Version Control

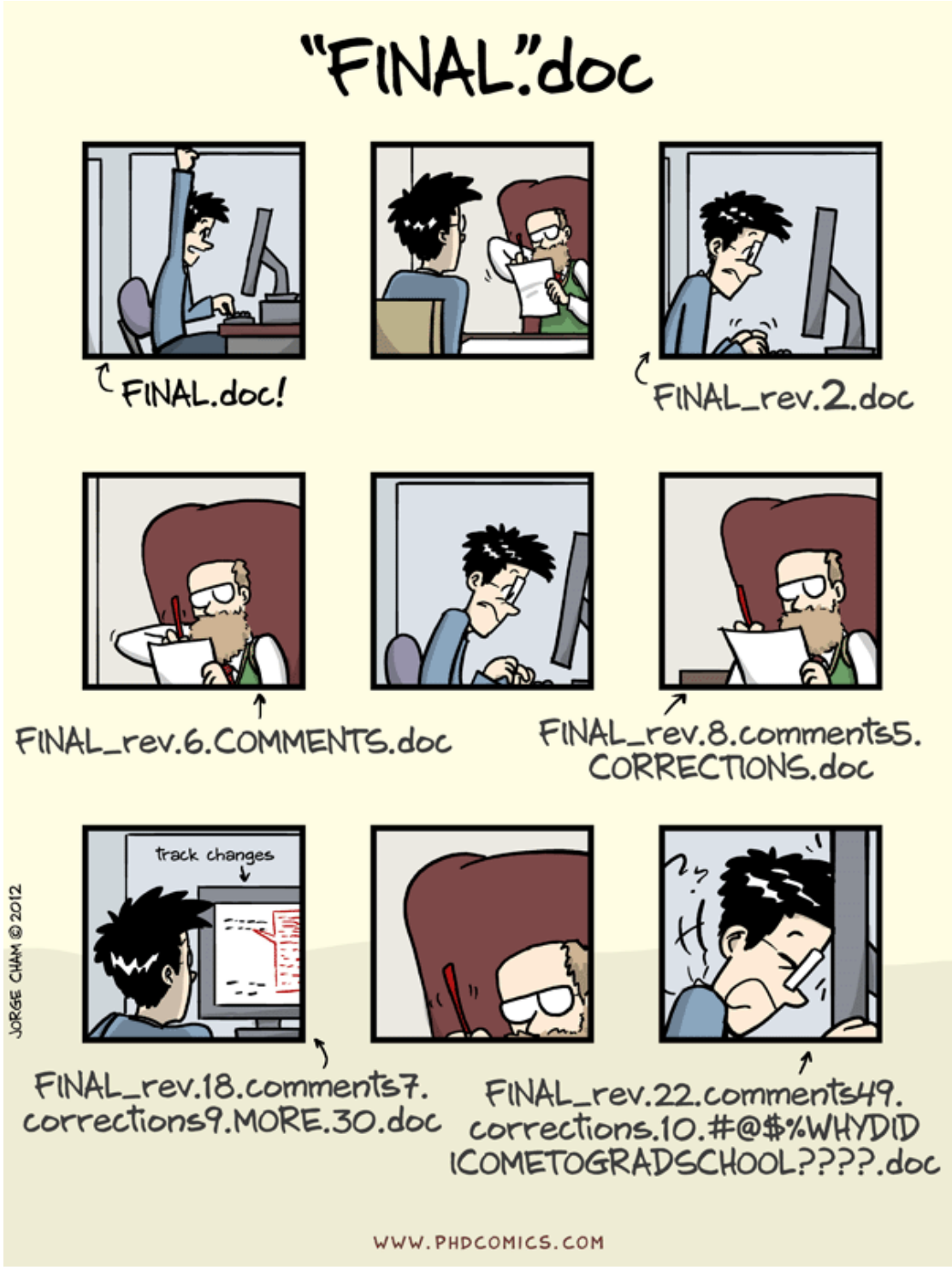
EOAS Software Carpentry Workshop

September 23rd, 2015

Automated Version Control

Learning Goals

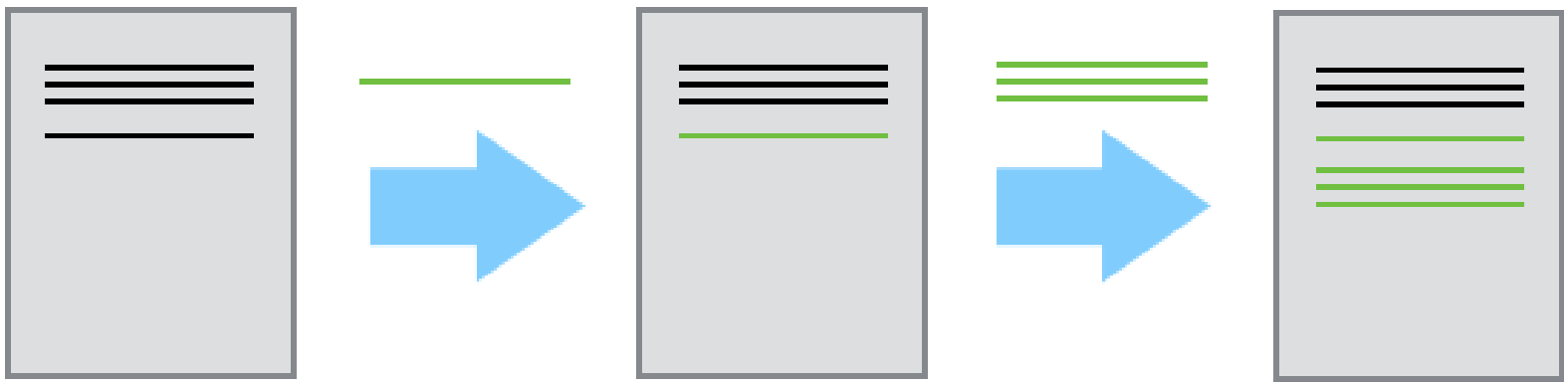
1. Understand the benefits of an automated version control system.
2. Understand the basics of how Mercurial works



"Piled Higher and Deeper" by Jorge Cham, http://www.phdcomics.com

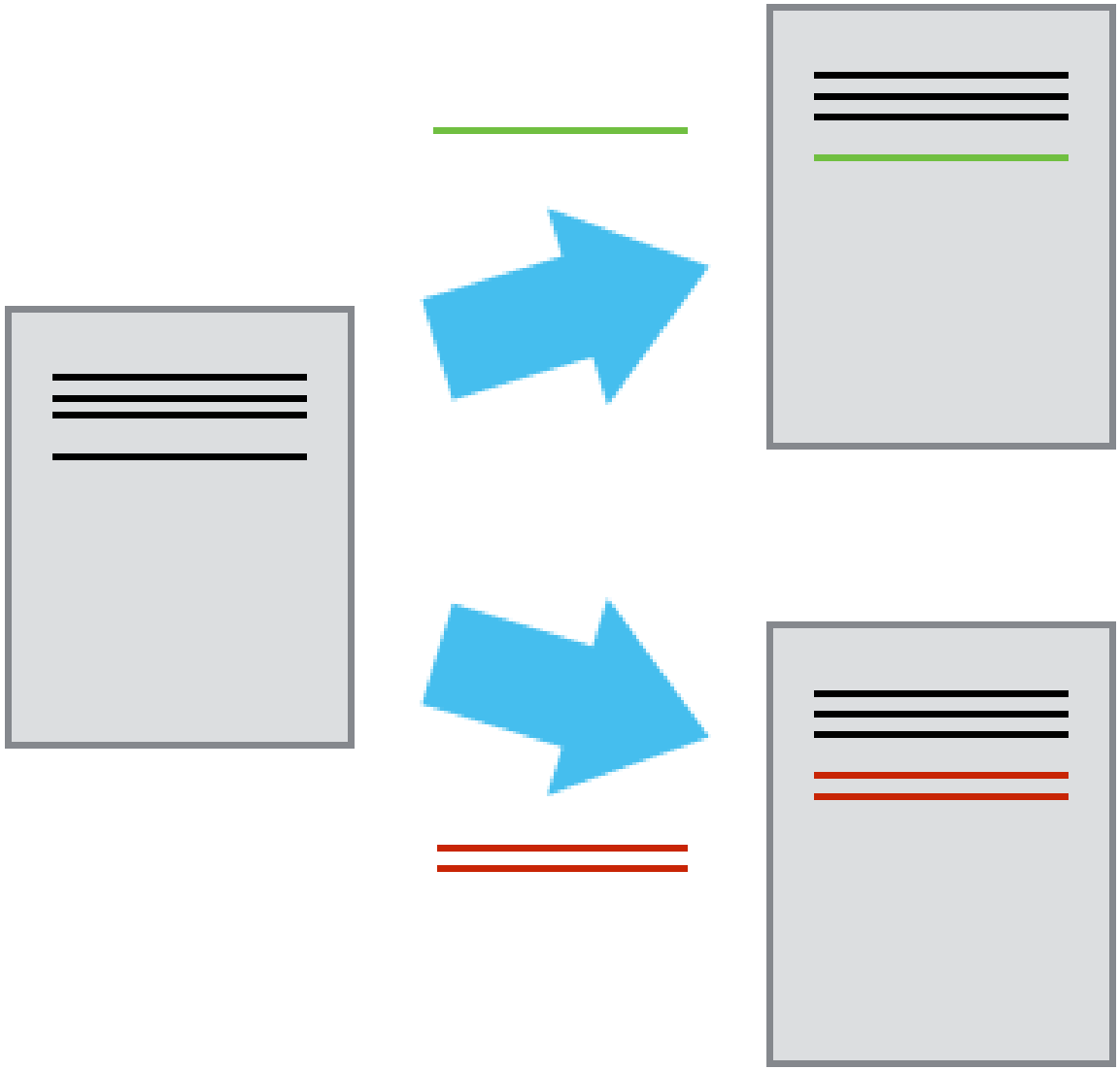
Automated Version Control

Changes are saved sequentially



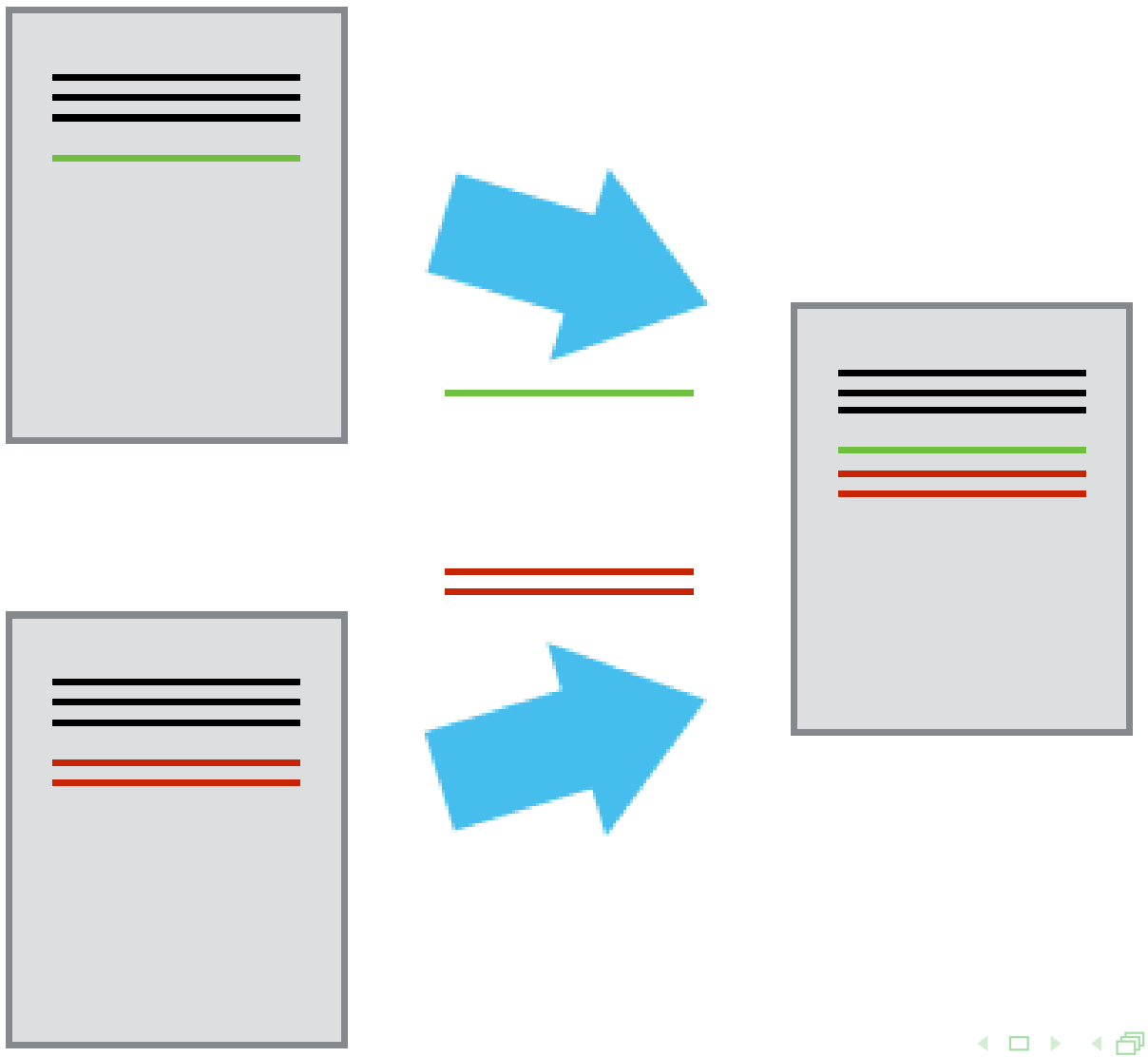
Automated Version Control

Different versions can be saved



Automated Version Control

Multiple versions can be merged



Configuring Mercurial

```
$ EDITOR=nano hg config --edit

[ui]
username = Vlad Dracula <vlad@tran.sylvan.ia>
editor = nano

[extensions]
color =

[color]
mode = win32
```

Creating a Repository

Learning Goal

1. Explain how to initialize a new Mercurial repository.

Lesson Commands

- mkdir forecast
- cd forecast
- hg init
- ls -a
- hg verify

Tracking Files

Learning Goals

1. Display the version control status of files in a repository and explain what those statuses mean.
2. Add files to Mercurial's collection of tracked files.
3. Record metadata about changes to a file.
4. Display the history of changes to files in a repository and explain the metadata that is recorded with each changeset.

Lesson Commands

- nano plan.txt
- hg status
- hg add plan.txt
- hg commit -m "Starting to plan the daily NEMO forecast system."
- hg log

Making Changes

Learning Goals

1. Display the uncommitted changes that have been made to tracked files.
2. Go through the modify-commit cycle for single and multiple files.

Lesson Commands

- nano plan.txt
- hg status
- hg diff
- hg commit -m "Note about atmospheric forcing."
- nano biblio.txt
- hg add biblio.txt
- hg commit -m "Added citation" biblio.txt

Exercise

Create a new Mercurial repository on your computer called bio. Write a three-line biography for yourself in a file called me.txt, commit your changes, then modify one line and add a fourth and display the differences between its updated state and its original state.

Exploring History

Learning Goals

1. Compare files with older versions of themselves.
2. Display the changes that were made to files in a previous changeset.

Lesson Commands

- hg diff --rev 1:2 plan.txt
- hg diff -r 0:2 plan.txt
- hg diff --change 1

Recovering Old Versions

Learning Goals

1. Restore older versions of files.
2. Use configuration aliases to create custom Mercurial commands.

Lesson Commands

- nano plan.txt
- hg revert plan.txt
- hg revert --rev 0 plan.txt
- hg status

Ignoring Things

Learning Goal

1. Configure Mercurial to ignore specific files and explain why it is sometimes useful to do so.

Lesson Commands

- mkdir inprogress
- touch plan.txt inprogress/a.out inprogress/b.out
- hg status
- nano .hgignore
- hg status --ignored

.hgignore

```
syntax: glob
*~
inprogress/
```

Ignoring Things

Learning Goal

1. Configure Mercurial to ignore specific files and explain why it is sometimes useful to do so.

Lesson Commands

- mkdir inprogress
- touch plan.txt inprogress/a.out inprogress/b.out
- hg status
- nano .hgignore
- hg status --ignored

Collaborating

1. Explain what remote repositories are and why they are useful.
2. Explain what happens when a remote repository is cloned.
3. Explain what happens when changes are pushed to or pulled from a remote repository.

- hg paths
- hg push
- hg pull
- hg clone
- hg log --graph
- hg update

We're going to explore collaborating via a remote repository clone on Bitbucket by pretending that we are going back and forth between our home and work computers. We'll simulate that by creating a directory for each location and moving our `planets/` repository into the work computer directory.

```
$ cd
$ cd Desktop/swc/
$ mkdir home-pc work-pc
$ mv planets/ work-pc/
```

These could just as easily be directories on our own and our supervisor's computer, or on the computers of a group of collaborators spread around the world.

Collaborating

1. Explain what remote repositories are and why they are useful.
2. Explain what happens when a remote repository is cloned.
3. Explain what happens when changes are pushed to or pulled from a remote repository.

- hg paths
- hg push
- hg pull
- hg clone
- hg log --graph
- hg update

Conflicts and Merging

1. Explain what conflicts are and when they can occur.
2. Resolve conflicts resulting from a merge.

- hg heads
- hg log -G
- hg merge --tool=kdiff3
- hg summary

Open Science

1. Explain how the GNU Public License (GPL) differs from most other open licenses.
2. Explain the four kinds of restrictions that can be combined in a Creative Commons license.
3. Correctly add licensing and citation information to a project repository.
4. Outline options for hosting code and data and the pros and cons of each.