Introduction to Git

Northeastern University IEEE
Taylor Skilling
Fall 2016

Outline

- What is Git?
- Why use Git?
- Installation and Setup
- Overview
- Example Workflow
- Introduction to Branching/Merging
- Further Reading and Questions

What is Git?

“An unpleasant or contemptible person” - Google

“The stupid content tracker” - Linus Torvalds

“An awesome way to track and share code”

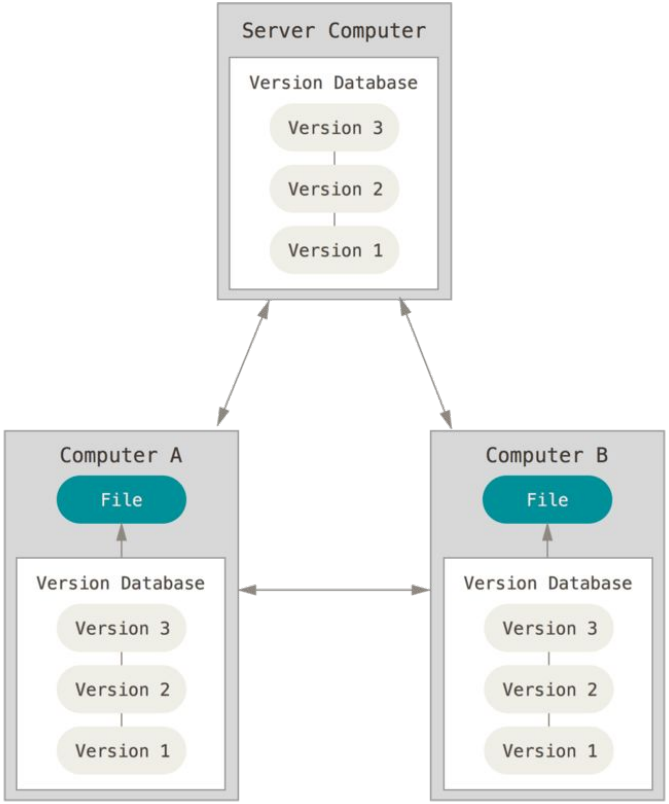
- Everyone else

In Short: A Version Control System (VCS)

Version Control Systems

Git is a Distributed VCS:

- Fast
- Simple
- Strong
- Robust



Creates snapshots, **not** differences

Why Git?

Local Operations

Integrity through checksums

Git only *adds* data

Relatively simple

Tons of support

Of course… coop

https://github.com/skillingt/IEEE

What about Github?

Web-based Git repository hosting service

Written in Ruby, Launched in 2008

Graphical User Interface to the typical command line Git

WHAT CAN GITHUB BE USED FOR?

A place to store your code!

Downloading libraries and drivers

Documentation

Graphics to analyze contributions

Website Hosting

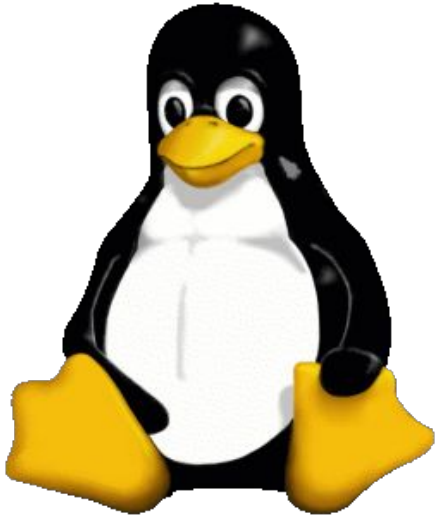
Showing off your personal project

BACK TO GIT AND THE COMMAND LINE!

Installation:

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Support for Windows, Mac and Ubuntu



Setup your git

```
git config --global user.name “YOUR NAME”

git config --global user.email “your@email”

git config --global core.editor vim
```

View your configuration settings:

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git config --list
user.name=Taylor Skilling
user.email=skilling.t@husky.neu.edu
core.editor=vim
```

Start/Clone A Project

Forking a github Repository:
https://github.com/skillingt/IEEE.git

Create a Local Clone of Your Fork:

git clone <URL>

To start your own repository:

- git init <repository name> **OR**
- Create a new project on https://github.com and clone
- This creates a .git subdirectory holding internal details

Getting help

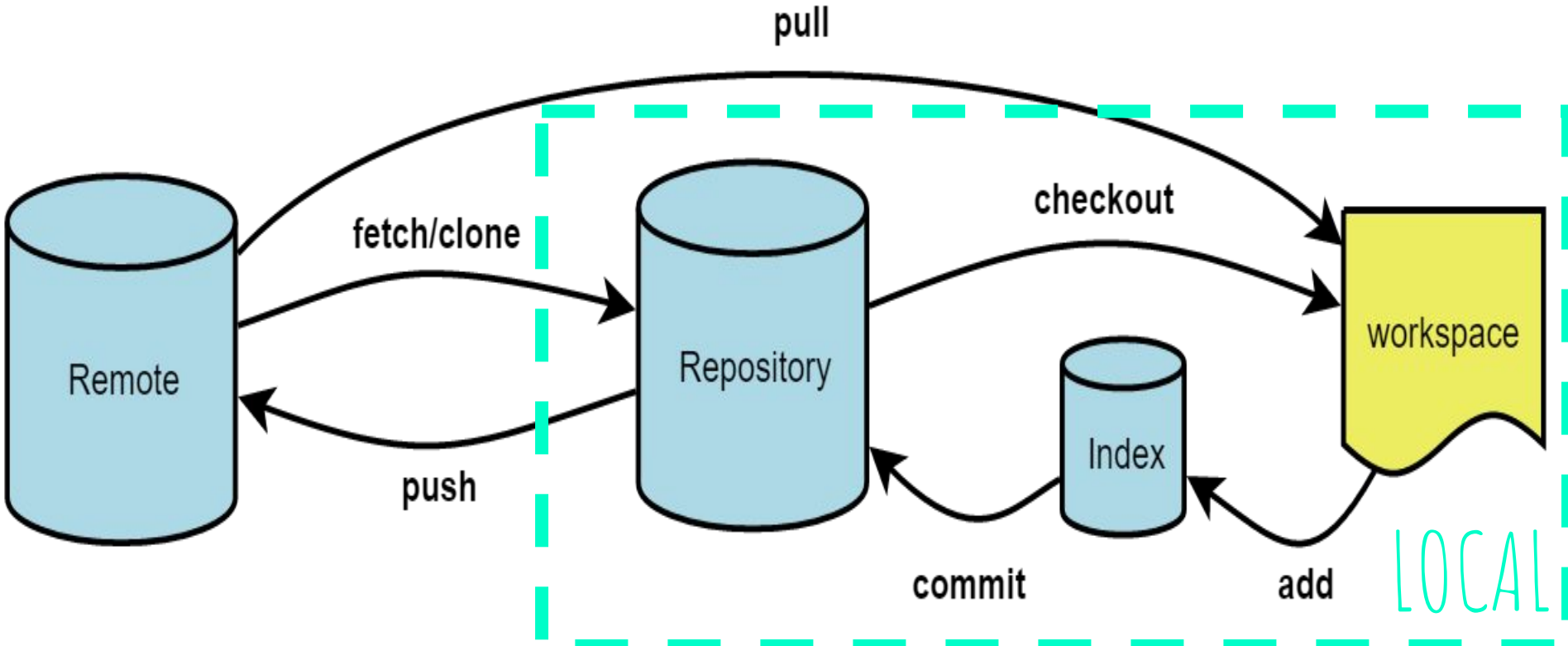
```
git help <verb>

git <verb> --help

man git <verb>
```

Example:

```
git help push
```

```
GIT-PUSH(1)                        Git Manual                        GIT-PUSH(1)

NAME
       git-push - Update remote refs along with associated objects

SYNOPSIS
       git push [--all | --mirror | --tags] [-n | --dry-run] [--receive-pack=<git-receive-pack>]
                [--repo=<repository>] [-f | --force] [--prune] [-v | --verbose] [-u | --set-upstream]
                [<repository> [<refspec>...]]

DESCRIPTION
       Updates remote refs using local refs, while sending objects necessary
       to complete the given refs.

       You can make interesting things happen to a repository every time you
       push into it, by setting up hooks there. See documentation for git-
       receive-pack(1).

OPTIONS
       <repository>
           The "remote" repository that is destination of a push operation.
           This parameter can be either a URL (see the section GIT URLS below)
           or the name of a remote (see the section REMOTES below).
```

OVERVIEW

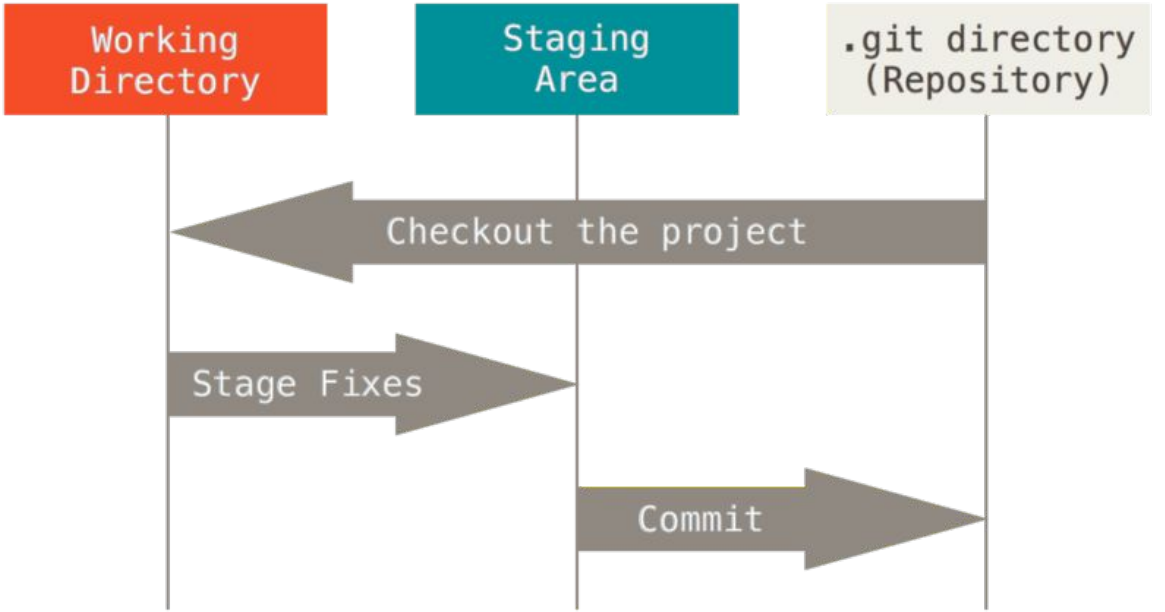

pull
fetch/clone
checkout
Remote
Repository
workspace
Index
push
commit
add
LOCAL

Working Locally

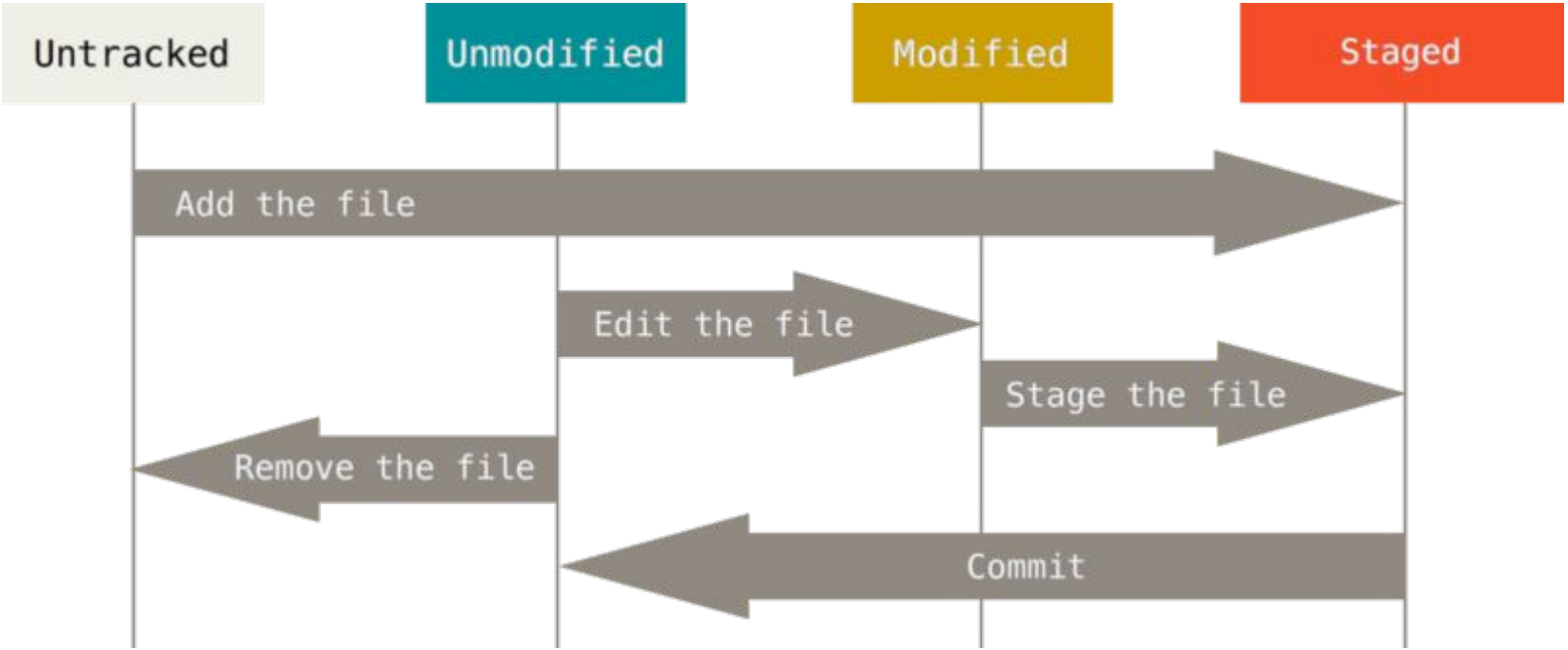
```
git branch <branchname>    (Create a new branch)
git checkout <branchname>  (Move to that new branch)
git add <filename(s)>      (Add files to the local repo)
git commit <filename(s)>   (Tell repo of changes)
git status                 (View what stage my files are in)
git diff <filename>        (View difference between commits)
git log                    (View history of commits)
```

Basic Local Workflow

1. Checkout a project
2. Make changes to one or many files
3. Add these files to the staging area or index
4. Commit these files to the repository



Stages of Files



Checking the Status

`git status`

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
nothing to commit (working directory clean)
```

Simply Clean Housekeeping

Creating a file

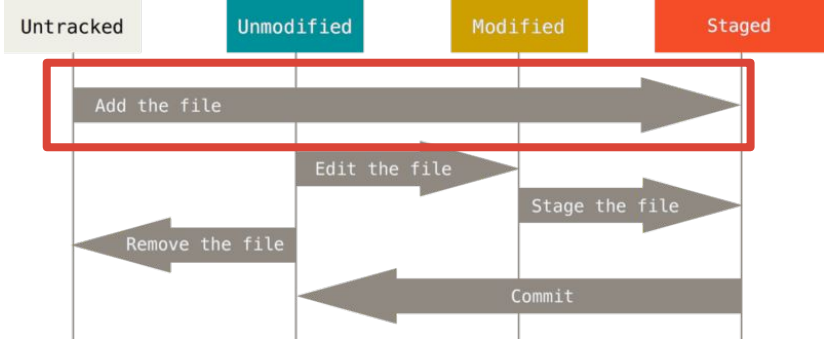
Create a new file!

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ vim HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ g++ -o HelloWorld HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ ./HelloWorld
Hello World!
```

Compile and Run

```
// Compile with g++ -o HelloWorld HelloWorld.cpp
// Run with ./HelloWorld from the command line

#include <iostream>

int main()
{
  std::cout << "Hello World!\n";
}
```

ADD IT



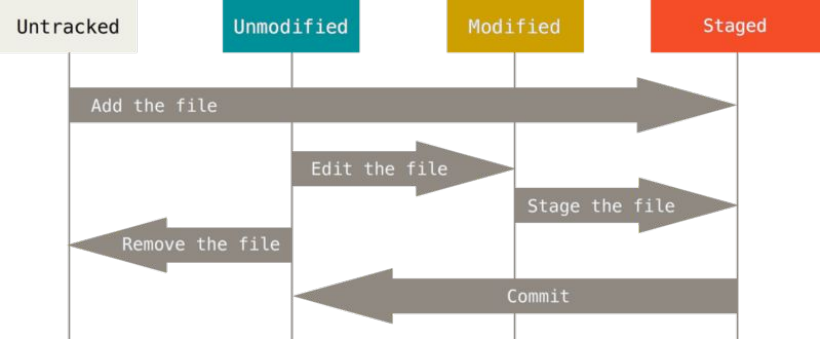
First, **check the status**

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
#       HelloWorld.cpp
nothing added to commit but untracked files present (use "git add" to track)
```

Now let's add it to the staging area

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git add HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   HelloWorld.cpp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
```

Modify again




```
// Compile with g++ -o HelloWorld HelloWorld.cpp
// Run with ./HelloWorld from the command line

#include <iostream>

int main()
{
  std::cout << "Hello World!\n";
  std::cout << "Git is the best...\n";
}
```

Added line

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ vim HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   HelloWorld.cpp
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   HelloWorld.cpp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
```

Difference Between Files

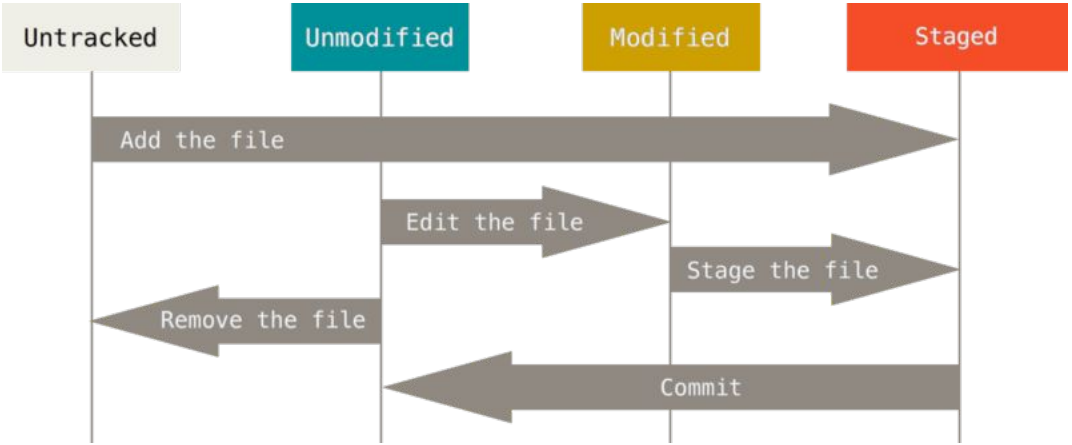
Use the command:

git diff

Shows the difference
between working directory and staged (unstaged changes)

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ vim HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git diff
diff --git a/HelloWorld.cpp b/HelloWorld.cpp
index 7431343..092d850 100644
--- a/HelloWorld.cpp
+++ b/HelloWorld.cpp
@@ -7,4 +7,5 @@ int main()
 {
   std::cout << "Hello World!\n";
   std::cout << "Git is the best...\n";
+  std::cout << "Can I see a difference?\n";
 }
```

Add it Again




```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git add HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   HelloWorld.cpp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
```

Difference Between Files (continued)

Use the command:

`git diff --staged`

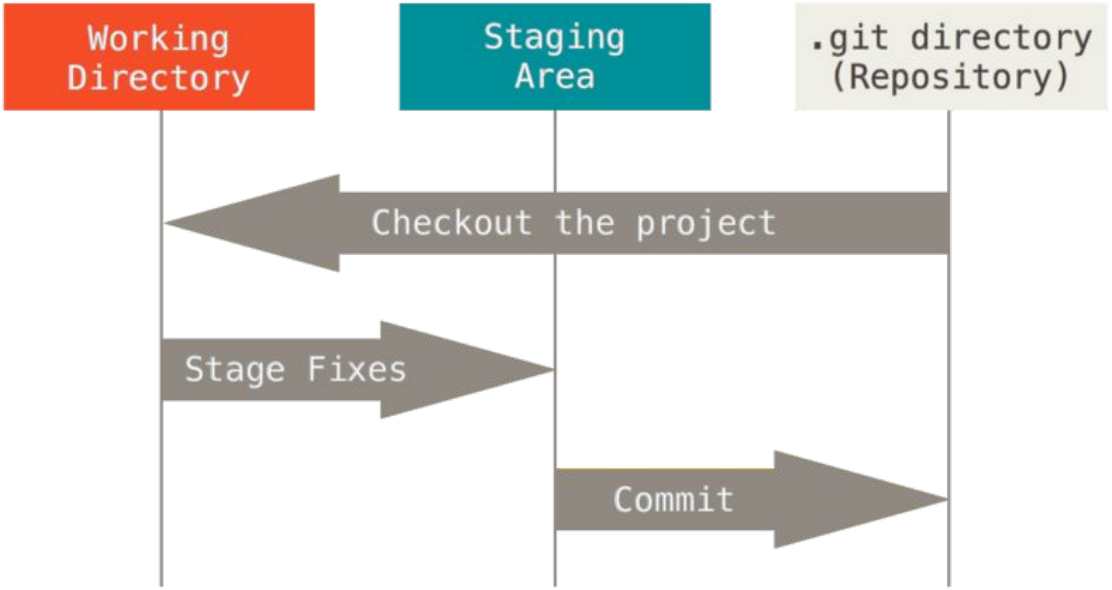
Shows the difference between what will go into the next commit and what is currently **staged**

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git diff --staged
diff --git a/HelloWorld.cpp b/HelloWorld.cpp
index 7431343..092d850 100644
--- a/HelloWorld.cpp
+++ b/HelloWorld.cpp
@@ -7,4 +7,5 @@ int main()
 {
   std::cout << "Hello World!\n";
   std::cout << "Git is the best...\n";
+  std::cout << "Can I see a difference?\n";
 }
```

Commit

What's this?

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git commit -m "Added HelloWorld.cpp"
[master a2dc305] Added HelloWorld.cpp
 1 file changed, 10 insertions(+)
 create mode 100644 HelloWorld.cpp
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
nothing added to commit but untracked files present (use "git add" to track)
```



Removing and Ignoring files

Oops… I accidentally added the executable…

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git add HelloWorld
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   HelloWorld
#       modified:   HelloWorld.cpp
#
```

I can remove with git rm

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git rm HelloWorld
error: 'HelloWorld' has changes staged in the index
(use --cached to keep the file, or -f to force removal)
```

What's wrong?

- Use -f to delete from hard drive
- Use --cached to simply remove it from being tracked

Removing and Ignoring (Continued)

Using git rm --cached

My file is still safe

But what if I never want to see it ever again?

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git rm --cached HelloWorld
rm 'HelloWorld'
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   HelloWorld.cpp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       HelloWorld
```

Add it to .gitignore!

Removing and Ignoring (Continued)

This file says:

- Ignore HelloWorld
- Don't ignore HelloWorld.cpp
- Ignore all .pyc files

Now we'll never have to worry about HelloWorld

```
IEEE — vim — 41×9
HelloWorld
!HelloWorld.cpp
*.pyc
~
~
~
~
~
```

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ vim .gitignore
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   HelloWorld.cpp
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .gitignore
```

You can also add .gitignore to .gitignore

Viewing previous Commits and History

git log

commit checksum

commits

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git log
commit 1fadcd179d8b7f9ad50f1dc803a0708dc68acded
Author: Taylor Skilling <skilling.t@husky.neu.edu>
Date:   Mon Nov 7 17:00:46 2016 -0500

    Merge branch 'master' of https://github.com/skillingt/IEEE

commit 0317d282414b0a969ef25e9e3b193ab73b5d668b
Author: Taylor Skilling <skilling.t@husky.neu.edu>
Date:   Mon Nov 7 16:07:42 2016 -0500

    Added time library

commit 2c27f5598b52830b0c88d2992b6e6233028707d3
Author: kafr15 <afriyie.k@husky.neu.edu>
Date:   Mon Nov 7 16:06:30 2016 -0500

    added function to tell time and date

commit 1a13c5d4d730174d74f5b6ec6a6eaa68fb73825f
Author: Taylor Skilling <skilling.t@husky.neu.edu>
Date:   Mon Nov 7 15:48:29 2016 -0500

    Created HelloWorld.py

commit ac572b6d757461bf7110ed3be6803174e3fbc764
Author: skillingt <skilling.t@husky.neu.edu>
Date:   Mon Nov 7 15:30:47 2016 -0500
```

Modifying

What if I forgot a line in my commit message, or I forgot to add a file to a commit?

Simple! Just use:

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git log
commit 3d6998aeaf0fd8a95d628f3b23ef60fbb52418e9
Author: Taylor Skilling <skilling.t@husky.neu.edu>
Date:   Tue Nov 8 16:33:27 2016 -0500

    Simply HelloWorld.cpp commit
```

```
git commit --amend
```

```
IEEE — vim — 115×28
Simply HelloWorld.cpp commit
I forgot to say I removed a line!
```

```
commit be5b9f18f6d5b0b209cc79ced575c798cf211daa
Author: Taylor Skilling <skilling.t@husky.neu.edu>
Date:   Tue Nov 8 16:33:27 2016 -0500

    Simply HelloWorld.cpp commit
    I forgot to say I removed a line!
```

Reset

I added a file to the staging area… but I didn't mean to…

What now?

git reset HEAD <file>

git checkout -- <file>

Git is great and
tells us if we forget!

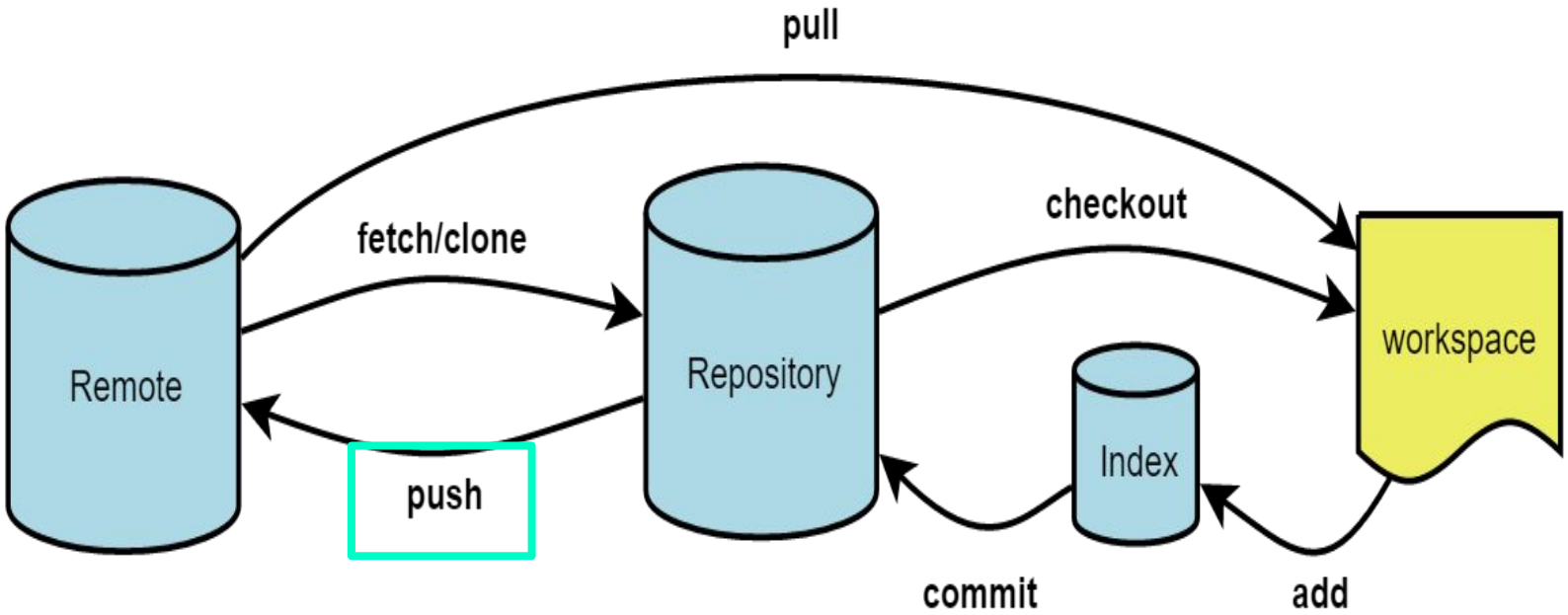
```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   test.txt
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory
#
#       modified:   HelloWorld.py
#
```

Moving Online: Push

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 436 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/skillingt/IEEE.git
   be5b9f1..66ce692  master -> master
```

`git push <remote> <branch>`

Push your changes to the remote so everyone can see them!



Moving Online: What is Remote?

Remote is where you push to and pull from

- Location of the repository on the Internet

Adding “Remotes” allow you to deploy codes to, or getting codes from multiple repositories

Use `git remote add origin <url>`

- “origin” is the standard name, but can be anything

Use `git remote -v` to view

Branches

In our examples:

branch = master

```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git branch
* master
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git branch development
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git branch
  development
* master
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git checkout development
Switched to branch 'development'
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git branch
* development
  master
```

But it's possible to have multiple branches!

git branch	Lists branches
git branch <branch_name>	Creates a new branch
git checkout <branch_name>	Switches to the desired branch

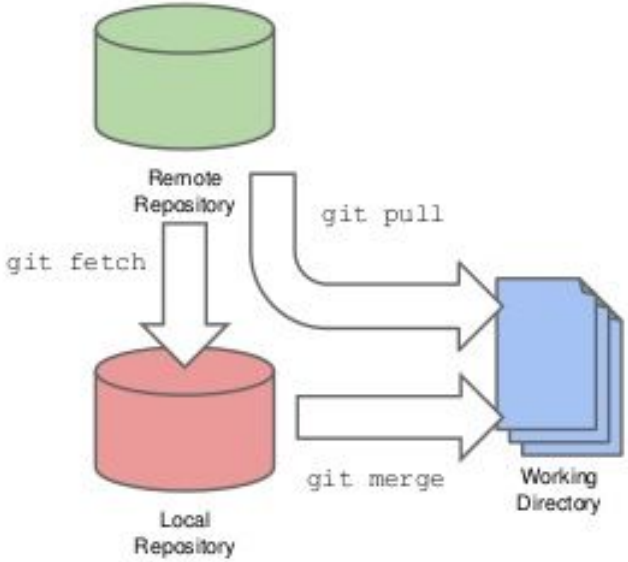
Fetch and Pull requests

`git fetch <remotename>`

Download changes from the remote branch, update data, but leave your branch unchanged

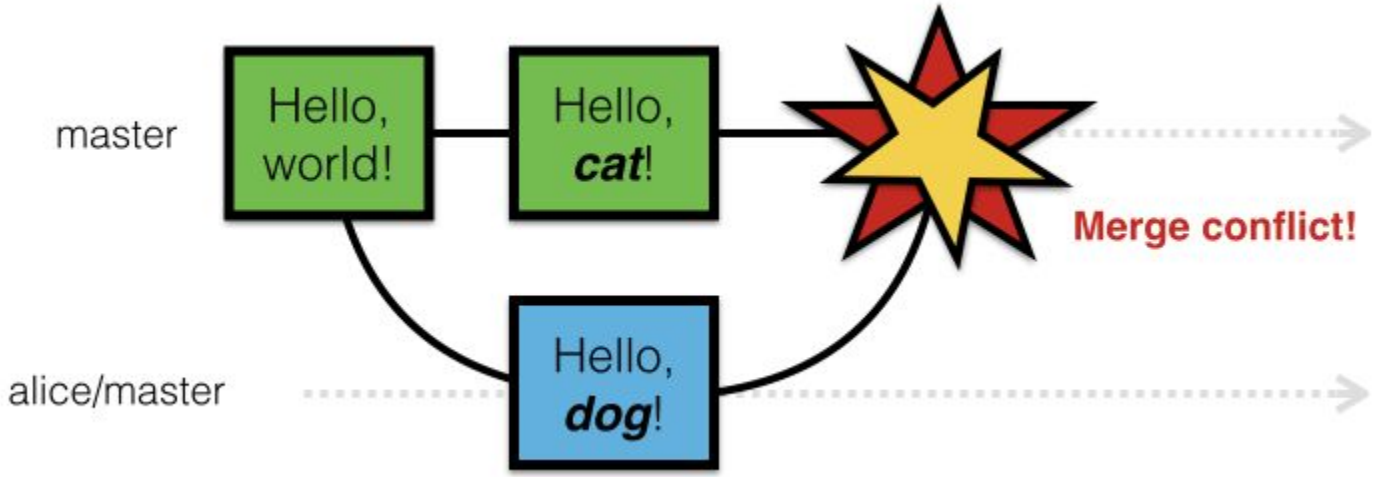
`git pull <remotename>`

Same as fetch, but will also merge the changes into your local branch



Everything is good, right?

What if your partner or coworker edits a file overnight, but you forgot to pull first? You've added hundreds of lines of code and are ready to commit, but wait…



How to Deal with Merging Conflict?

Merge conflicts can happen when you're pulling a repository that you've been editing, trying to merge two branches, or when you've edited a file and try to push without pulling…

Git attempts to merge the two files automatically, and will if there are no conflicts…

Sh create_merge_conflict.sh

When there are, use **git mergetool** or edit the files yourself

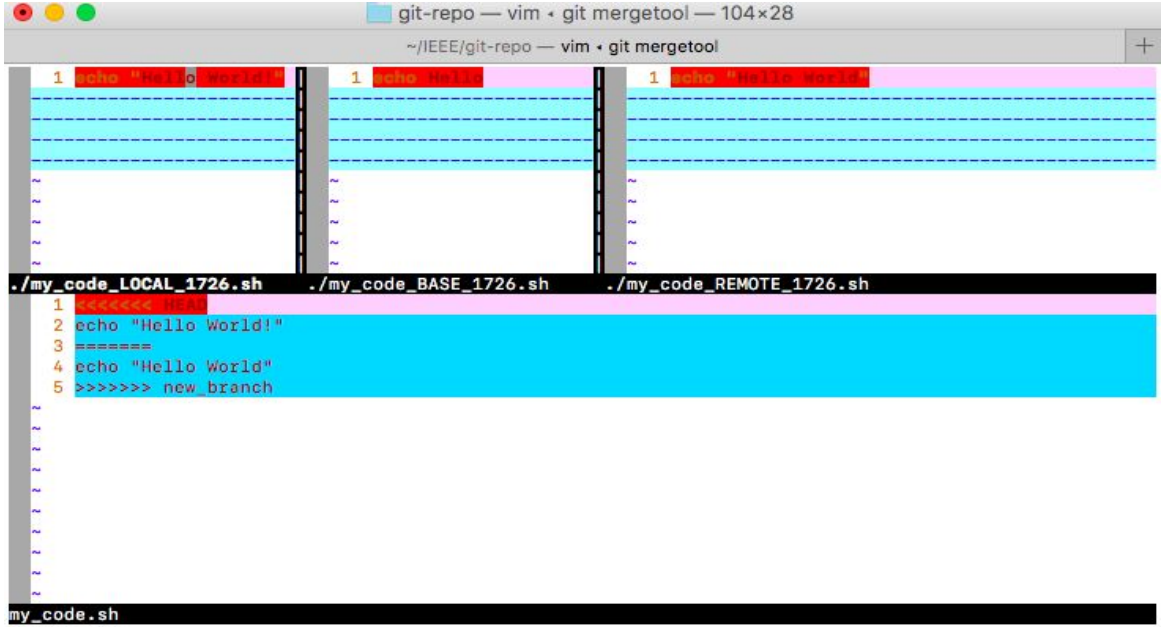
```
Taylors-MacBook-Pro:IEEE TaylorSkilling$ git mergetool
merge tool candidates: opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse ecmerge p4merge araxis bc3
 emerge vimdiff
```

Git Merge Tool

cd git-repo *(Move to the directory where there is conflict)*

git config merge.tool vimdiff *(Select the preferred tool)*

git mergetool *(run)*



git commit --all

(Commit the changes)

More advanced Stuffs?

Git Documentation:

https://git-scm.com/docs

Topics:

- Rebasing
- Merging

Project Examples:

https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project

Questions?

Thank You!

Backup Slides

Using GitHub Desktop

DOWNLOAD GITHUB DESKTOP AT https://desktop.github.com/

CREATE AN ACCOUNT