PURDUE
UNIVERSITY

Day 1

Version Control Workshop: Git and GitHub

Cyrus Vandrevala [1]
Nicolás Guarín-Zapata[2]
[1] Physics Department
[2] Civil Engineering Department
October 30-31, 2014

Overview

Disclaimer

What is Version Control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

Why is Version Control Important?

1. Keep Track of Code History
2. Concurrent Teamwork
3. Coordinate Coding Environments
4. Due Diligence Checks
5. Share Code

Everybody Should Use Version Control!

What Options Are Available?

Option #1: Client-Server Version Control Systems

Advantages

1. A Single Admin Keeps Track of the Project
2. There is a Single Master Version of the Code
3. It is Relatively Easy to Learn

Disadvantages

1. There Is Only One Admin/Server
2. You Need a Network Connection to Work
3. Operations Can Be Slow

Examples include Concurrent Versions System (CVS) and Subversion (SVN).

What Options Are Available?

Option #2: Distributed Version Control Systems

Advantages

1. You Don't Need a Network Connection
2. Multiple Coding Environments
3. It Encourages Collaboration and Modularity

Disadvantages

1. Can Be Difficult to Learn
2. Teams Need to Talk About Conventions
3. It is Really Easy To Create Unorganized Code

Examples include Git/GitHub and Bazaar.

Why Git and GitHub?

1. It Keeps Track of Detailed Metadata (More Than Others)
2. Branching is Encouraged (Which Modularizes Development)
3. GitHub Has a Great Social Community

Why Git and GitHub?

Full Disclosure...

1. It Isn't the Best for Binary Files
2. GitHub Distinguishes Between Public and Private Repos

Version Control in Academia

1. It Creates Reproducible Research
2. It Helps Train New Group Members
3. It Encourages Collaboration
4. It Encourages Good Code Practices

Version Control in Academia

Some Useful Skills That You Should Learn Are:

1. Bash
2. Markdown

Setting Up Git - Linux

You can use the package management tool that comes with your distribution (use sudo):

1. yum install git
2. apt-get install git

Setting Up Git - Mac

There are three main ways to install Git:

1. Install the Xcode Command Line Tools and Type "git" Into the Terminal
2. Binary Installer: http://git-scm.com/download/mac
3. Git/GitHub GUI: https://mac.github.com/

Setting Up Git - Windows

There are three main ways to install Git:

1. Binary Installer: http://git-scm.com/download/win
2. msysGit: http://msysgit.github.io/
3. Git/GitHub GUI: https://windows.github.com/

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

The Git Cycle

1. Synchronize Version (git pull)
2. Make Changes to Code
3. Stage Changes for Commit (git add)
4. Commit Changes (git commit)
5. Push to Origin (git push)

Review Changes

View Log (git log)
View Staged Changes (git status)

Recovering Past Versions

Organizing Past Versions (MD5 Hash)
Recovering a Previous Version (git checkout)

What is Branching?

Create a New Branch (git checkout -b)

Merge Two Branches (git merge)

Synchronize Two Branches (git rebase)

List All Branches (git branch)

Delete a File (rm vs. git rm)

Delete a File (rm vs. git rm)

Undoing Changes (git revert vs. git reset)

Public vs. Private Repositories

Bitbucket and GitHub

Set Up Git Remote

Review the Git Cycle

Pull Requests

Forking a Repository

Thank you for your attention.