

PRECISION HEALTH  
ANALYSIS BOOTCAMP

# Git(Hub)

Amrit Singh, PhD

Department of Anesthesiology, Pharmacology and Therapeutics, UBC

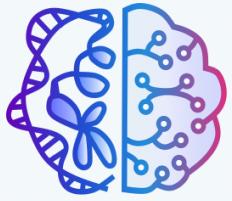
Centre for Heart Lung Innovation

July 26, 2022 | 12:00-14:00

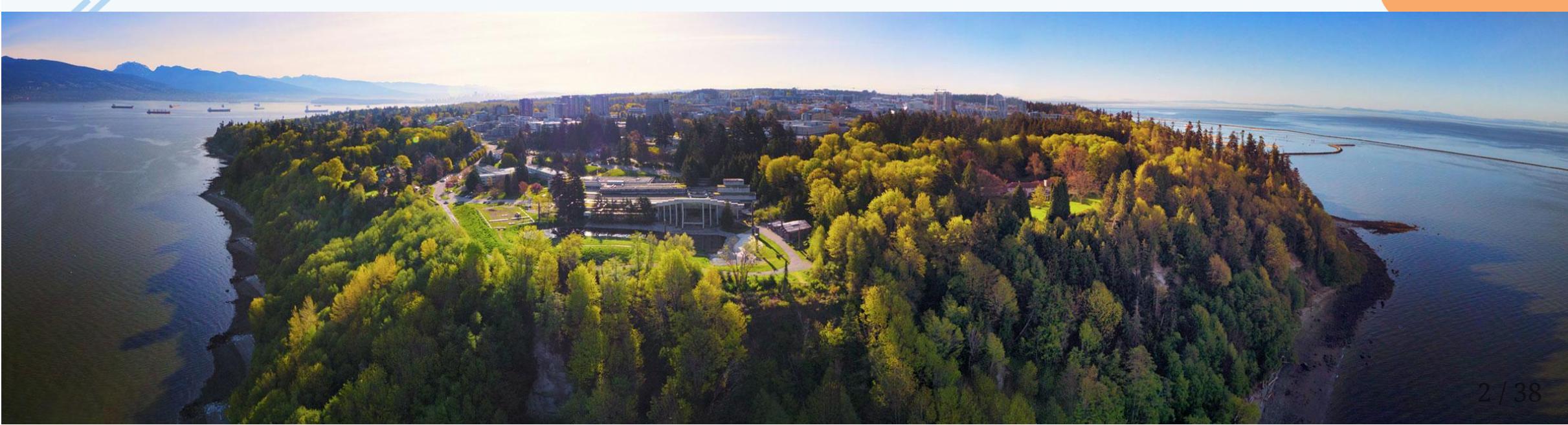
 lab (I am hiring!)

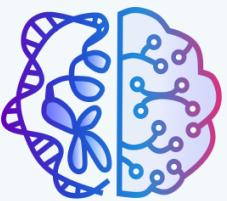
 workshop material

 asingh\_22g



We would like to begin by acknowledging that the land on which we gather is the traditional, ancestral, and unceded territory of the xwməθkwəy̓əm (Musqueam) People.





# Copyright Information



## Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

This is a human-readable summary of (and not a substitute for) the license. [Disclaimer](#).

### You are free to:

**Share** — copy and redistribute the material in any medium or format

**Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.



**Attribution** — You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.



**ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.

**No additional restrictions** — You may not apply legal terms or [technological measures](#) that legally restrict others from doing anything the license permits.

Read more here:  
<https://creativecommons.org/licenses/by-sa/4.0/>



Why are you attending this workshop? What do you hope to learn?

16

...

Sockeye with RStudio

Sharing code with team.

What is github

Rstudio<sup>fast</sup>

# How to find right code

Chinook

# Version control

Github

Pull request

## Learn version control

Protips

Why does git matter? fork Sockeye with GitHub

Best practice for Git

# Learning outcomes

1. Differentiate between Git and GitHub and be able to setup the connection on sockeye
2. Use git to copy open-source code and test the code locally
3. Make changes and update codebase (including undo)
4. Implement the basics of Git(Hub) Flow

# Workshop setup

## log into sockeye

- make sure you are connected to vpn
- [ARC quickstart](#)

```
ssh <cwl>@sockeye.arc.ubc.ca
```

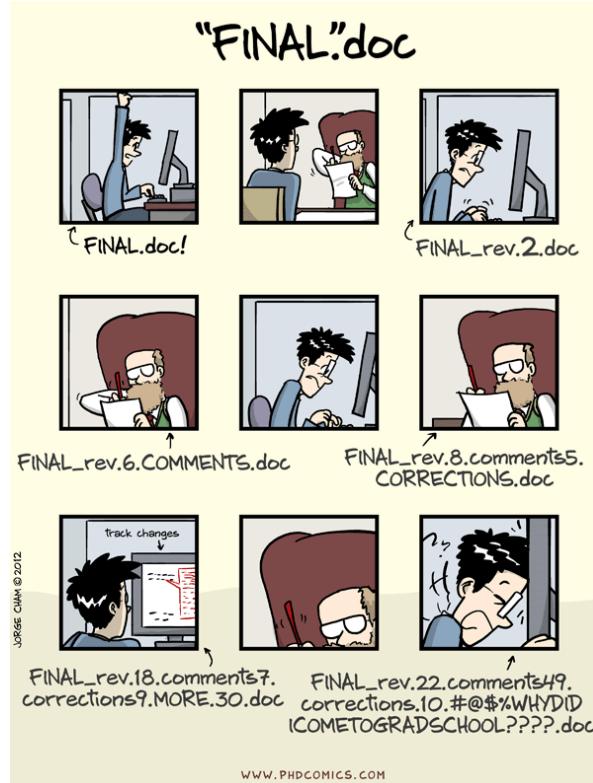
## Make your workspace for this workshop

```
mkdir -p /scratch/tr-precisionhealth-1/Workshops/StudentSpaces/$USER/ && cd "$_"
```

## Copy starter material for this workshop

```
cp -R /project/tr-precisionhealth-1/PrecisionHealthVirtualEnvironment/Workshops/github/ ./
```

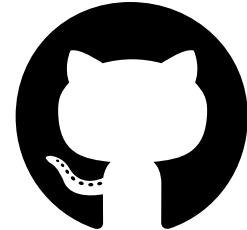
# Problem



version control

# Solution

# git



- distributed version control system

```
module load git  
git --version
```

- Git reference
- Git homepage

GitHub

- use module avail to check which software is installed.

# Example projects

- React
- PyTorch
- GitHub Pages
- Book
- Manuscript

# GitHub features

- README
- issues
- pull requests
- projects
- actions
- wiki
- insights

# People behind Git and GitHub

## Linus Torvalds



**Linus Torvalds**  
torvalds

[Follow](#)

162k followers · 0 following

Overview    Repositories 6

Popular repositories

- linux**  
Linux kernel source tree  
C 135k 43.9k
- test-tlb**  
Stupid memory latency and TLB tester  
C 478 175
- subsurface-for-dirk**  
Forked from subsurface/subsurface  
Do not use - the real upstream is Subsurfac

- main developer of the Linux kernel (released in 1991)
- developed Git to maintain the Linux kernel (2005)

## Microsoft acquires GitHub



- GitHub: Chris Wanstrath and Tom Preston-Werner
- Microsoft: Satya Nadella

# Git configuration

```
git config --global user.name GITHUB_USERNAME  
git config --global user.email GITHUB_EMAIL
```

## list credentials

```
git config --list
```

## git config file location

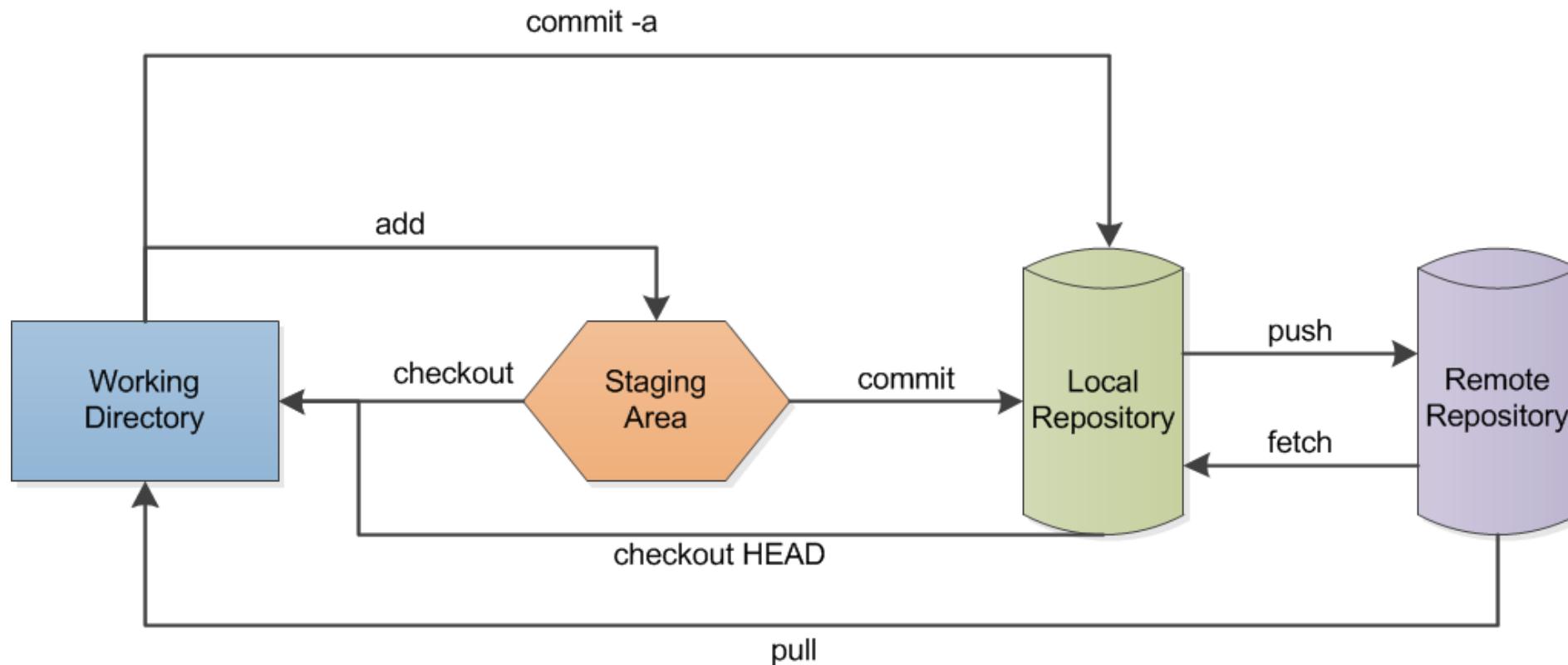
```
git config --show-origin --get credential.helper
```

- manual for every git command

```
git command --help
```

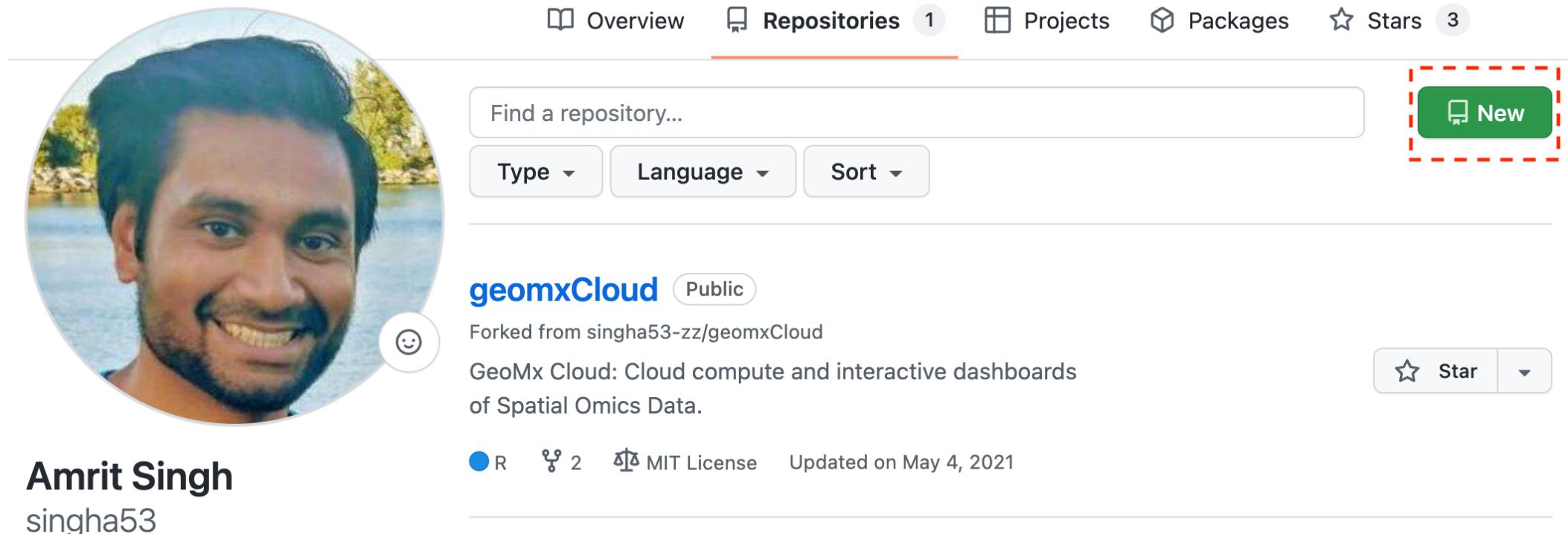
# Git Flow

- Git data flow, Lbhtw, and CC-BY-SA-3.0.



# Create new or add existing repository (repo) to GitHub

Make remote repo called test



The screenshot shows a GitHub user profile for 'Amrit Singh'. On the left is a circular profile picture of a smiling man with dark hair and a beard. Below the picture, the name 'Amrit Singh' and the handle 'singha53' are displayed. To the right of the profile picture is the user's GitHub dashboard. At the top, there are tabs: 'Overview', 'Repositories' (which is highlighted with a red underline), 'Projects', 'Packages', and 'Stars' (with a value of 3). A search bar says 'Find a repository...' and there are dropdown menus for 'Type', 'Language', and 'Sort'. A green button labeled 'New' is highlighted with a red dashed box. Below the dashboard, the user's first repository, 'geomxCloud', is listed. It is described as 'Public' and was 'Forked from singha53-zz/geomxCloud'. The repository description reads: 'GeoMx Cloud: Cloud compute and interactive dashboards of Spatial Omics Data.' At the bottom of the repository card, it shows 'R 2 MIT License Updated on May 4, 2021'.

Amrit Singh  
singha53

Click here to create new repository.

# Create new repo on GitHub

## Make local repo

```
mkdir test ## make test folder
cd test ## move into test folder
echo "# test" >> README.md ## create file
git init ## initialize git repo
git add README.md ## move file to staging area
git commit -m "first commit" ## associate message with change
git branch -M 'main' ## rename master branch to main
git remote add origin https://github.com/GITHUB_USERNAME/test.git ## add location of remote repo
git push -u origin 'main' ## update remote repo
git status
```

Username for '<https://github.com>': GITHUB\_USERNAME  
Password for '[https://GITHUB\\_USERNAME@github.com](https://GITHUB_USERNAME@github.com)': PERSONAL\_ACCESS\_TOKEN

- get PERSONAL\_ACCESS\_TOKEN by going to Profile Pic --> Settings --> Developer Settings on left-side bar --> Personal access tokens --> Generate new token --> Enter name, select repo and hit the Generate token button at the bottom of the page
- Note: Replace remote origin using: <git remote set-url origin git://new.url.here>
- [Try Problem Set 0](#)

# Add existing repo to GitHub

Save credentials for future use

```
git config --global credential.helper cache  
git config --unset credential.helper ## unset credentials
```

## Add local repo

- delete old repo on Github and locally
- create empty repo on Github named `reproducible_analysis_demo`
- add a local repo and update remote repo

```
cd reproducible_analysis_demo  
git init  
git add .  
git commit -m "project setup"  
git remote add origin https://github.com/GITHUB_USERNAME/reproducible_analysis_demo.git  
git branch -M 'main'  
git push -u origin 'main'  
git status
```

- Try Problem Set 1

# Copy remote repo

## navigate to working directory

```
cd /scratch/tr-precisionhealth-1/Workshops/StudentSpaces/$USER/github
```

- use `pwd` to check current location

## clone remote repository

```
git clone https://github.com/GITHUB_USERNAME/REPO_NAME.git
```

- Try Problem Set 2a: clone public repo
- Try Problem Set 2b: clone private repo

# Master --> Main

In June 2020, GitHub announced that it was moving the default branch name from master to the more neutral name, main. GitLab followed suit in a few months later. Tobie Langel makes the salient point on why changing the name is a good thing:



Tobie Langel  
@tobie

Replying to @DEGoodmanWilson

Turns out I was wrong. The master terminology in Git is historically tied to the master/slave metaphor and not the master copy one (which makes it both racist and a dumb metaphor 🤦 ): [mail.gnome.org/archives/desktop/](http://mail.gnome.org/archives/desktop/)...

10:42 AM · Jun 9, 2020 · Twitter Web App

221 Retweets 109 Quote Tweets 593 Likes

# Pulse check



How is the pace of this workshop?

17

...

too slow



6%

slow



12%

too fast



6%

fast



41%

just right



35%

# Scenario 1: Simplest project workflow

## 1) make Environment file

create a .env file

- make sure you are in  
reproducible\_analysis\_demo/

```
vi .env
```

- hit "i" on keyboard then add the following to  
.env

```
PASSWORD=top_secret_code
```

- hit "Esc" then type ":wq" (write quit) then hit  
Enter on keyboard

## 2) ignore large files

add .env to .gitignore

```
ls -la  
git status  
vi .gitignore
```

- hit "i" on keyboard then add the following to  
.gitignore

```
.env
```

- hit "Esc" then type ":wq" (write quit) then hit  
Enter on keyboard

[GitHub size limits](#)

# Scenario 1: Simplest project workflow

## 3) save local repo and update remote repo

save changes locally

```
git status  
git add .  
git commit -m "add env vars and ignore file"
```

- how to write a good commit message

save changes to remote

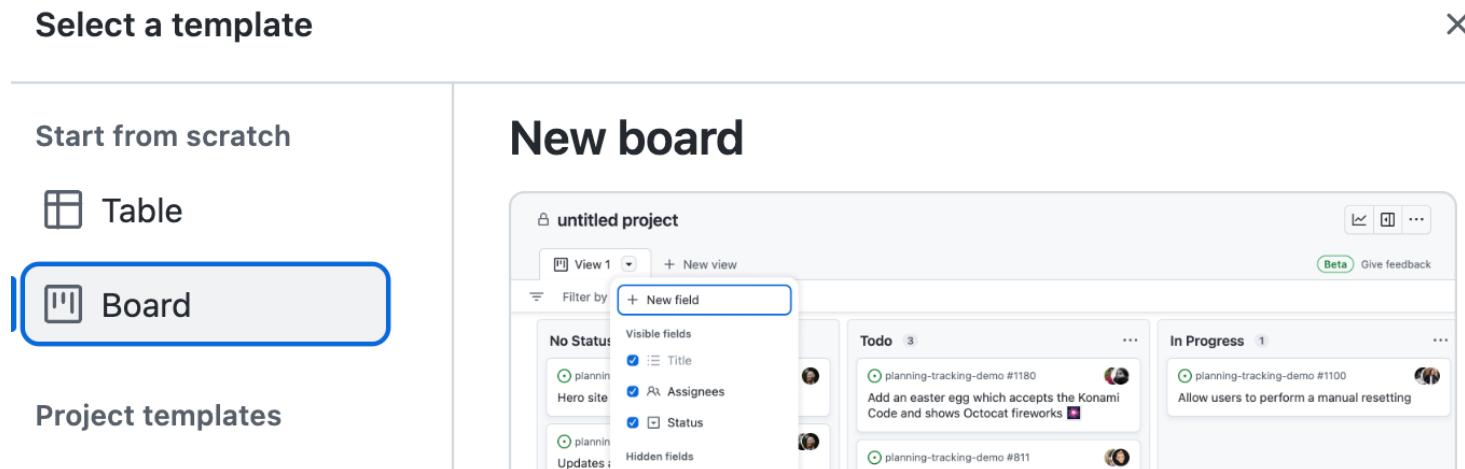
```
git push origin main
```

- undo a commit
- Try Problem Set 3

# Scenario 2: Feature development (self/team)

## 1) use GitHub to manage project

- make a new issue
  - subject: update README
  - description: change GITHUB\_USERNAME and GITHUB\_ACCOUNT\_NAME
- make a new project board



# update README #1

[Edit](#)[New issue](#)[Open](#)

singha53 opened this issue 43 minutes ago · 0 comments



singha53 commented

43 minutes ago · edited

[Owner](#)[...](#)

change GITHUB\_USERNAME and  
GITHUB\_ACCOUNT\_NAME

Assignees



singha53



Labels

documentation



reproducible\_analysis\_demo

[Beta](#)[View 1](#)[+ New view](#)[Filter by keyword or by field](#)

Todo 0

In Progress 1

Done 0

reproducible\_analysis\_demo #1

update README

## 2) Create a new branch to work on an issue

```
git branch  
git checkout -b readme ## git checkout -b BRANCH_NAME  
git branch
```

- Try Problem Set 4

## 2) Move between branches

```
git checkout main  
git checkout readme
```

## 3) see log of changes

```
git log
```

# make changes to repo

```
vi README.md
```

- hit "i" on keyboard and change GITHUB\_USERNAME and GITHUB\_ACCOUNT\_NAME
- hit "Esc"
- type ":wq" (write quit) then hit Enter on keyboard

## check which changes were made

```
git diff  
git status
```

# Code review prior to merging into main repo (Pull Request)

update local and remote repo

```
git add .
git commit -m "update README"
git push origin readme
```

- Try Problem Set 5

The screenshot shows a GitHub repository page. At the top, there's a light gray header with the repository name 'singha53/reproducible\_analysis\_demo' in blue, indicating it's public. Below the header, there are navigation tabs: 'Code' (which is underlined in orange), 'Issues' (with a count of 1), 'Pull requests', 'Actions', 'Projects', 'Wiki', and a shield icon. A yellow banner at the bottom left says 'readme had recent pushes 1 minute ago'. On the right side of the banner is a green button labeled 'Compare & pull request'.

# Pull Request (PR)

- write a PR
- reference issue

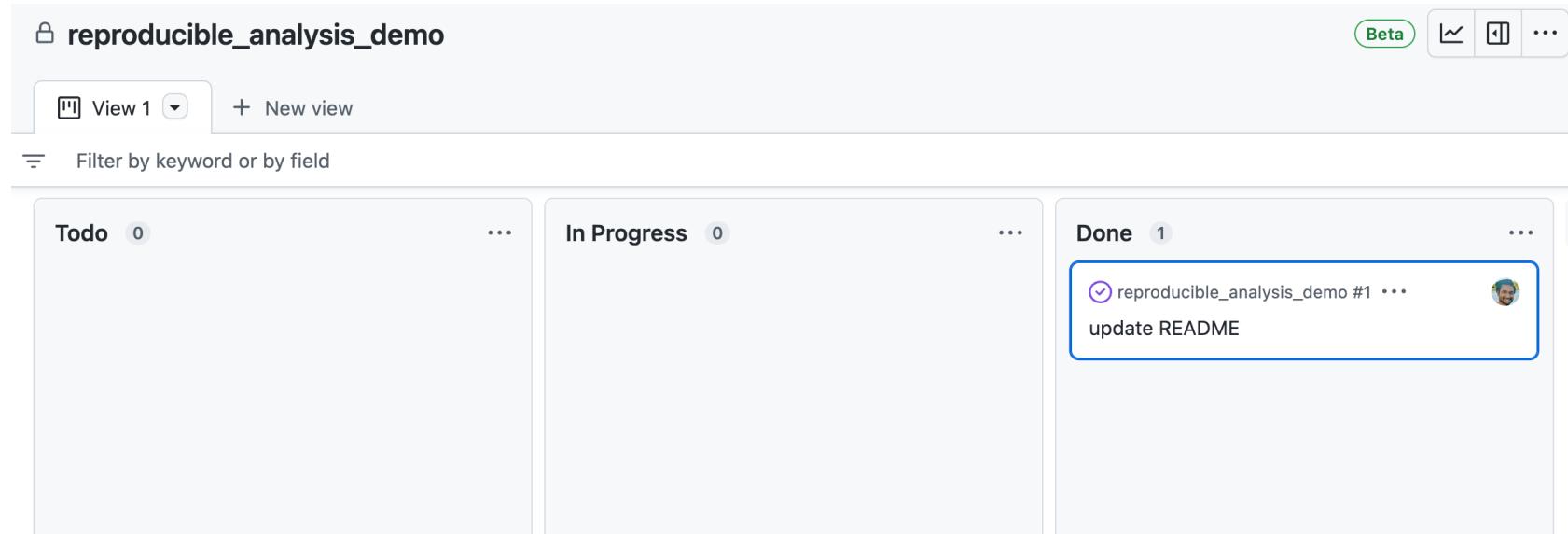
## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, there are dropdown menus for 'base: main' and 'compare: readme'. A green checkmark indicates that the branches are 'Able to merge'. Below this, a commit message 'update README' is entered. There are 'Write' and 'Preview' buttons, along with rich text editing tools (bold, italic, etc.). The commit message is preceded by '#1'. To the right, there are sections for 'Reviewers' (which is empty) and 'Assignees' (which lists 'singha53').

# Pull Request (PR)

- merge PR
- close issue and move to done



- GitHub Flow
- Try Problem Set 6

# Reflect remote changes locally

- move to main branch

```
git checkout main
```

- pull changes from remote branch

```
git pull origin main
```

- delete feature branch

```
git branch -d readme
```

# Resources

## Slides

- xaringan

## Git(Hub)

- Get Git!
- Let's Git started
- .gitignore
- GitHub Actions
- Git cheatsheet

# Problem Set

PS0) Turn a folder into a git repo (initialize) and then uninitialize. [Solution](#)

PS1) Create a Github repo named 'reproducible\_analysis\_demo' with README.md, .gitignore and MIT [License](#). Create a local repo named 'reproducible\_analysis\_demo'. Push changes to github. [Solution](#)

PS2a) Clone this public repo [repo](#) and play hangman. [Solution](#)

PS2b) create private repo on github and clone to local machine. [Solution](#)

PS3) For a given up-to-date github repo do the following: [Solution](#)

- commit a file then undo commit
- commit and file then push to remote then revert to previous commit

PS4) Branching: [Solution](#)

- create new branch but stay on existing branch
- make a new branch and move to new branch

PS5) To prevent modifying the main branch directly, add protection. [Solution](#)

PS6) Revert a pull request. [Solution](#)

# Solutions

Solution to PSo: Turn a folder into a git repo (initialize) and then uninitialized.

initialize git repo

```
mkdir test  
cd test  
git init  
git status  
ls -la
```

uninitialize git repo

```
rm -rf .git  
git status
```

**Solution to PS1: Create a Github repo named 'reproducible\_analysis\_demo' with README.md, .gitignore and MIT License. Create a local repo named 'reproducible\_analysis\_demo'. Push changes to github.**

```
git pull origin main --allow-unrelated-histories
```

- fix merge conflicts in README.md
- open README.md (vi README.md) and remove comments (hit dd on keyboard)

```
<<<<< HEAD
```

```
>>>>> committag
```

- hit ':wq' to write and quit
- commit changes and push to remote

```
git add .
git commit "fix merge conflicts"
git push origin main
```

# Solution to PS2a: Clone this public repo **repo** and play hangman

## Clone repo

```
git clone https://github.com/salifm/cli-games.git  
cd cli-games  
cd Hangman
```

## play game

```
module spider node  
module load node-js  
node hangman.js
```

# Solution to PS2b: Clone private repo

- create repo with README on github
- clone repo

## Method 1

```
git clone https://github.com/GITHUB_USERNAME/REPO_NAME.git
```

Username for '<https://github.com>': GITHUB\_USERNAME  
Password for '[https://GITHUB\\_USERNAME@github.com](https://GITHUB_USERNAME@github.com)': PERSONAL\_ACCESS\_TOKEN

- get PERSONAL\_ACCESS\_TOKEN by going to Profile Pic --> Settings --> Developer Settings on left-side bar --> Personal access tokens --> Generate new token --> Enter name, select repo and hit the Generate token button at the bottom of the page

## Method 2

```
git clone https://GITHUB_USERNAME:PERSONAL_ACCESS_TOKEN@github.com/GITHUB_USERNAME/REPO_NAME.git
```

# Solution to PS3: Undo commit

## commit a file then undo commit

```
mkdir test
cd test
git init
touch README.md
git add README.md
git commit -m "initialize repo"
touch anotherfile.txt
git add anotherfile.txt
git commit -m "add another file"
git log
```

### a) revert to previous commit and remove changes

```
git reset --hard HEAD~1
```

### b) revert to previous commit and retain changes

```
git add anotherfile.txt
git commit -m "add another file"
git log
git reset --soft HEAD~1
```

# Solution to PS4: Branching

create new branch but stay on existing branch

```
git branch BRANCH_NAME
```

make a new branch and move to new branch

```
git checkout -b BRANCH_NAME
```

# Solution to PS5: Protect master branch

- Settings --> Braches (under Code and automation) --> Add branch protection rule
- Branch name pattern: main
- select **Require a pull request before merging**
- caveat: PRs makers can approve their own PRs

# Solution to PS6: Revert Pull request

- click on revert button which create another PR to undo changes

## update README #4

Merged singha53 merged 1 commit into main from readme 8 minutes ago

Conversation 0 Commits 1 Checks 0 Files changed 1

singha53 commented 10 minutes ago

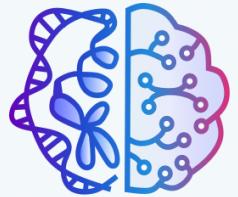
#1

update README e63172e

singha53 self-assigned this 10 minutes ago

singha53 merged commit 095168f into main 8 minutes ago

Revert



PRECISION HEALTH  
ANALYSIS BOOTCAMP

# THANK YOU!

July 26, 2022 | 12:00-14:00

🔗 lab (I am hiring!)  
🔗 workshop material  
🐦 asingh\_22g