Version Control Workshop: Git and GitHub (Day 1)

Cyrus Vandrevala ¹
Nicolás Guarín-Zapata²
¹ Physics Department
² Civil Engineering Department

October 30-31, 2014



GitHub



Version Control Workshop: Git and GitHub (Day 1)







Overview

- Introduction to Version Control
- 2 Work Flow in Computational Science
- Setting Up Git On Your Machine
- 4 The Basic Git Work Flow
- 6 Git Branches
- 6 Git Delete Commands
- Combining Git With GitHub



2014-10-30

└─Overview

Version Control Workshop: Git and GitHub (Day 1)

Overview

Introduction to Version Control

Work Flow in Computational Science
Setting Up Git On Your Machine

The Basic Git Work Flow

Git Branches

Git Delete Commands

Combining Git With GitHub

We Encourage Participation!

- Post Questions That You Might Have in the Repo
- Recommend Other Sources That You Found Useful
- Remember, We Do Not Know Everything!



─We Encourage Participation!

We Encourage Participation!

- Post Questions That You Might Have in the Repo Recommend Other Sources That You Found Useful
 - Remember, We Do Not Know Everything!

What is Version Control?

Introduction

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.

- Pro Git, Chapter 1



Version Control Workshop: Git and GitHub (Day 1)

Introduction to Version Control

—What is Version Control?

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.

- Pro Git, Chapter 1

What is Version Control?

Introduction

It allows you to revert files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

- Pro Git, Chapter 1



Version	${\sf Control}$	Wo	rkshop:	${\sf Git} {\sf and} $	GitHub	(Day 1)	
└_Intro	duction	to \	Version	Control			

└─What is Version Control?

What is Version Control?

It allows you to revert files back to a previous state, revert the entire project back to a previous state, compase changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

- Pro Git, Chapter 1

 One of the first major version control systems was the Revision Control System (RCS). It was released by Walter F. Tichy while he was at Purdue University (RCS: A System for Version Control).

- Keep Track of Code History
- 2 Concurrent Teamwork
- 3 Coordinate Coding Environments
- Oue Diligence Checks
- Share Code



Introduction

Why is Version Control Important?

- Keep Track of Code History
- 2 Concurrent Teamwork

- Share Code



- Keep Track of Code History
- Concurrent Teamwork
- 3 Coordinate Coding Environments
- Oue Diligence Checks
- Share Code



- Keep Track of Code History
- Concurrent Teamwork
- 3 Coordinate Coding Environments
- Oue Diligence Checks
- Share Code



- Keep Track of Code History
- Concurrent Teamwork
- 3 Coordinate Coding Environments
- Oue Diligence Checks
- Share Code



Introduction

Basic Git

GitHub

Deleting

Why is Version Control Important?

- Keep Track of Code History
- Concurrent Teamwork
- 3 Coordinate Coding Environments
- Oue Diligence Checks
- Share Code

Everybody Should Use Version Control!



Keep Track of Code History
 Concurrent Teamwork

Coordinate Coding Environments
Due Diligence Checks
Share Code

Everybody Should Use Version Controll

Work Flow

What Options Are Available?

Option #1: Client-Server Version Control Systems

Advantages

Introduction

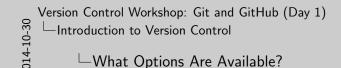
- A Single Admin Keeps Track of the Project
- There is a Single Master Version of the Code
- It is Relatively Easy to Learn

Disadvantages

- There Is Only One Admin/Server
- You Need a Network Connection to Work
- Operations Can Be Slow

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





What Options Are Available?

of the Project

Option #1: Client-Server Version Control Systems Disadvantages A Single Admin Keeps Track There Is Only One

Admin/Server There is a Single Master You Need a Network Version of the Code Connection to Work lt is Relatively Easy to Learn Operations Can Be Slow

Examples include Concurrent Versions System (CVS) and Subversion

Work Flow Setting Up Git Basic Git Branching Deleting GitHub

What Options Are Available?

Option #2: Distributed Version Control Systems

Advantages

Introduction

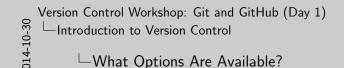
- You Don't Need a Network Connection
- Multiple Coding Environments
- It Encourages Collaboration and Modularity

Disadvantages

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

Examples include Git, Mercurial, and Bazaar.





What Options Are Available?

Option #2: Distributed Version Control Systems Advantages Disadvantages You Don't Need a Network Can Be Difficult to Learn

Connection Multiple Coding Environments It Encourages Collaboration

and Modularity

Teams Need to Talk About Conventions a It is Really Easy To Create Unorganized Code

Examples include Git, Mercurial, and Bazzar.

Why Git and GitHub?

- It Keeps Track of Detailed Metadata (More Than Others)
- Branching is Encouraged (Which Modularizes Development)
- Most Operations in Git are Local (Which Increases Speed)
- GitHub Has a Great Social Community





Why Git and GitHub?

- It Keeps Track of Detailed Metadata (More Than Others)
 Branching is Encouraged (Which Modularizes Development)
 - Most Operations in Git are Local (Which Increases Speed)
 GitHub Has a Great Social Community

Why Git and GitHub?

Introduction

Full Disclosure...

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



Why Git and GitHub?

Full Disclosure...

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

Version Control in Academia

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



└Version Control in Academia

Version Control in Academia

- It Creates Reproducible Research
 It Helps Train New Group Members
- It Helps Train New Group Members
 It Encourages Collaboration
- It Encourages Good Code Practices

Branching

Reproducible Research

Introduction

Purdue is already taking action in this subject:

 Purdue University Research Repository (PURR): https://purr.purdue.edu/

We can create a DOI for a GitHub repository using zenodo.org:

• https://guides.github.com/activities/citable-code/



Reproducible Research

Reproducible Research

Purdue is already taking action in this subject: • Purdue University Research Repository (PURR): https://purr.purdue.edu/

We can create a DOI for a GitHub repository using zenodo.org: • https://guides.github.com/activities/citable-code/

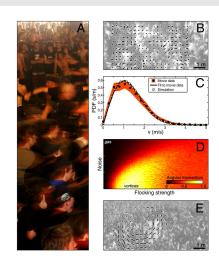
Reproducible Research

Introduction

A Cool Example:

Silverberg, Jesse L., et al. "Collective motion of humans in mosh and circle pits at heavy metal concerts." Physical review letters 110.22 (2013): 228701.

Repo: https://github.com/ mattbierbaum/moshpits





Version Control Workshop: Git and GitHub (Day 1) -Work Flow in Computational Science

Reproducible Research

Reproducible Research

A Cool Example:

Silverberg, Jesse L., et al. 'Collective motion of humans in mosh and circle pits at heavy metal concerts." Physical review letters 110.22 (2013): 228701. Repo: https://github.com/ mattbierbaum/moshpits





Branching

Some Useful Skills

Some Useful Skills That You Should Learn Are:

- Bash
- Markdown
- Vim and/or Emacs



Some Useful Skills

Some Useful Skills That You Should Learn Are:

- Bash
- Bash
 Markdown
- Vim and/or Emacs

Branching

GitHub

Setting Up Git - Linux

Introduction

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

- yum install git
- apt-get install git



Version Control Workshop: Git and GitHub (Day 1)

Setting Up Git On Your Machine

Setting Up Git - Linux

Setting Up Git - Mac

Introduction

There are three main ways to install Git:

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



• The GUI only implements a subset of the full Git functionality, so it is best to learn how to use the command line.

Branching

Setting Up Git - Windows

Introduction

There are three main ways to install Git:

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



Setting Up Git - Windows

Setting Up Git - Windows

These are three main ways to install Git

George Installer. http://git.erm.com/download/win

myorick: http://myoricy.github.io/
GG(GIAbb GUV http://windows.github.com/

• The GUI only implements a subset of the full Git functionality, so it is best to learn how to use the command line.

Introduction

Branching

GitHub

Setting Up Git - Installing From Source

You can also install GitHub from source. See the Git website for full instructions on how to do that.



Setting Up Git - Installing From Source

http://git-scm.com/

Branching

Setting Up Git - Config File

Introduction

Git stores user information in /etc/gitconfig, /.gitconfig, and /your-project/.git/config. To set up your information:

- git config --global user.name "Cyrus Vandrevala"
- git config --global user.email cyrus.vandrevala@gmail.com
- git config --global core.editor vim



Version Control Workshop: Git and GitHub (Day 1)

—Setting Up Git On Your Machine

└─Setting Up Git - Config File

Setting Up Git - Config File

Git stores user information in /etc/gitconfig, /-gitconfig, and /your-project/-git/config. To set up your information:

git config —global user.name "Cyrus Vandrevala" eit confie —elobal user.email cyrus vandrevala@mail.com

ø git config -global core editor vim

GitHub

Setting Up Git - Config File

Introduction

You can double check the information you entered by using:

git config --list



Setting Up Git - Config File

You can double check the information you entered by using:

• git config – Siz

Setting Up a New Git Repo

Introduction

- Create a New Directory (mkdir my-awesome-directory)
- Navigate Into the Directory (cd my-awesome-directory)
- Initialize the Directory (git init)

The git init command creates a hidden directory called .git that contains all of the metadata for the project. You should never change anything in .git directly!





Setting Up a New Git Repo

◆ Create a New Directory (mkdir my-awesome-directory)
 ◆ Navigate Into the Directory (cd my-awesome-directory)
 ◆ Initialize the Directory (git init)

The git init command creates a hidden directory called .git that contains all of the metadata for the project. You should never change anything in .git directly!

Retrieving an Existing Git Repo

- Navigate to the Directory Where You Want to Store the Project
- 2 Run git clone https://mydirectory.com/

- Git supports many transfer protocols (including SSH)
- Remember, you are creating a standalone copy of the entire project.



Retrieving an Existing Git Repo

 Navigate to the Directory Where You Want to Store the Project

■ Run git clone https://mydirectory.com/

Nun git clone https://mydirectory.com/

 Git supports many transfer protocols (including SSH)
 Remember, you are creating a standalone copy of the entire project.

- Synchronize Your Repo (git pull)
- Make Changes to Your Code
- 3 Stage Changes for Commit (git add)
- Ommit Changes Locally (git commit)
- O Push Changes to Origin (git push)



- Synchronize Your Repo (git pull)
- Make Changes to Your Code
- Stage Changes for Commit (git add)
 Commit Changes Locally (git commit)
- Push Changes to Origin (git push)

Introduction

Files in your project can be in one of three states:

- Modified
- Staged
- Committed



The Basic Git Work Flow

Files in your project can be in one of three states:

• Modified
• Staged
• Committed

└─The Basic Git Work Flow

- Modified files have been changed on your computer, but they are not in the database yet.
- Staged files means that you have tagged a modified file to be included in the next commit.
- Committed files are safely stored in your local database.

Introduction

In order to determine which files are in which state, you can use (most to least detail):

- git diff (unstaged changes only)
- @ git status
- git status -s



In order to determine which files are in which state, you can use (most to least detail):

- git diff (unstaged changes only)
- git status
- 🜢 git status -s

GitHub

The Basic Git Work Flow

Introduction

In order to get a full history of your commits, you can use:

git log

Every commit is labeled with a SHA-1 checksum.



Version Control Workshop: Git and GitHub (Day 1)

—The Basic Git Work Flow

└─The Basic Git Work Flow

The Basic Git Work Flow

In order to get a full history of your commits, you can use • git log Every commit is labeled with a SHA-1 checksum.

Branching

The Basic Git Work Flow

Introduction

In order to ignore certain files in your commits, you can change:

.gitignore

There are lots of .gitignore templates online at: https://github.com/github/gitignore



The Basic Git Work Flow In order to ignore certain files in your commits, you can change gitignore There are lots of .gitignore templates online at:

https://github.com/github/gitignore

Branching

The Basic Git Work Flow

Shortcuts:

- git commit -m "My message" Commit with a message.
- git commit -a -m "My message" Commit without staging with a message.



Shortcur

git commit -m "My message"
Commit with a message.

 git commit -a -m "My message"
Commit without staging with a message.

- Pretty much every version control system has some form of branching. This means that you diverge from the main line of development and continue to do work without changing the main line.
- Usually this is an expensive process because you have to copy all of the source code in the directory into a new branch.
- However, branching is where git truly shines. The git branch is extremely lightweight. This encourages branching in order to add new features



- Pretty much every version control system has some form of branching. This means that you diverge from the main line of development and continue to do work without changing the main line.
- Usually this is an expensive process because you have to copy all of the source code in the directory into a new branch.
- However, branching is where git truly shines. The git branch is extremely lightweight. This encourages branching in order to add new features.



- Pretty much every version control system has some form of branching. This means that you diverge from the main line of development and continue to do work without changing the main line.
- Usually this is an expensive process because you have to copy all of the source code in the directory into a new branch.
- However, branching is where git truly shines. The git branch is extremely lightweight. This encourages branching in order to add new features.



	Version Control Workshop: Git and GitHub (Day 1
-30	Git Branches
1-10	
014-	└─What is Branching?

- Pretty much every version control system has some form of branching. This means that you diverge from the main line of development and continue to do work without changing the main line.
- Usually this is an expensive process because you have to copy all of the source code in the directory into a new branch.
- However, branching is where git truly shines. The git branch is extremely lightweight. This encourages branching in order to add new features.

Branching

GitHub

How Does Branching Work?

Introduction

Let's look at a couple of examples from Pro Git (2nd Edition). This book is licensed under the Creative Commons Attribution Non-Commercial Share Alike 3.0 License.

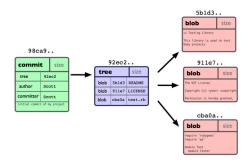


Let's look at a couple of examples from Pro Git (2nd Edition).

This book is licensed under the Creative Commons Attribution

Non-Commercial Share Alike 3.0 License.

How Does Branching Work?

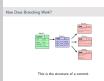


This is the structure of a commit.



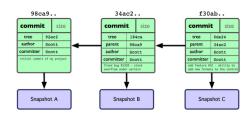
Version Control Workshop: Git and GitHub (Day 1) —Git Branches

└─How Does Branching Work?



How Does Branching Work?

Introduction



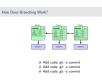
Add code; git -a commit # Add code; git -a commit # Add code; git -a commit



2014-10-30

Version Control Workshop: Git and GitHub (Day 1) —Git Branches

└─How Does Branching Work?

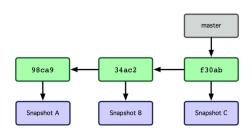


Introduction

GitHub

Deleting

How Does Branching Work?



Every project starts off with a master branch.



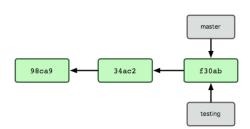
⊢How Does Branching Work?

How Does Branching Work?

Branching

Basic Git

How Does Branching Work?



git branch testing



Version Control Workshop: Git and GitHub (Day 1) 2014-10-30 Git Branches

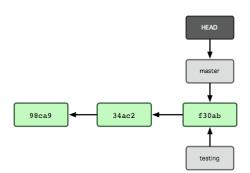
⊢How Does Branching Work?

How Does Branching Work? git branch testing

GitHub

How Does Branching Work?

Introduction



HEAD is still on the master branch.



Git Git

Version Control Workshop: Git and GitHub (Day 1) —Git Branches

└─How Does Branching Work?

New Does Branching Work?

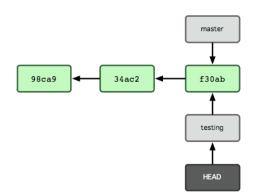
Introduction

Branching

Basic Git

GitHub

How Does Branching Work?



git checkout testing

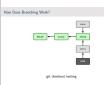


2014-10-30

Version Control Workshop: Git and GitHub (Day 1)

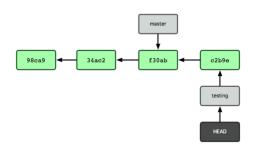
Git Branches

└─How Does Branching Work?



How Does Branching Work?

Introduction



Add new code to testing git -a commit



2014-10-30

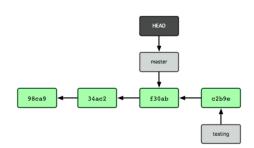
Version Control Workshop: Git and GitHub (Day 1) Git Branches

└─How Does Branching Work?



Branching

How Does Branching Work?



git checkout master



Version Control Workshop: Git and GitHub (Day 1) 2014-10-30 ⊢How Does Branching Work?

Git Branches

How Does Branching Work? git checkout master

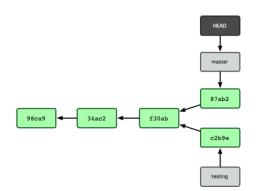
Branching

Basic Git

GitHub

How Does Branching Work?

Introduction



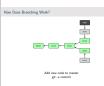
Add new code to master git -a commit



2014-10-30

Version Control Workshop: Git and GitHub (Day 1) Git Branches

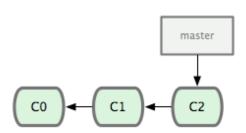
└─How Does Branching Work?



GitHub

How Does Merging Work?

Introduction



Suppose we have a project with a few current commits.



└─How Does Merging Work?

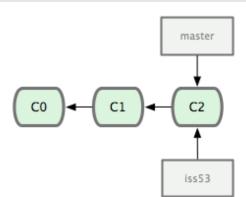
How Does Merging Work?

Branching

Basic Git

GitHub

Introduction



git checkout -b iss53 (git branch iss53; git checkout iss53)



Version Control Workshop: Git and GitHub (Day 1) 2014-10-30 Git Branches

└─How Does Merging Work?

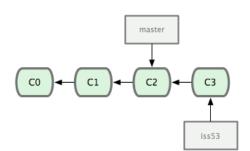


Branching

GitHub

How Does Merging Work?

Introduction



Add code to iss53 git -a commit



Version Control Workshop: Git and GitHub (Day 1)

Graph Git Branches

How Does Merging Work?

How Does Merging Work?

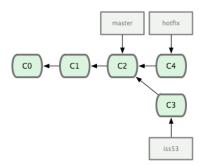
(a) — (a) — (b) — (c) —

Introduction

Branching

Basic Git

How Does Merging Work?



git checkout master git checkout -b hotfix Add code to hotfix git -a commit

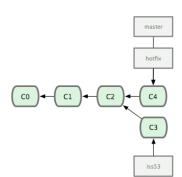


Version Control Workshop: Git and GitHub (Day 1) 2014-10-30 Git Branches

└─How Does Merging Work?



GitHub



git checkout master git merge hotfix



2014-10-30

Version Control Workshop: Git and GitHub (Day 1)

Git Branches

└─How Does Merging Work?

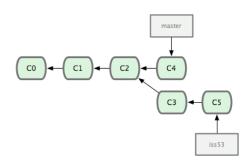


Branching

Basic Git

How Does Merging Work?

Introduction



git branch -d hotfix git checkout iss53 # Add code to iss53 branch git -a commit



Version Control Workshop: Git and GitHub (Day 1) Git Branches

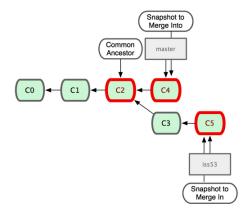
└─How Does Merging Work?

How Does Merging Work? git checkout isa53 # Add code to isa53 branch

Basic Git

How Does Merging Work?

Introduction



We want to merge iss53 to master



Version Control Workshop: Git and GitHub (Day 1)

—Git Branches

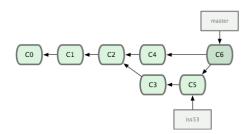
└─How Does Merging Work?



Basic Git

GitHub

How Does Merging Work?



git checkout master git merge iss53



Version Control Workshop: Git and GitHub (Day 1)

Git Branches

└─How Does Merging Work?

How Does Merging Work?

I The Company of the Compan

Branching

Merge Conflicts

Introduction

\$ git merge iss53 Auto-merging index.html CONFLICT (content): Merge conflict in index.html Automatic merge failed; fix conflicts and then commit the result.

- Sometimes we run into merge conflicts
- git status is useful in these cases



Version Control Workshop: Git and GitHub (Day 1)

Git Branches

└─Merge Conflicts

Merge Conflicts

git merge iso53 Auto-merging index.html COMFALCT (content): Merge conflict in index.html Automatic merge failed; fix conflicts and then commit the

Sometimes we run into merge conflicts
 git status is useful in these cases

Merge Conflicts

Introduction

```
<<<<< html
<div id="footer">contact :
email.support@github.com</div>
======
<div id="footer">
 please contact us at
support@github.com
</div>
>>>>> iss53:index.html
```

The "=====" divides the two types of code.



Version Control Workshop: Git and GitHub (Day 1) 2014-10-30 Git Branches

└─Merge Conflicts

Merge Conflicts

occcook MEAD: index.html email.supportPulthub.com</div> Odiv ide"footer"> c/divo >>>>>> issi3:index.html

The "---- divides the two types of code.

Introduction

Branching

GitHub

Deleting a File (rm vs. git rm)

- If you delete a file in your filesystem, you still need to commit your changes with git add file_removed.
- Or you can use git rm file_name.



Version Control Workshop: Git and GitHub (Day 1)

Git Delete Commands

-Deleting a File (rm vs. git rm)

Deleting a File (rm vs. git rm)

 If you delete a file in your filesystem, you still need to commit your changes with git add file_removed.
 Or you can use git rm file_name. Introduction

Branching

Deleting

GitHub

Deleting a File (rm vs. git rm)

If you rm a file, it will delete it locally, but it will still exist in your git directory. In order to fully delete a file, you need to use git rm



Version Control Workshop:	Git and GitHub (Day 1)	
☐Git Delete Commands		

If you rm a file, it will delete it locally, but it will still exist in your git directory. In order to fully delete a file, you need to use git rm

Deleting a File (rm vs. git rm)

└─Deleting a File (rm vs. git rm)

 More discussion at: http://stackoverflow.com/questions/7434449/why-use-git-rm-to-remove-a-file-instead-of-rm

Deleting

GitHub

Deleting a File

Introduction

If you want to delete a file that has been staged, but not committed use:

• git rm --cached



Version Control Workshop: Git and GitHub (Day 1)

Git Delete Commands

└─Deleting a File

Deleting a File

If you want to delete a file that has been staged, but not committed use:

• git rm —cached

Branching

Deleting

GitHub

If you want to move a file use:

• git mv

Moving a File

If you want to move a file use:

Introduction

GitHub

Discarding Changes to Unstaged Files

If you want to discard changes to unstaged files use:

• git checkout -- filename

Just keep in mind that branching is better practice...



Version Control Workshop: Git and GitHub (Day 1)

Git Delete Commands

└─Discarding Changes to Unstaged Files

Discarding Changes to Unstaged Files

If you want to discard changes to unstaged files use:

• git checkout — filename

Just keep in mind that branching is better practice....

Branching

Deleting

GitHub

Amending Staged Files

Introduction

In order to remove a file from the staged environment use:

• git reset filename



Version Control Workshop: Git and GitHub (Day 1)

Git Delete Commands

—Amending Staged Files

Amending Staged Files

In order to remove a file from the staged environment use:

• git reset filename

Branching

Deleting

GitHub

Amending Existing Commits

So you say you want to amend an existing commit? Why? I purposely didn't add anything here. Don't do it...



Version Control Workshop: Git and GitHub (Day 1)

Git Delete Commands

-Amending Existing Commits

Amending Existing Commits

So you say you want to amend an existing commit? Why?
I purposely didn't add anything here. Don't do it...

Branching

GitHub

Amending Commits

Introduction

Ok, fine...

• git commit --amend

But you are missing the point of version control...



Version Control Workshop: Git and GitHub (Day 1) —Git Delete Commands

└─Amending Commits

Amending Commits

Ok, fine...

• git commit --amend

But you are missing the point of version control...

Remote Repositories

Introduction

- Part of the strength of Git is linking your repository with other remote repositories
- We will mostly talk about this in the context of Git and GitHub



Remote Repositories

 Part of the strength of Git is linking your repository with other remote repositories

 We will mostly talk about this in the context of Git and GitHub

Branching

GitHub

View All Remote Repositories

In order to view all of the remote repositories for a project use:

• git remote -v



Version Control Workshop: Git and GitHub (Day 1)

Combining Git With GitHub

└─View All Remote Repositories

View All Remote Repositories

In order to view all of the remote repositories for a project use:

• git remote -v

Introduction

Branching

GitHub

Add Remote Repositories

In order to add a remote repository use:

git remote add some_url



Version Control Workshop: Git and GitHub (Day 1)

Combining Git With GitHub

Add Remote Repositories

Add Remote Repositories

In order to add a remote repository use: • git remote add some_url Introduction

Pull From Remote Repositories

In order to pull data from a remote repository use:

git fetch some_url git -a commit



In order to pull data from a remote repository use: git fetch some_url git -a commit Introduction

GitHub

Deleting

Push to Remote Repositories

In order to push data to a remote repository use:

git push remote_name branch_name



Version Control Workshop: Git and GitHub (Day 1) —Combining Git With GitHub

└─Push to Remote Repositories

Push to Remote Repositories

In order to push data to a remote repository use: git push remote_name branch_name

GitHub

GitHub Demo

Introduction

It's easier to demonstrate this than to write slides about it...





There is a lot more to learn! We did not discuss:

- Tagging
- Aliases

Introduction

- Advanced Remote Control
- The --hard Option
- Custom Environments
- Scripting and Extending Git
- And Much More!



Version Control Workshop: Git and GitHub (Day 1)

There is a lot more to learn! We did not discuss: Tagging

Aliases

 Advanced Remote Control . The -hard Option

Custom Environments

. Scripting and Extending Git a And Much Morel

Thank You For Your Attention.





Introduction

Version Control Workshop: Git and GitHub (Day 1) —Combining Git With GitHub

Thank You For Your Attention.