Terminal + Git Workshop

Rutgers University Student Linux Users Group

Objective

- Introduction
- Terminal
- git

Introduction

- Welcome to RU SLUG (Rutgers University Student Linux Users Group).
- We like Linux/BSD!

Terminal

What is the Terminal

- It's an interface that allows the user to interact with the OS
- The Terminal takes commands given by the user and prints back some results.
- How to get started:
  - If you're on Windows: Use Powershell.
  - If you're on Mac OS X: Use item (or whatever you use).
  - If you're on Linux or BSD, use whatever Terminal you have installed!

Basic commands

- ls
- pwd
- cd
- mkdir
- touch
- cp/mv
- Cool commands that will make you look cool 😎

ls and pwd command

```
afe123x@inspiron-7506 ~/Documents/clubs/ruslug $ ls
gitoverview.png
output.s
part1.png
part2.png
part3.png
part4.png
terminalgit.md
third
third.c
afe123x@inspiron-7506 ~/Documents/clubs/ruslug $ pwd
/home/afe123x/Documents/clubs/ruslug
afe123x@inspiron-7506 ~/Documents/clubs/ruslug $
```

- ls will list all the files in your working directory

cd command

- the "change directory" command.

```
afe123x@inspiron-7506 ~/Documents/clubs $ pwd
/home/afe123x/Documents/clubs
afe123x@inspiron-7506 ~/Documents/clubs $ cd ruslug
afe123x@inspiron-7506 ~/Documents/clubs/ruslug $ pwd
/home/afe123x/Documents/clubs/ruslug
afe123x@inspiron-7506 ~/Documents/clubs/ruslug $
```

Special things

- `..` means parent directory
- `.` means current directory

Example: `cd ..` will change directory to the parent directory.

mkdir

```
mkdir <folder-name>
```

- This will make a new directory named folder-name
- **notes**: Directory and folder can be used interchangeably.

touch and cat

```
touch mike
```

- This will make a file named mike

```
cat mike
```

- This command will print out the contents of mike.

cp/mv Commands

```
cp <src> <dest>
```

- This will copy a file, src, to a destination, dest.

```
mv <src> <dest>
```

- This will move a file, src, to its destination, dest
  - We can also rename files with mv.

Removing Files

```
rm <file>
```

- This removes **file**.
- **warning**: This will permanently delete the file.

Things you should know

- Redirection
- Piping

Redirection

- Instead of seeing your output in the Terminal, you can **redirect** it into a file.
- Instead of typing the input to a program, you can **redirect** a file into the program.

```
ls > output.txt # Instead of looking at the output in the terminal
cat < output.s # instead of typing to standard input.
ls 2> output.txt #will print error messages to output.txt

ls >> output.txt # Will append the output of ls to the output.txt file
```

Piping

We can take the stdout of a program and use it as the stdin of another program.

Ex:

```
cat output.s | grep mov
```

- This will take the output of `cat output.s`, and use it as input for `grep mov`.

What is Version Control?

- Your Opinions?

Version Control System

- A system that tracks changes made to files over time
  - Allows for Collaboration

Types of Version Control

- Centralized Version Control
- Distributed Version Control

Git

- A Version Control System created by **Linus Torvalds**
  - He needed to somehow track changes made to the Linux Kernel
- It's a Distributed Version Control System.

Github Desktop

- What about Github Desktop???

Github Desktop

What the fuck did you just fucking say about me, you little bitch? I'll have you know I graduated top of my class in the Navy Seals, and I've been involved in numerous secret raids on Al-Quaeda, and I have over 300 confirmed kills. I am trained in gorilla warfare and I'm the top sniper in the entire US armed forces. You are nothing to me but just another target. I will wipe you the fuck out with precision the likes of which has never been seen before on this Earth, mark my fucking words. You think you can get away with saying that shit to me over the Internet? Think again, fucker. As we speak I am contacting my secret network of spies across the USA and your IP is being traced right now so you better prepare for the storm, maggot. The storm

Github Desktop

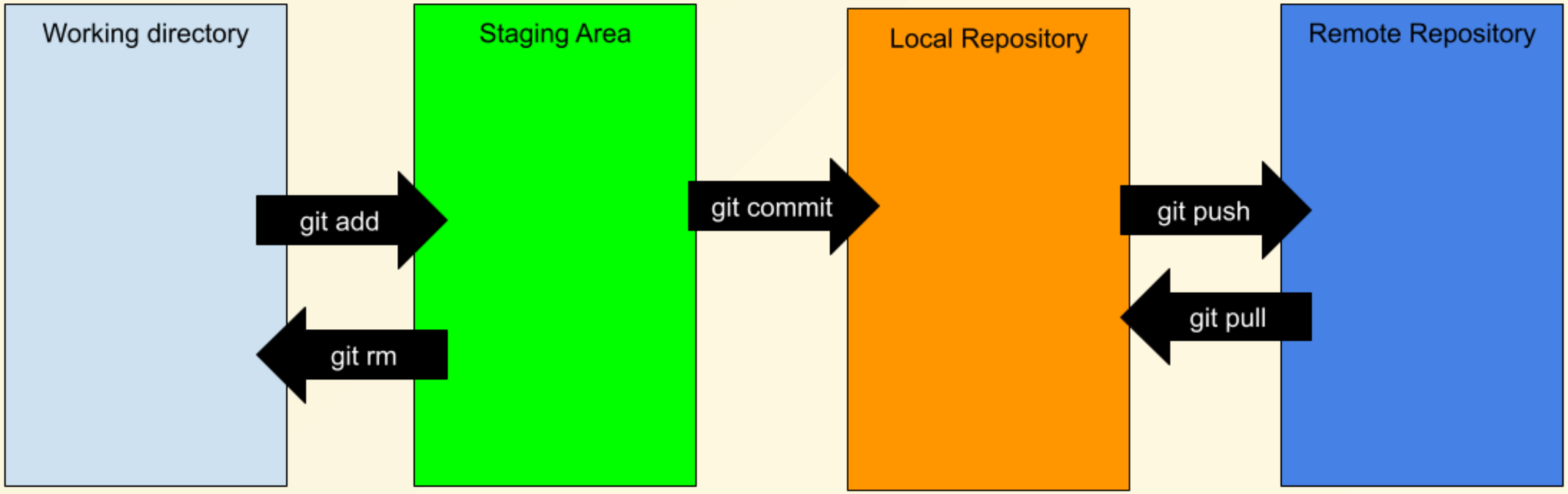
miserable ass off the face of the continent, you little shit. If only you could have known what unholy retribution your little "clever" comment was about to bring down upon you, maybe you would have held your fucking tongue. But you couldn't, you didn't, and now you're paying the price, you goddamn idiot. I will shit fury all over you and you will drown in it. You're fucking dead, kiddo.

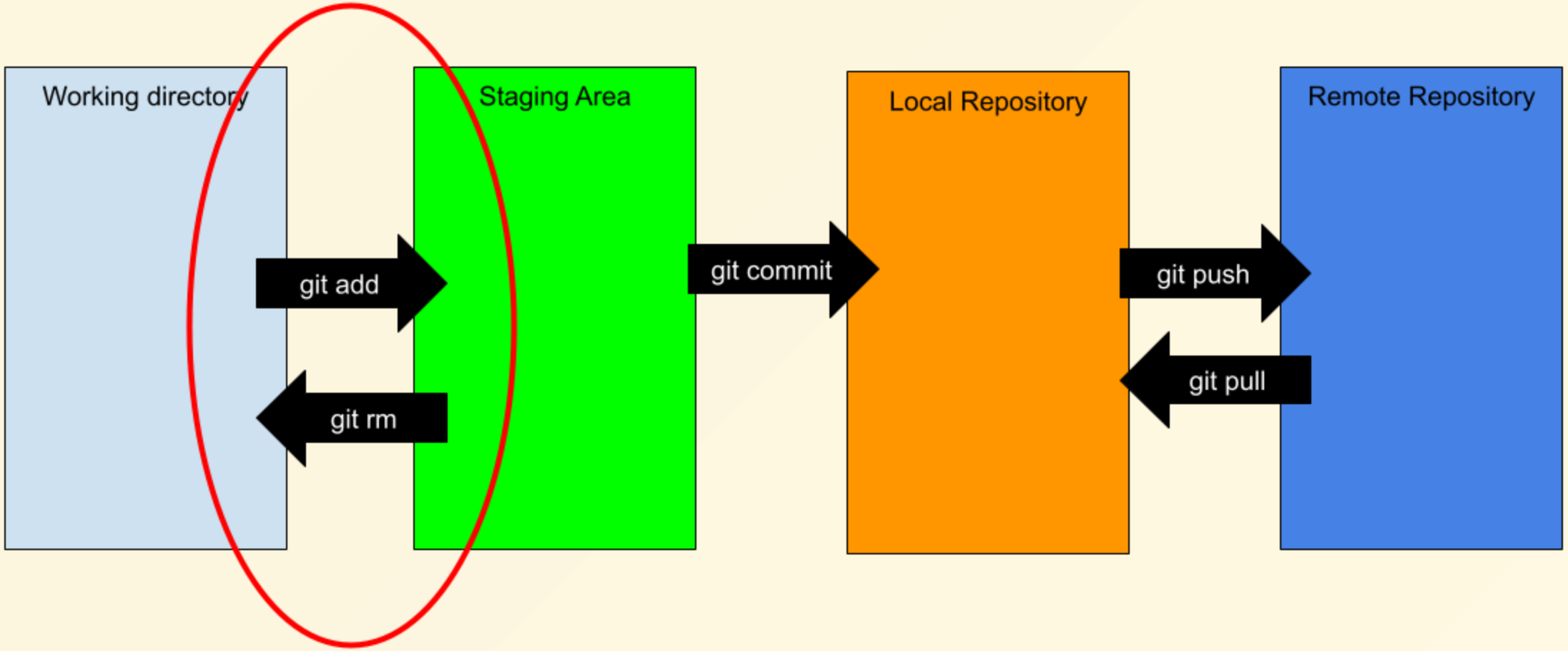
Initializing a git repository

```
git init
```

- This will initialize a git folder in your local directory.
  - You'll have many new folders: .git, .gitignore, etc.

Overview


Working directory
Staging Area
Local Repository
Remote Repository
git add
git rm
git commit
git push
git pull


Working directory
Staging Area
Local Repository
Remote Repository
git add
git rm
git commit
git push
git pull

Viewing Changes made to unstaged changes

```
git diff
git diff <commit-hash-1> <commit-hash-2>
```

Adding Changes to the Staging Area

```
git add <file-name>
```

- This will add changes made to file-name to the staging area.

```
git add .
```

- This will add the changes made to all files in the current directory to the staging area.

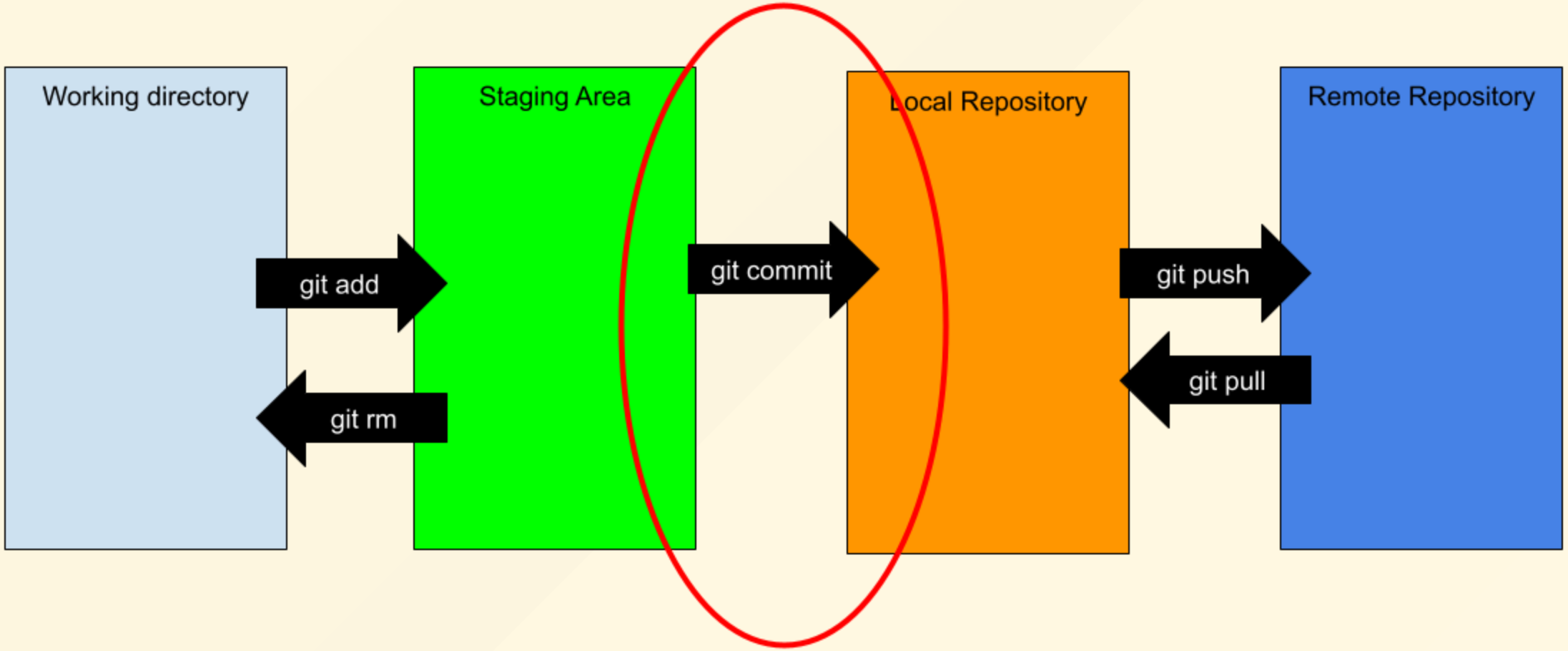
Removing Changes from the Staging Area

```
git reset <file-name> # . if you want to move the changes from the staging area
```

- This is one way to do it (Better way imo).

```
git rm --cached <file-name>
```

- Another way to do it.
  - You should have the `--cached` flag, otherwise it's over.

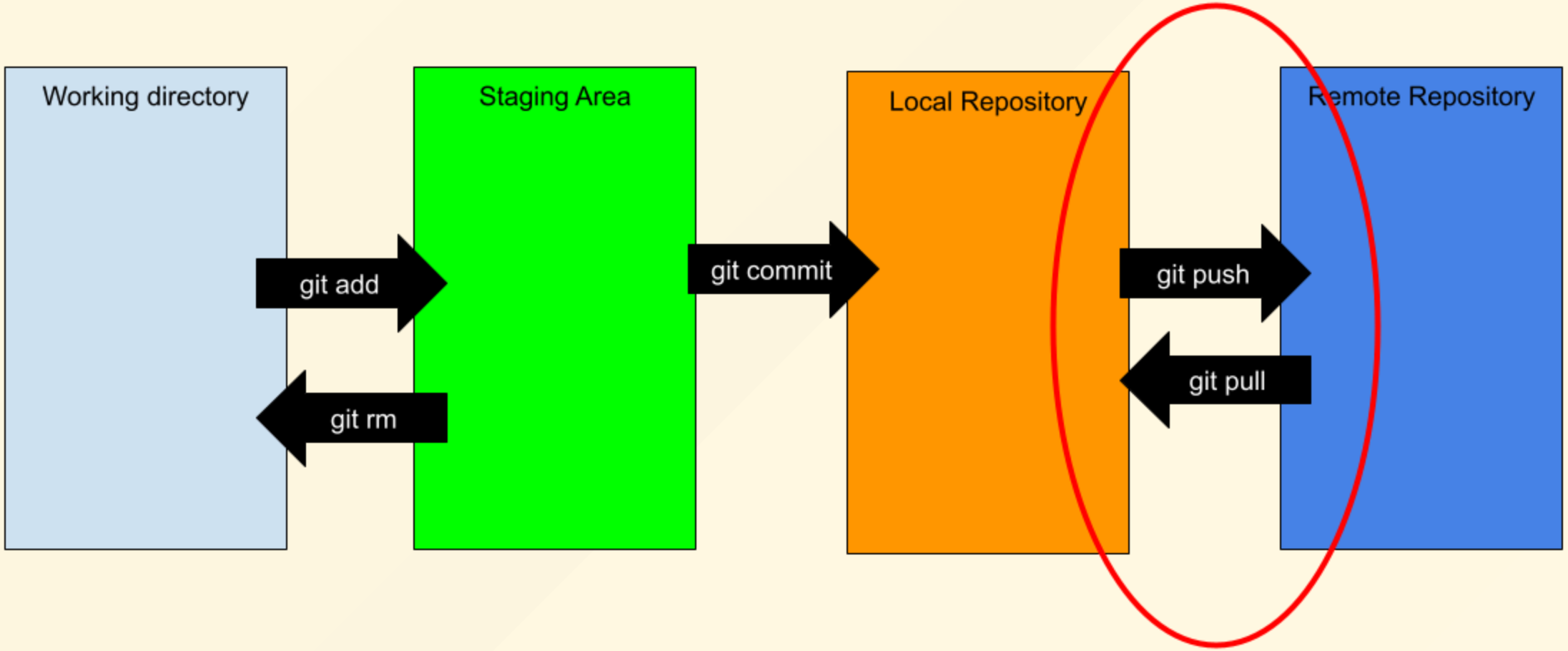

Working directory
Staging Area
Local Repository
Remote Repository
git add
git rm
git commit
git push
git pull

Making a Commit

- Here, we move the changes from the staging area to the local repository.

```
git commit -m "commit message"
```

- You can add a commit message to specify what changes (or just saying oh franny) you made.


Working directory
Staging Area
Local Repository
Remote Repository
git add
git rm
git commit
git push
git pull

Putting the Changes on the Remote Repository

- Here, we will take the changes committed to the local repository and store them in the remote repository.

```
git push -u origin <branch-name>
```

- This will store the changes to the remote repository.
  - if we specify `... origin main`, We'll push changes to the main branch.
- We'll talk about branching later!

Pulling Changes

- Pulling will bring the changes from the remote repository to your local repository

```
git pull
```

Congrats

- You know the Basics of git. Thanks for coming!!!

Congrats

- You know the Basics of git. Thanks for coming!!! SIKE!!!

Next Topics

- .gitignore
- Branching

gitignore

- Let's say you have the following program:

```
fn main(){
  println!("I' never touching C again!");
}
```

- You compile the code, and commit the binary.

- You realize you made a typo, and make the revision

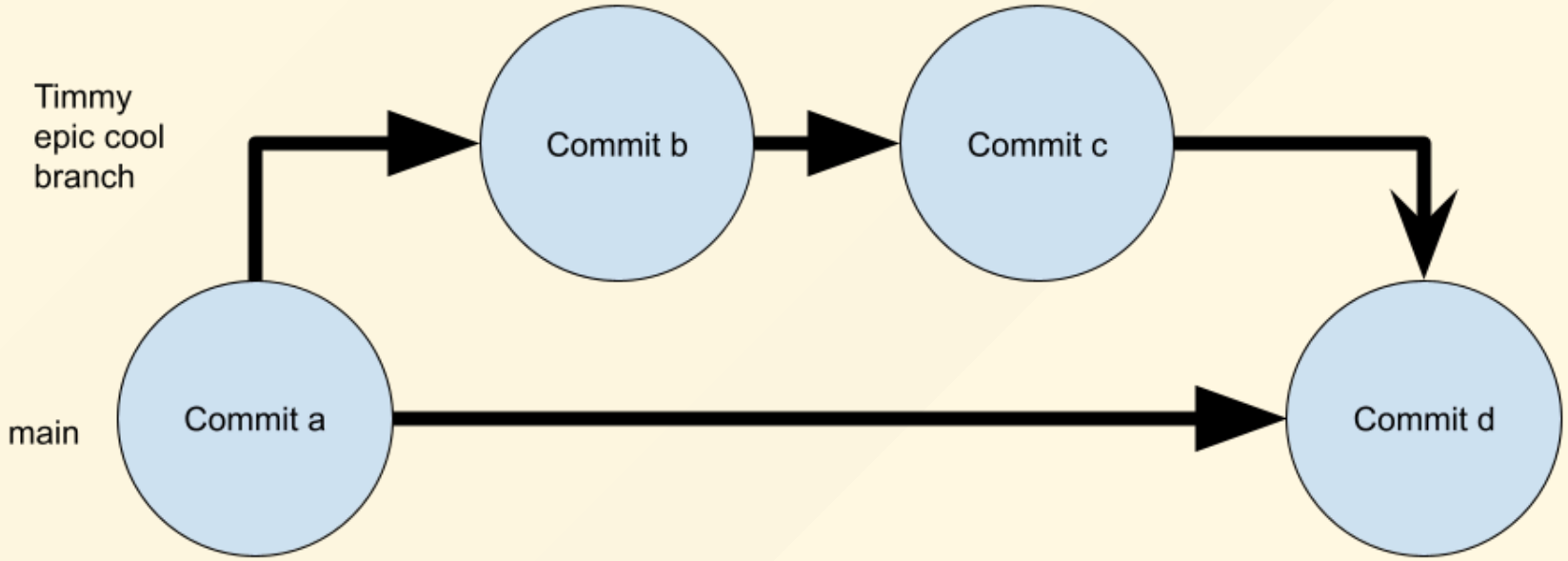
```
fn main(){
  println!("I'm never touching C again!");
}
```

- When you do git status, it will ask you to commit not only the code but also the binary. This can be annoying and unnecessary. you can use a `.gitignore` file for this.

Example gitignore

```
*.png
*.exe
*.o
```

Branching



- Branching means you diverge from the main line of development and continue to do work without messing with that main line.

Creating a new branch

```
git branch <branch-name>
git branch ohfranny
```

Switching Branches

```
git checkout <branch-name>
git checkout ohfranny # for example
```

Pushing Branch

```
git push -u origin <branch-name>
git push -u origin ohfranny
```

Best Practices with Branches

- When you're collaborating on code:
  - Make your own branch
  - Do your work and push on the branch
  - Once you know that your code works as intended, merge it with the main branch.
- Not great practice to always push to the main branch!

Fixing Mistakes

Everyone Makes mistakes

Undoing Changes to unstaged file

```
git checkout <file-name>
```

Changing the latest commit message

```
git commit --ammend -m "new message"
```

Moving commit to another branch

- Let's say we've been committing to the incorrect branch.

```
git checkout <correct-branch>
git cherry-pick <hash>
```

Going back to a Previous Commit

- Let's say your code is awful, and you need to go back to a previous commit.
  - You'd use git reset!
- There are three types:
  - soft
  - mixed (default)
  - hard

Soft

```
git reset --soft <hash>
```

- We'll go back to the commit, but keep the changes to the staging directory.

Mixed

```
git reset <hash>
```

- Here, the changes are in the working directory

Hard

```
git reset --hard <hash>
```

- Resets all the track files to match the commit hash specified

Removing files not being tracked

```
git clean -df
```

- This will clean your directories and folders.

Undoing git changes

```
git reflog
git checkout <reflog-hash>
```

stash

```
git stash save "stash message."
git stash list
git stash apply <stash-number> # Will still be on list
git stash pop # Will get rid of stash on the list and apply it.
git stash drop <stash-number> # will remove from stash
```

Yipee!!!

You know how to use git; you're officially cool

Github

- Are git and github the same?

Github

- Are git and github the same? NO!!! OOHHOHOHOHO!!!
- Github hosts git repositories.
- git handles local repositories, and can connect to remote repositories

Connecting git and github

- We can use ghcli, but NAH!!!!
- We'll use the good ol' technique of ssh

Make a Github Account

- Raise your hand if you already have a github account!

Setting up Global Variables

```
$ git config --global user.name "Your name here"
$ git config --global user.email "your_email@example.com"
```

Generating SSH key

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Copying public key

```
cat ~/.ssh/id_rsa.pub | wl-copy
```

- If you don't have wl-copy, or any copying utility, just copy it manually.

Paste it into Github

- Go to your github Account Settings
- Click “SSH Keys” on the left.
- Click “Add SSH Key” on the right.
- Add a label (like “My laptop”) and paste the public key into the big text box.

Testing

```
ssh -T git@github.com
```

- If successful, you should get a message like this:

```
Hi AFE123x! You've successfully authenticated, but GitHub does not provide shell access.
```

Your Turn!!!