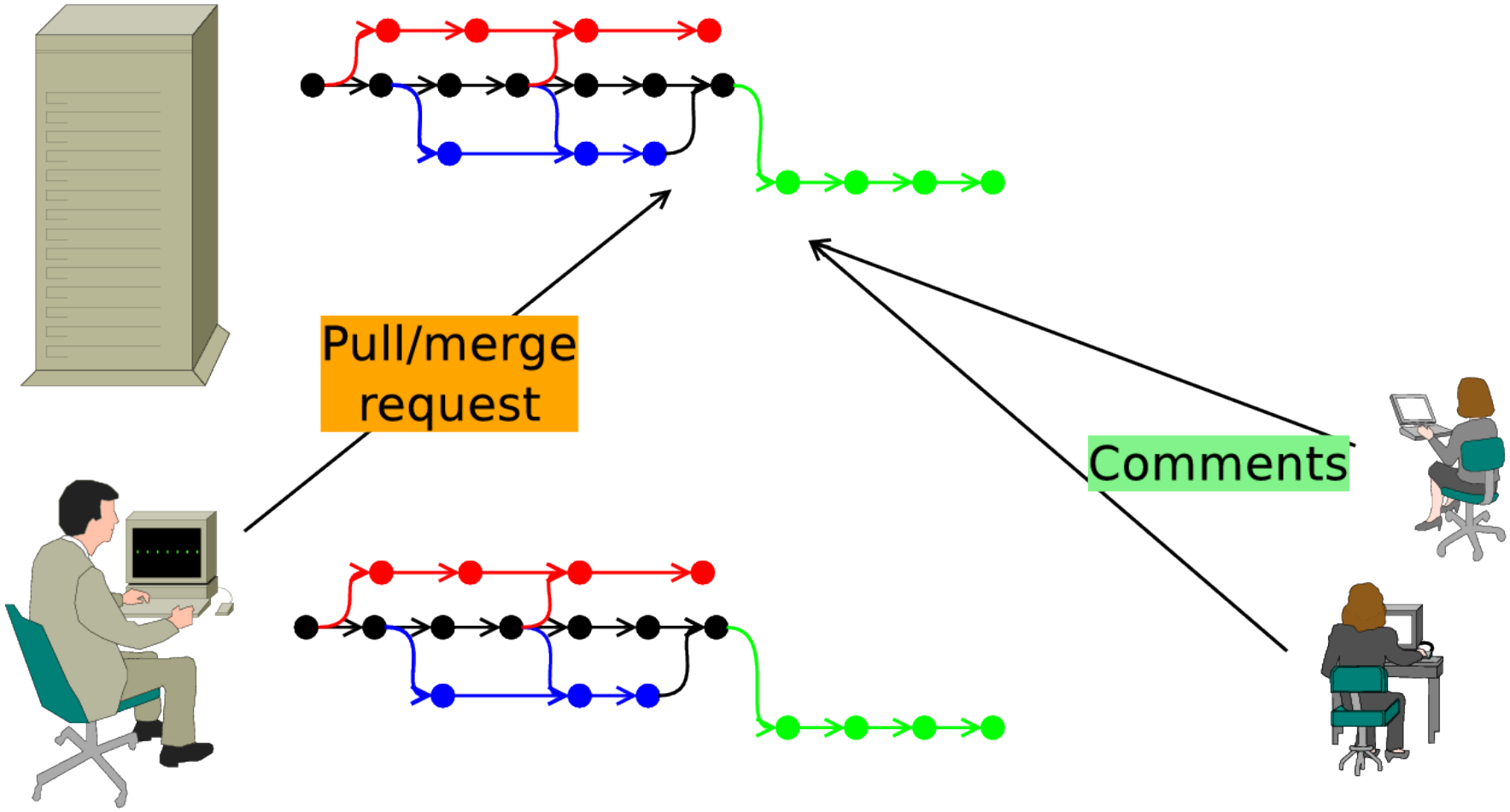
Pull request and code review



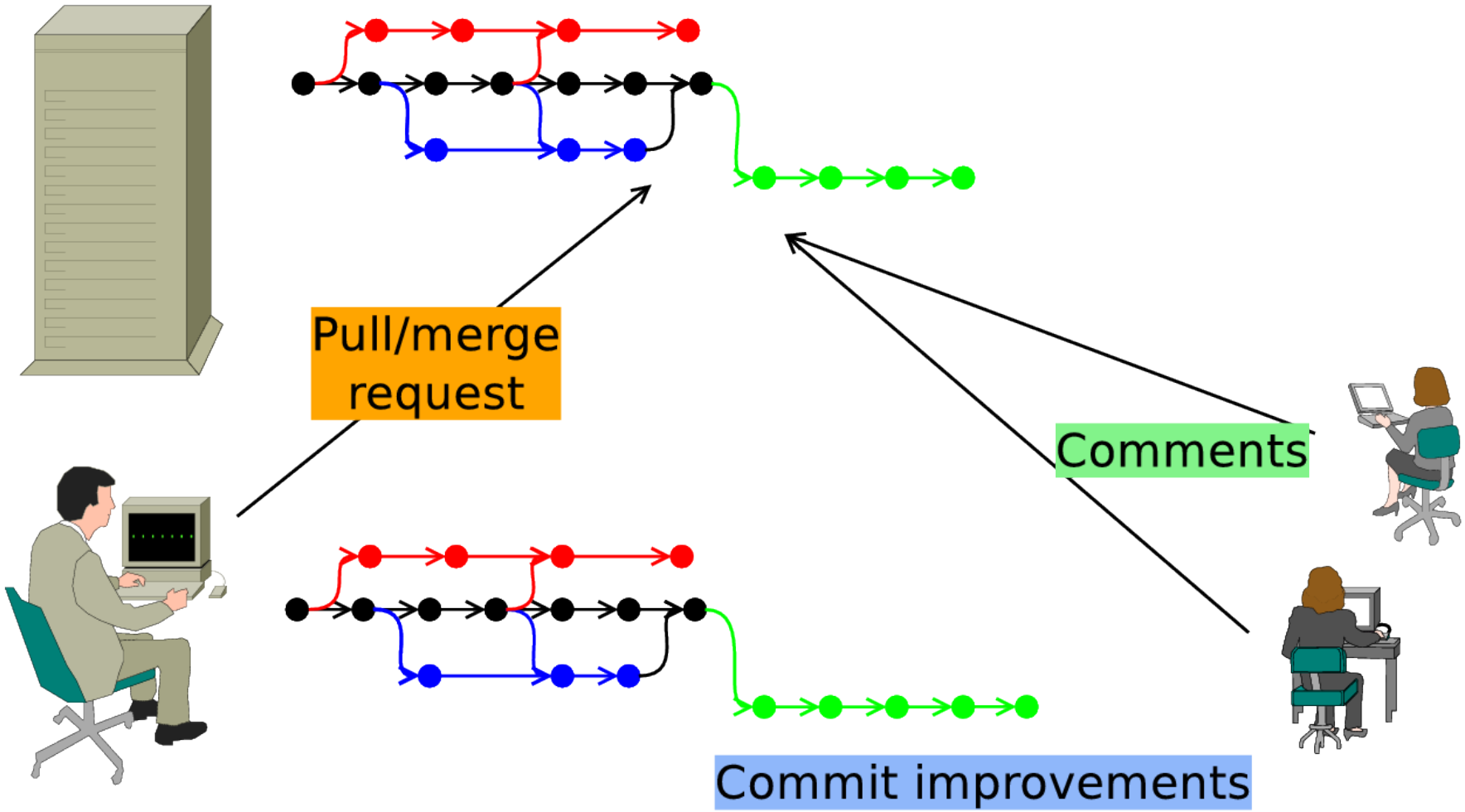




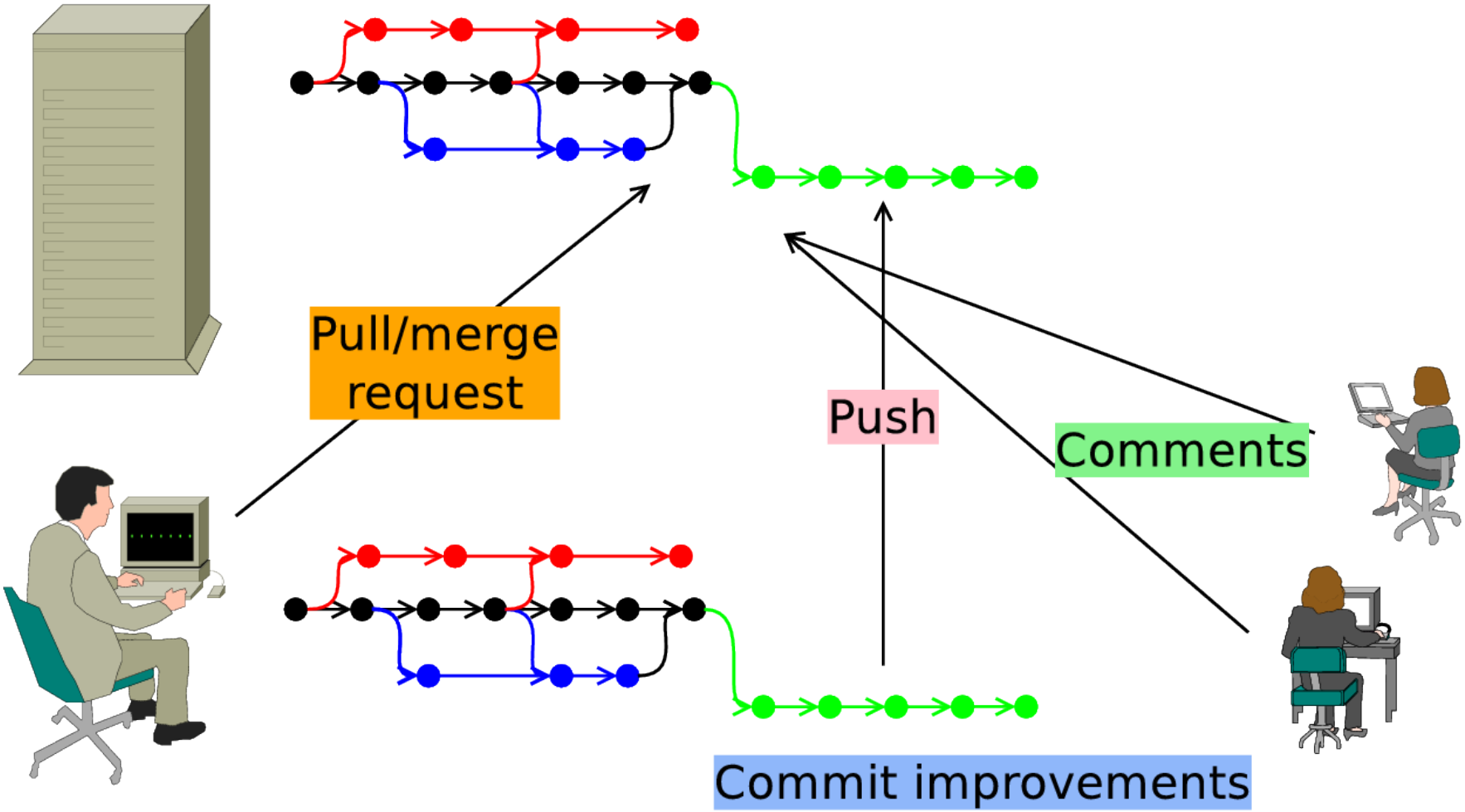
Pull request and code review



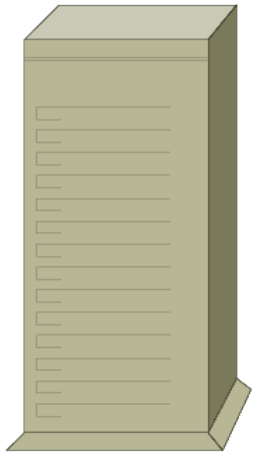
Pull request and code review

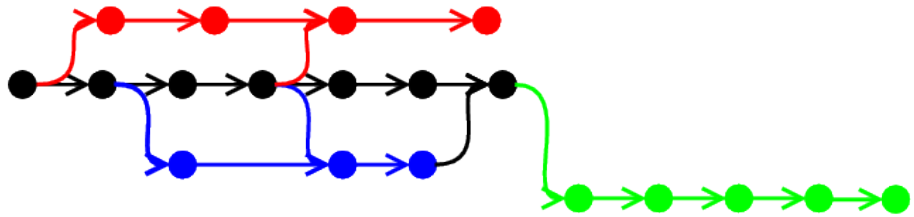


Pull request and code review

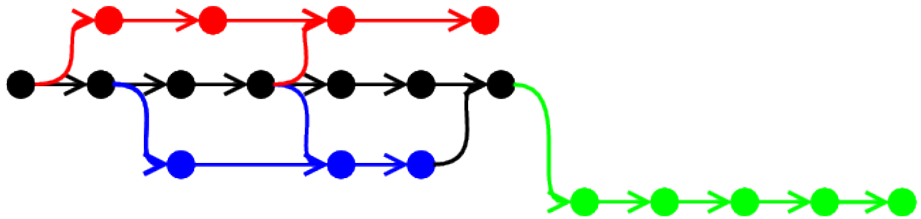


Pull request and code review

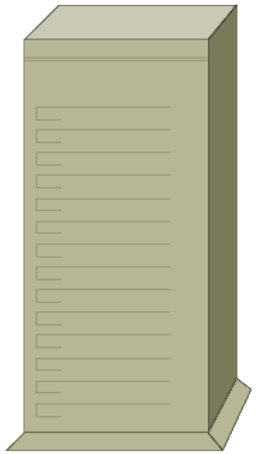


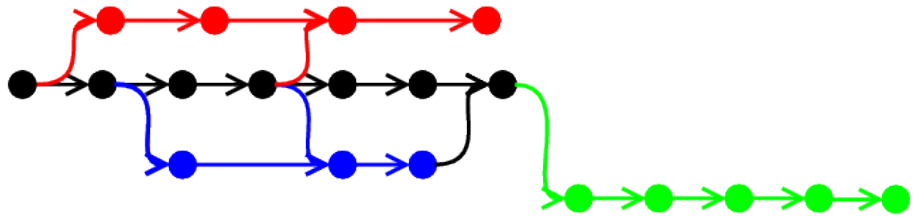


Consensus achieved

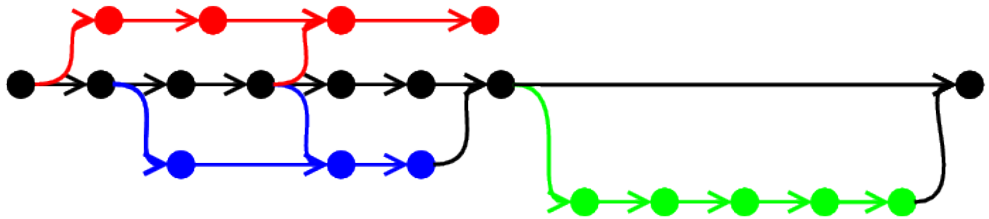


Pull request and code review



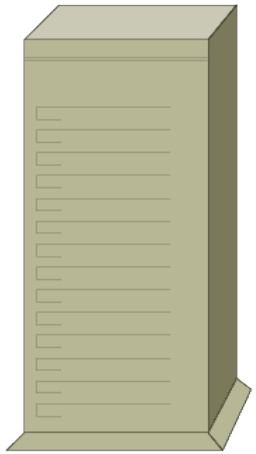


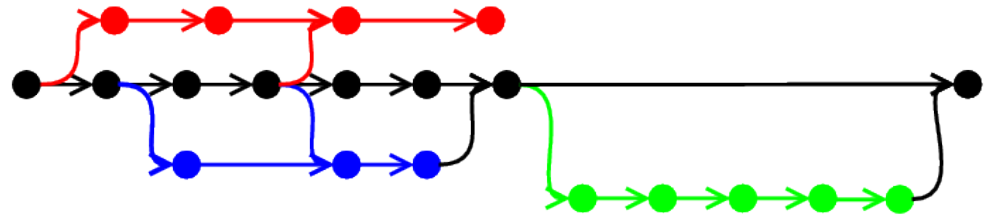
Consensus achieved



Merge

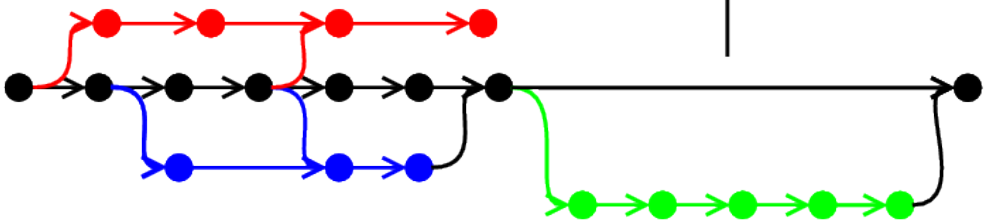
Pull request and code review





Consensus achieved

Push



Merge

Build configuration

How do you make your software work on different machines, with different compilers, different library locations...

In the beginning there was make

We can write a Makefile containing rules for compiling our code:

```
foo: foo.c foo.h bar.o
	gcc -o foo foo.c foo.o

bar.o: bar.c bar.h
	gcc -c bar.c
```

Now typing:

```
>> make foo
```

will cause exactly those complations which are needed in order to build foo.

In the beginning there was make

With make you can:

- specify that certain files are built from other files.
- let make figure out which files have changed and what order to build in.
- make basic choices about things like which compiler to use by setting environment variables.

but make cannot:

- automatically find the right libraries for you.
- specify different compilation configurations for different environments.
- easily turn on and off additional features.
- help make configuration compiler-independent.

Configuration generation tools

- Typically use a scripting language to write a program which knows how to build a project.
- Use a variety of scripting languages.
- Typically provide large libraries of functionality for common tasks (e.g. finding Python, or common libraries).
- May build the project directly (e.g. SCons), or write Makefiles which build the project (e.g. automake/autoconf CMake).
- Typically honour conventional environment variables (e.g. CC, CFLAGS, PETSC_DIR) and so "play nice" alongside system adminstration tools like environment modules.