

# Version Control

## EOAS Software Carpentry Workshop

September 20th, 2016

# Automated Version Control

## Learning Goals

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

# "FINAL".doc



FINAL.doc!



FINAL\_rev.2.doc



FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc



FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc



FINAL\_rev.22.comments49.  
corrections.10.##\$%WHYDID  
ICOMETOGRADSCHOOL????.doc



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

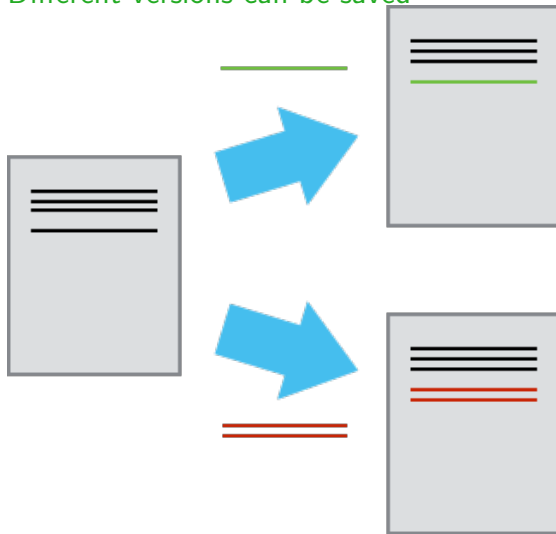
# Automated Version Control

Changes are saved sequentially



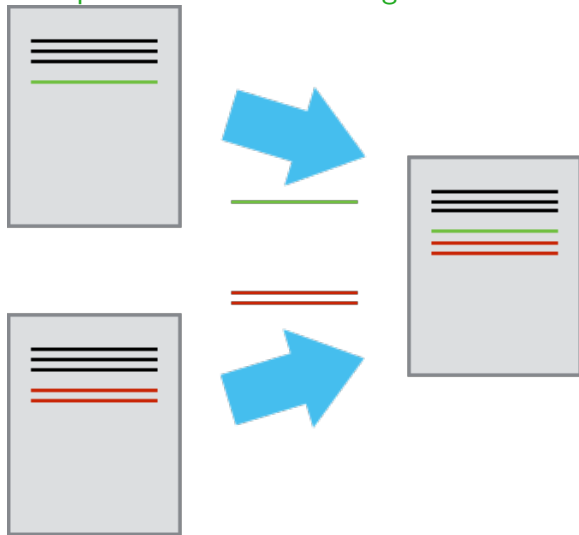
# Automated Version Control

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."
- hg commit plan.txt -m "Add note about Fraser river data source."

## 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 --rev 0 plan.txt
- hg revert 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`

# Remote Repositories

## Learning Goals

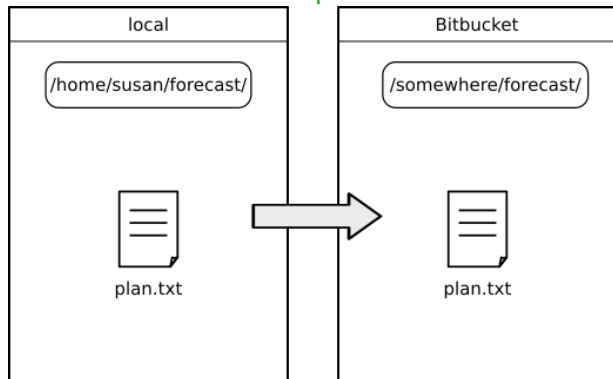
1. Explain what remote repositories are and why they are useful.
2. Explain what happens when a remote repository is cloned.
3. Explain what happens when changes are pushed to or pulled from a remote repository.

## Lesson Commands

- `hg push`
- `hg config --local`
- `hg paths`
- `hg pull`

# Remote Repositories

## Local and Bitbucket Repos After 1st Push



# Remote Repositories

## Learning Goals

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.

## Lesson Commands

- `hg push`
- `hg config --local`
- `hg paths`
- `hg pull`

# Exercise

Create a repository on Bitbucket, clone it, add a file, push those changes to Bitbucket. Look at the timestamp of the change on Bitbucket. How does Bitbucket record times, and why?

# Working with Clone Repositories

## Learning Goals

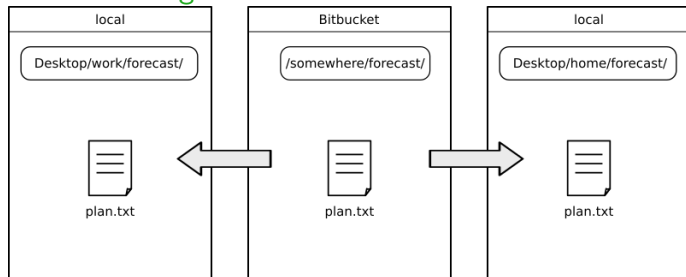
1. Explain how to push, pull, update files, and update metadata among clones of a repository.
2. Display a simple visualization of the state of a repository and explain how updating the repository affects its state.

## Lesson Commands

- `hg clone`
- `hg add`
- `hg commit`
- `hg push`
- `hg pull`
- `hg log --graph`
- `hg update`

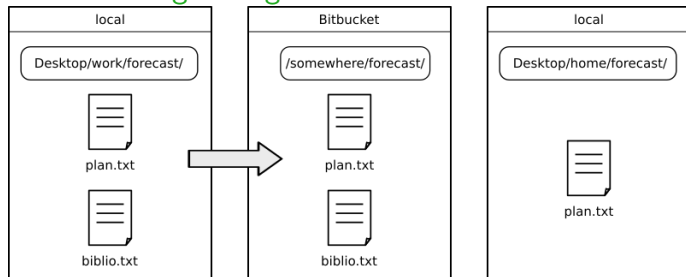
# Working with Clone Repositories

## After Creating work and home Clones



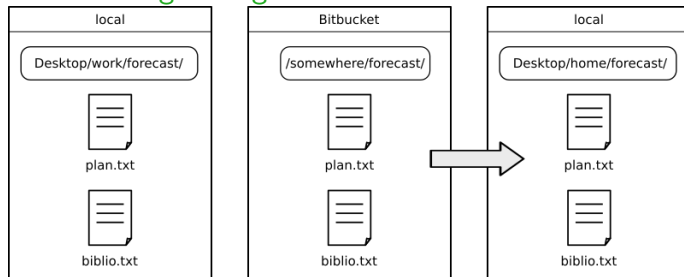
# Working with Clone Repositories

## After Pushing Change from work Clone



# Working with Clone Repositories

## After Pulling Change into home Clone



# Working with Clone Repositories

## Learning Goals

1. Explain how to push, pull, update files, and update metadata among clones of a repository.
2. Display a simple visualization of the state of a repository and explain how updating the repository affects its state.

## Lesson Commands

- `hg clone`
- `hg add`
- `hg commit`
- `hg push`
- `hg pull`
- `hg log --graph`
- `hg update`

# Collaboration

## Learning Goals

1. Explain the differences between public and private repositories on Bitbucket.
2. Configure user and group access settings for Bitbucket repositories.

# Merging Changes from Different Clones

## Learning Goals

1. Explain how Mercurial handles changes that make a repository's history diverge.
2. Explain what merges are.

## Lesson Commands

- hg commit
- hg push
- hg pull
- hg heads
- hg log -G
- hg merge
- hg status
- hg diff
- hg summary

# Merge Conflicts

## Learning Goals

1. Explain what merge conflicts are and when they can occur.
2. Resolve conflicts resulting from a merge using the KDiff3 tool.

## Lesson Commands

- `hg incoming`
- `hg pull`
- `hg update`
- `hg log --graph`
- `hg merge --tool=kdiff3`