

Coding Best Practices

Zach Griffith and James Bourbeau

UW-Madison

IceCube Bootcamp 2018

Madison, WI

Virtual environments

Installing Python packages

- pip is a package installer maintained the Python Packaging Authority

Installs numpy from
<https://pypi.org/>

- Easy to use pip (e.g. `pip install numpy`)
- Installs packages into a `site-packages/` directory by default

```
$ pip --version
pip 9.0.1 from /cvmfs/icecube.opensciencegrid.org/
py2-v3/RHEL_7_x86_64/lib/python2.7/site-packages
(python 2.7)
```

Installing Python packages cont.

- pip knows about package version numbers
 - ▶ `pip install numpy`
 - ▶ `pip install numpy==1.13.0`
 - ▶ `pip install numpy>=1.11.1`
- pip works with version control systems
 - ▶ `pip install git+https://github.com/numpy/numpy.git`
 - ▶ Also works with SVN, Mercurial, and Bazaar

Hmmm...what now?

```
$ ssh cobalt
$ pip install --upgrade numpy
. . .
OSError: [Errno 30] Read-only file system: '/cvmfs/
icecube.opensciencegrid.org/py2-v3/RHEL_7_x86_64/
bin/f2py'
```

- IceCube researchers all share the cobalt machines
- IT team install packages for global use
- Virtual environments to the rescue!

What is a virtual environment?

- An independent Python environment with its own `pip` and `site-packages/`.
- Each virtual environment is isolated from other virtual environments, i.e. there are no shared packages.

What is a virtual environment?

Computer

System wide Python

- pip
- site-packages

Project A Python

- pip
- site-packages

Project B Python

- pip
- site-packages

Virtual environment use cases

- You don't have permissions to install in the global `site-packages/` directory
- You have projects with different dependencies
 - ▶ Project A needs `numpy==0.11.1`
 - ▶ Project B needs `numpy==0.14.0`
- You want an easy way to create a reproducible computing environment

Creating virtual environments

- Python 3.3+ has the standard library package `venv`
- For Python 2.6+, you need the open-source package `virtualenv` (also maintained by PyPA)
 - ▶ Already installed in CVMFS `py2-v3`
- Functionally similar, with some minor differences

Working with virtual environments

- Python 2.6+ version

```
$ virtualenv ~/.venvs/project-a
New python executable in /home/jbourbeau/.venvs/project-a/bin/python
Installing setuptools, pip, wheel...done.
$ source ~/.venvs/project-a/bin/activate
(project-a) $
```

- Python 3.3+ version

```
$ python3 -m venv ~/.venvs/project-a
$ source ~/.venvs/project-a/bin/activate
(project-a) $
```

Working with virtual environments

- Creates a (basically) empty environment for you to install packages into

```
(project-a) $ pip list
Package      Version
-----
pip          10.0.1
setuptools  39.2.0
wheel       0.31.1
```

Working with virtual environments

- To exit a virtual environment, just type deactivate!

```
(project-a) $ pip --version
pip 10.0.1 from /home/jbourbeau/.venvs/project-
a/lib/python2.7/site-packages/pip (python 2.7)
(project-a)
$ deactivate
$ pip --version
pip 9.0.1 from /cvmfs/
icecube.opensciencegrid.org/py2-v3/
RHEL_7_x86_64/lib/python2.7/site-packages
(python 2.7)
```

Demo

Conclusions

- Virtual environments are isolated Python environments
- They're an incredibly useful tool for when you:
 - ▶ Don't have permissions to install in global site-packages/
 - ▶ Have projects with different dependencies
 - ▶ Want an easy way to reproduce a computing environment
- Use them

Some additional resources

- [virtualenvwrapper](#) — Set of extensions to virtualenv. It will save you keystrokes!
- [conda](#) — Open-source package management system and environment management system.
 - ▶ NOTE: conda environments don't play well with IceCube software.
 - ▶ "[Conda: Myths and Misconceptions](#)" article by Jake VanderPlas
- David Beazley's "Modules and Packages: Live and Let Die!" talk @ PyCon 2015 [[YouTube](#)]

Version Control

What is version control?

- A way to track changes in files over time
- Commit changes incrementally to a file repository

Why use version control?

- Version control backs up your code base!

```
$ nano my_awesome_code.py
```

- Days later:

```
$ rm -rf *
```

- Oh no...
- with version control, it's okay!

Why use version control?

- Track the development of your code base with incremental changes!
 - Don't make your own versions...

```
~/cool_code $ mv * old_code_2a
```

Why use version control?

- **Collaborate easily with colleagues!**
 - ▶ Work on the same code with separate copies
 - ▶ Let version control handle finding conflicts and merging code

Why use version control?

- You'll likely be required to:

internal data repository
basic requirements for reproducibility

before publishing an analysis in a refereed journal...

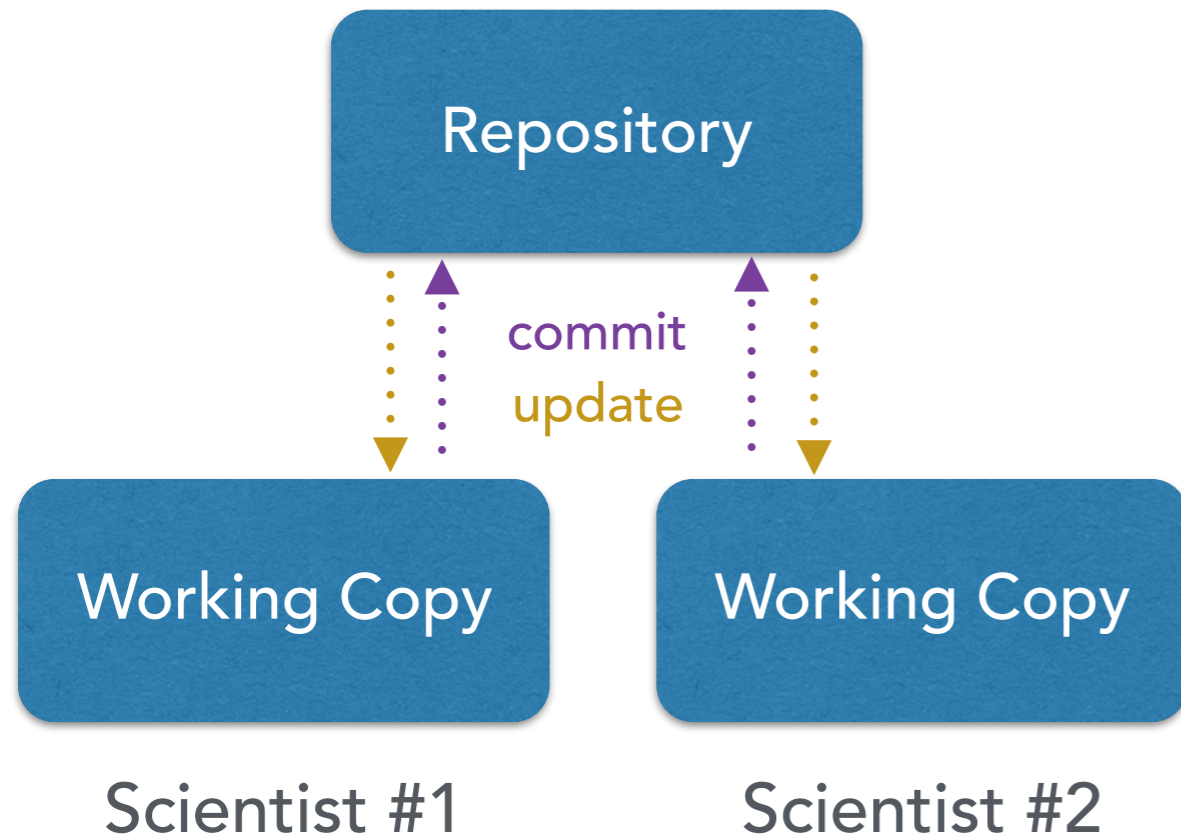
- 1 tag software & scripts in an analysis release package
- 2 store final sample data files with full information on disk
- 3 list all simulation files used in the analysis
- 4 document analysis steps and how to use the software to reproduce results

How to use version control?



- SVN: the version control system of choice for IceCube software projects
- SVN is a centralized version control system

Centralized Version Control















- Code repository is stored on a remote server
- check out the current version to your workspace
- `commit` changes to the repository
- `update` your copy with changes from the repository

Where is the remote server?

- The official IceCube software repository is at:
 - ▶ <http://code.icecube.wisc.edu/projects/icecube>

source: IceCube

View revision: View diff against:

Name ▲	Size	Rev	Age	Author	Last Change
▶  analyses		160503 	8 weeks	zgriffith	updating to avoid naming conflicts, add options to load from original ...
▶  Attic		147624 	21 months	olivas	i think we can safely move this to the attic
▶  ctest		47838 	10 years	blaufuss	The offline-trunk no longer has the needed dart goodies and all dart ...
▶  meta-projects		161758 	19 hours	olivas	"advancing stable combo."
▶  papers		160517 	8 weeks	dxu	documenting the initial version submitted to ApJ on Dec. 18, 2017
▶  projects		161761 	5 hours	jvansanten	PropagateNu?() should not modify the frame directly
▶  sandbox		161763 	2 minutes	atrettin	make confusion matrix compatible again
▶  tools		110391 	5 years	nwhitehorn	Add some missing headers to make this compile against libc++.

- Your work in progress code goes in the "sandbox"

The Sandbox

- The place for putting your analysis projects
- Also where to store projects which may eventually become “official”

source: [IceCube / sandbox](#)

View revision: View diff against:

Name	Size	Rev	Age ▾	Author	Last Change
↑ ../					
▷ yanez		161763	7 minutes	atrettin	make confusion matrix compatible again
▷ jgonzalez		161762	102 minutes	jgonzalez	enable compression
▷ detosi		161757	19 hours	delia.tosi	as used for level3 processing
▷ terliuk		161754	22 hours	andrii.terliuk	Changes
▷ kjm		161751	27 hours	chraab	FRA.generate_report: restore compatibility with single-sim-file.
▷ reimann		161750	28 hours	reimann	updated cleand up and made nice class.
▷ ymakino		161749	31 hours	ymakino	super minor change
▷ Gen2-Scripts		161743	32 hours	ymakino	new segment, muongun_clsims.py
▷ carott		161742	38 hours	carott	Arduino script from Seokmin
▷ gluesenkamp		161735	2 days	gluesenkamp	removed some prints
▷ mhuennefeld		161732	2 days	mhuennefeld	deleted branch_boost
▷ cweaver		161720	4 days	cweaver	Remove redundant (and harmful) link to boost_python
▷ CORSIKA_nu_simulation		161719	4 days	kjero	Done for the day
▷ carguelles		161714	4 days	carguelles	better use case

SVN Demo

Your first SVN repository

- You can make directories on the remote server with:

```
$ svn mkdir <url to new dir on server> -m "commit message"
```

- Conventionally each IceCube member makes their own sandbox directory within which to store projects
- Make your own!

```
$ svn mkdir http://code.icecube.wisc.edu/svn/sandbox/<username>/ -m "Making my personal sandbox"
```

Your first SVN repository

- Make a repository in your personal directory on the remote server:

```
$ svn mkdir http://code.icecube.wisc.edu/svn/sandbox/<username>/  
myfirstsandbox -m "Making my first sandbox entry"  
$ mkdir my_first_project  
$ cd my_first_project  
~/my_first_project $
```

- Now checkout the project from the server with **checkout!**

```
~/my_first_project $ svn co http://code.icecube.wisc.edu/svn/sandbox/  
<username>/my_first_project .
```

Making changes: `svn add` and `svn commit`

- Before committing, we need to tell svn about the files we care about:

```
~/my_first_project $ svn add README
A          README
~/my_first_project $
```

- Now we're ready to commit to the remote repo!

```
~/my_first_project $ svn commit
Adding          README
Transmitting file data .
Committed revision 163434.
~/my_first_project $
```

Commit Messages

- without a -m commit message, a text editing window will open to write the message
- Good practice: write a single line briefly describing the change your commit implements
 - if desired, put a longer description in a separate paragraph

```
add a README file

include a brief description of the project and installation instructions.
TODO: add conda installation support.
--This line, and those below, will be ignored--

A   README
~
```

Viewing Changes

- Print a list of all files not included or changed from the checked-out revision with `svn status`:

```
~/my_first_project $ svn status
?      requirements.txt
M      README
~/my_first_project $
```

- For tracked files with changes, view the difference from the checked-out revision with `svn diff`:

```
~/my_first_project $ svn diff requirements.txt
--- requirements.txt      (revision 163435)
+++ requirements.txt      (working copy)
@@ -1,3 @@
 numpy
+scipy
+matplotlib >= 2.0
```

Updating from the Remote Repo

- Update to the latest revision version from the remote repo with `svn update`:

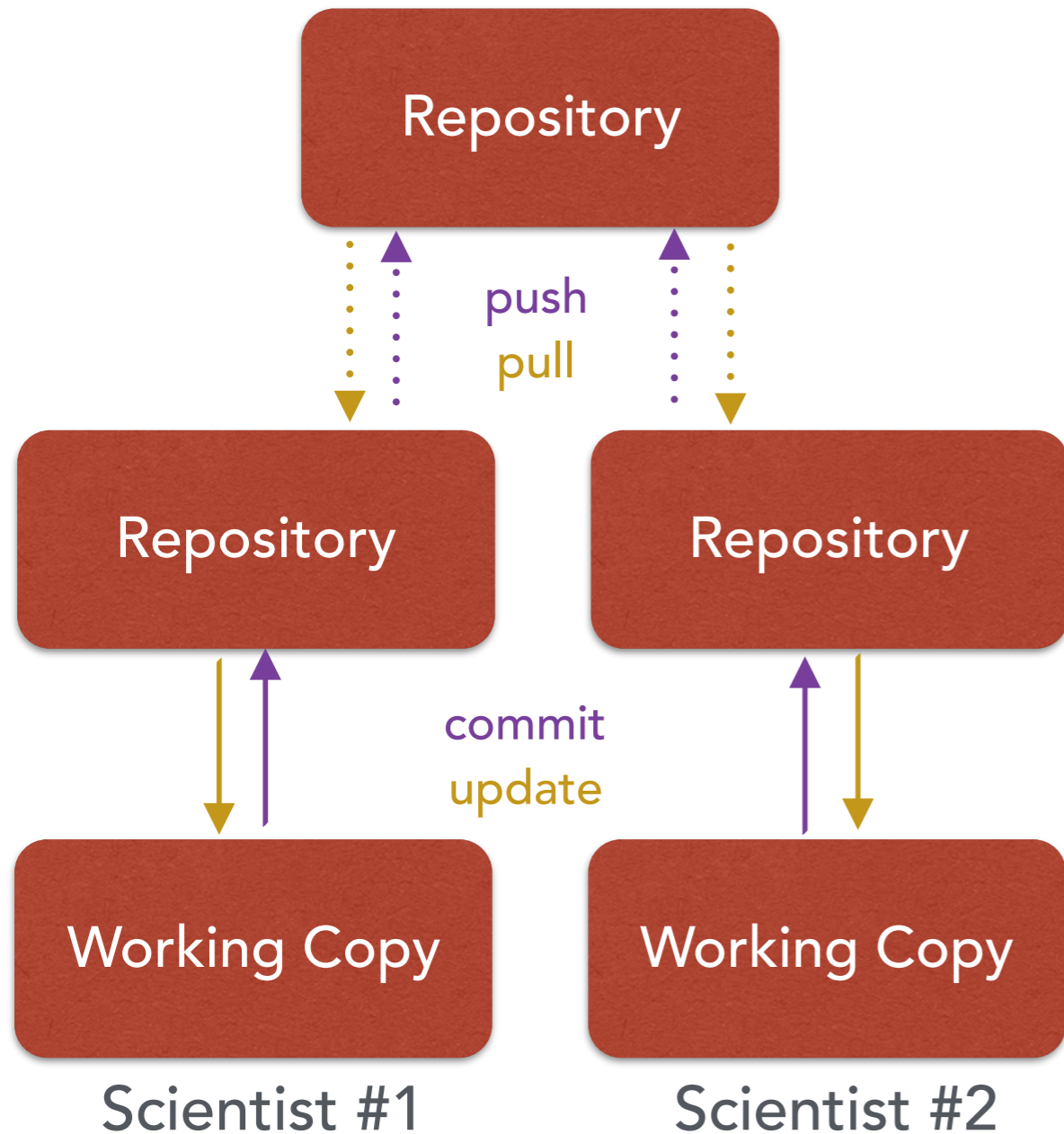
```
~/my_first_project $ svn commit
Updating '.':
U    README
A    requirements.txt
Updated to revision 163435.
```


Split up commits with `svn changelist`

- Sometimes you change a bunch of files but it doesn't make sense to commit them all at once
- You can make changelists of files to commit:

```
~/my_first_project $ svn status
?      hello_world.py
M      requirements.txt
~/my_first_project $ svn changelist requirements requirements.txt
A [requirements] requirements.txt
~/my_first_project $ svn commit --changelist requirements -m "update requirements"
Sending      requirements.txt
Transmitting file data .
Committed revision 163437.
~/my_first_project $ svn status
?      hello_world.py
~/my_first_project $
```

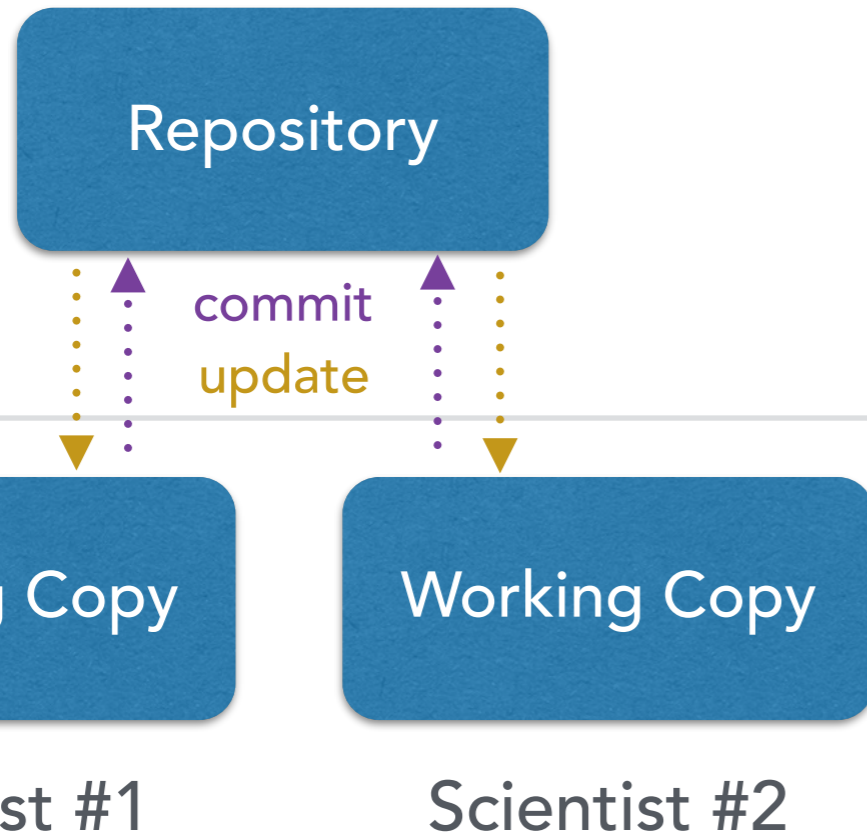
Distributed Version Control



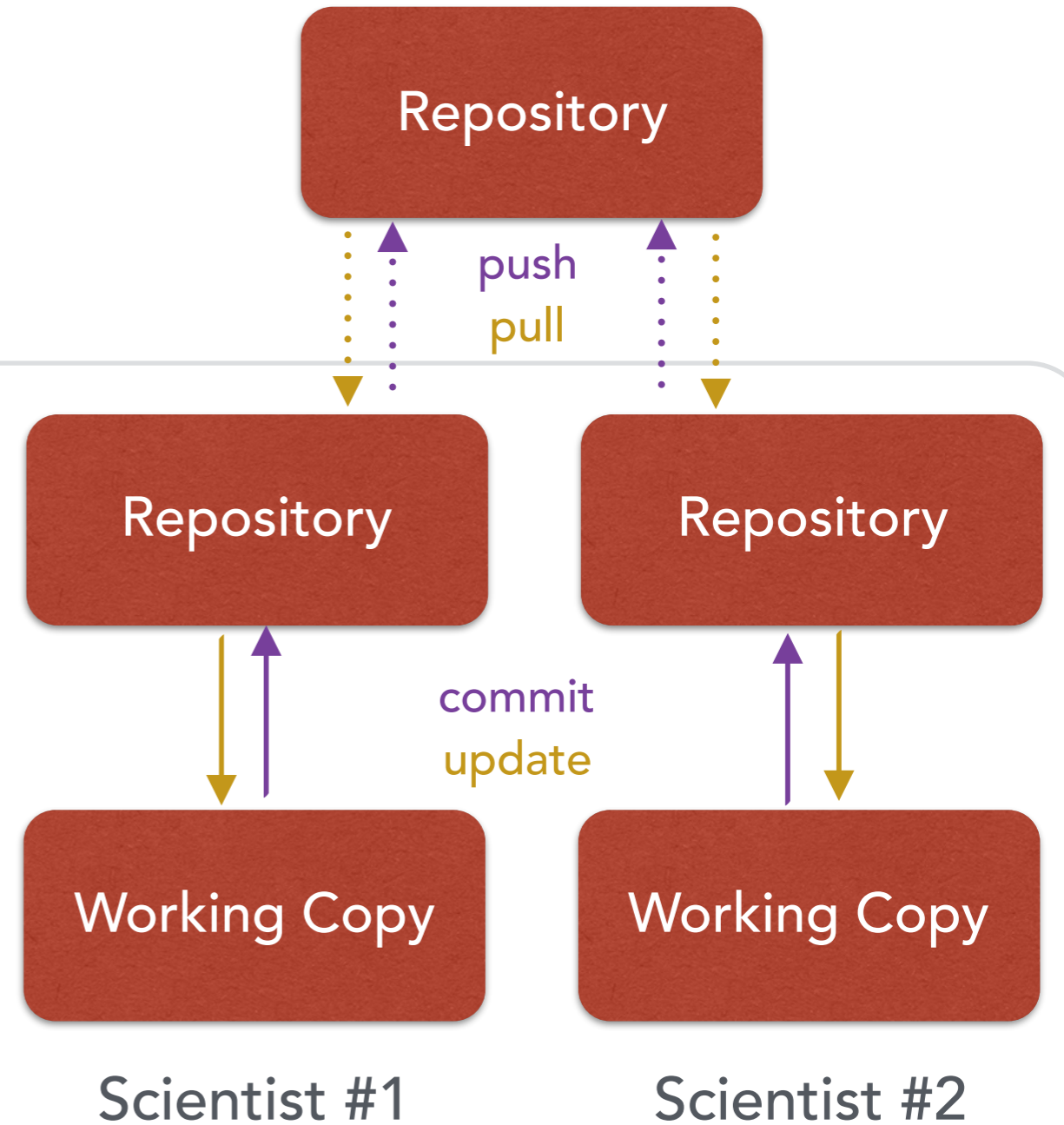
- Full copies of code repositories are made to each user's workspace
- All commits are done locally - no internet connection required!
- The most popular distributed VCS:



Centralized



Distributed



Local Machines

Github



- a web-based hosting service for version control using git
- WIPAC is on github!
 - ▶ <https://github.com/WIPACrepo>

The screenshot shows the GitHub profile for the Wisconsin IceCube Particle Astrophysics Center. The profile name is "Wisconsin IceCube Particle Astrophysics Center" and it is a "Research Center at the Univ. of Wisconsin—Madison". The location is "Madison, Wisconsin, U.S.A." and the website is "http://wipac.wisc.edu". Below the profile information, there are four statistics: "Repositories 74", "People 126", "Teams 14", and "Projects 0".

Github



- Home of many open source projects used in the scientific community
 - ▶ matplotlib
 - ▶ scipy
 - ▶ scikit-learn
 - ▶ etc.
- If you use them, consider contributing!
- Excellent git and github tutorial:
 - ▶ <https://www.youtube.com/watch?v=RrdECLvHW6g&feature=youtu.be&t=8342>