

SM-2302 Software for Mathematicians

Git & GitHub: Version control

Dr. Haziq Jamil

Mathematical Sciences, Faculty of Science, UBD

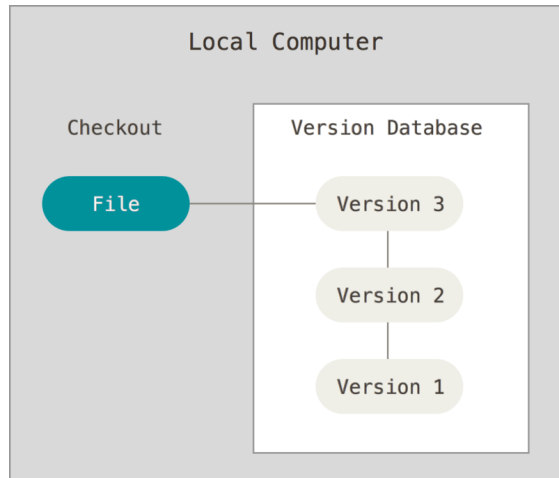
<https://github.com/sm2302-aug24>

Semester I 2024/25

last modified: 2024-07-06

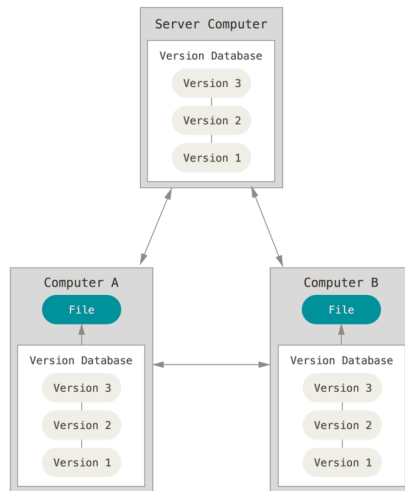
Why version control?

- Simple formal system for tracking changes to a project over time
- Time machine for your projects
 - Track blame and/or praise
 - Remove the fear of breaking things
- Learning curve can be a bit steep, but when you need it you *REALLY* need it



Why Git?

- Distributed
 - Work online or offline
 - Collaborate with large groups
- Popular and Successful
 - Active development
 - Shiny new tools and ecosystems
 - Fast
- Tracks any type of file
- Branching
 - Smarter merges



Verifying git installation

Git should already be installed in the lab PCs. Verify by launching the terminal and typing

```
haziqj@Haziqs-MacBook-Air ~ % git --version  
git version 2.39.3 (Apple Git-146)
```

On your own PCs, you can install git by following the directions in Happy Git and GitHub for the useR.

Git vs GitHub

Central Repository



Typically located on remote server



Exclusively consists of ".git" repository folder



Meant for team to share and exchange data



Local Repo



CENTRAL
REPOSITORY

GitHub



Local Repo



Local Repo



Local Repository



Typically located on local machine

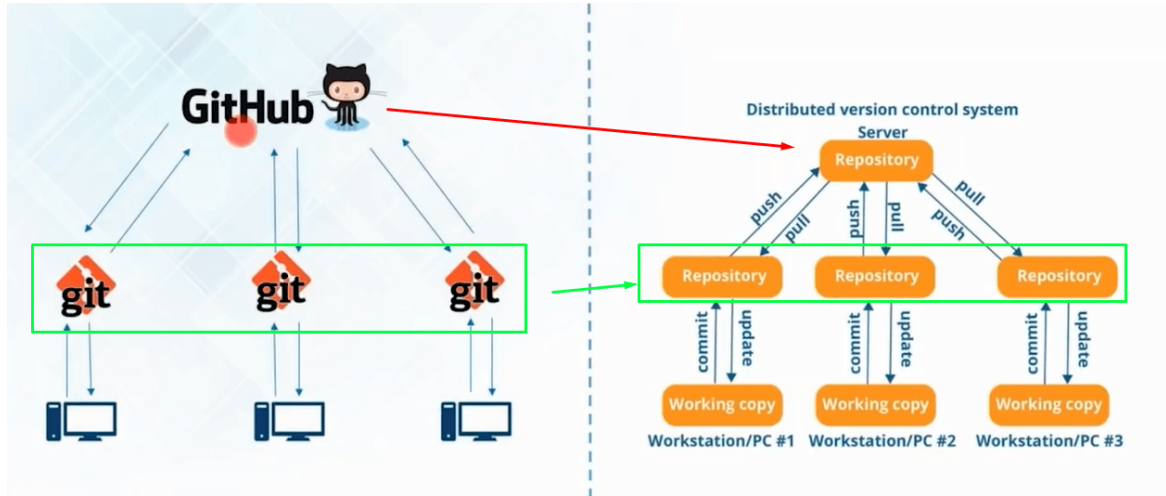


Resides as a ".git" folder inside your project's root

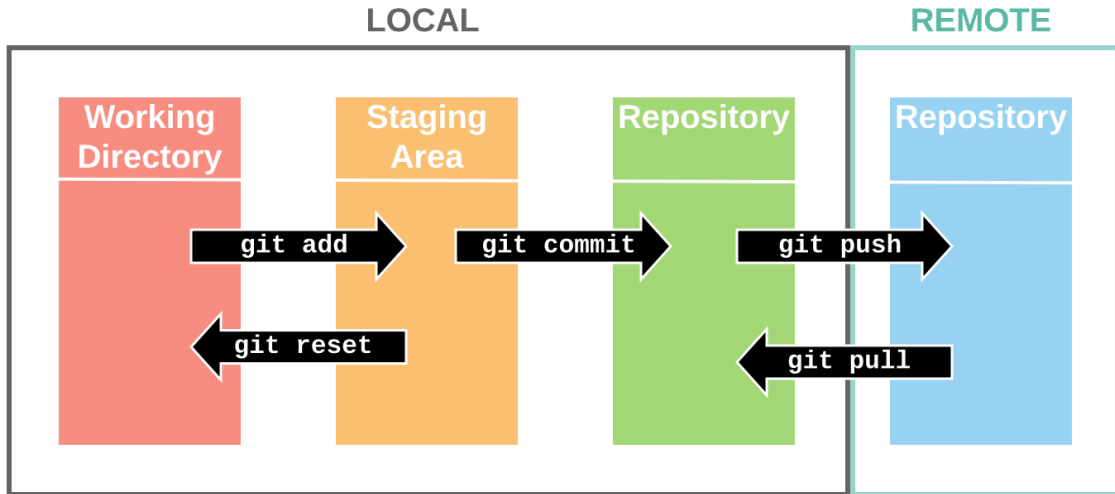


Only admin of the machine can work with this repo.

Git vs GitHub (cont.)



Git in a nutshell



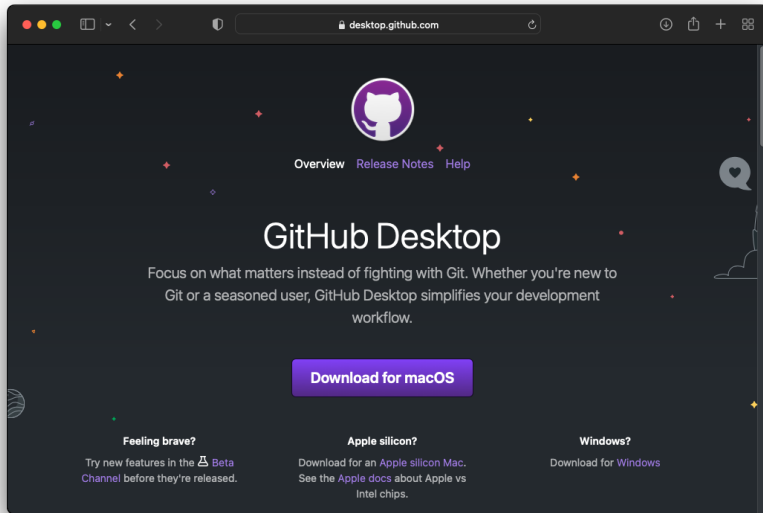
<https://support.nesi.org.nz/hc/en-gb/articles/360001508515-Git-Reference-Sheet>

Quick tutorial

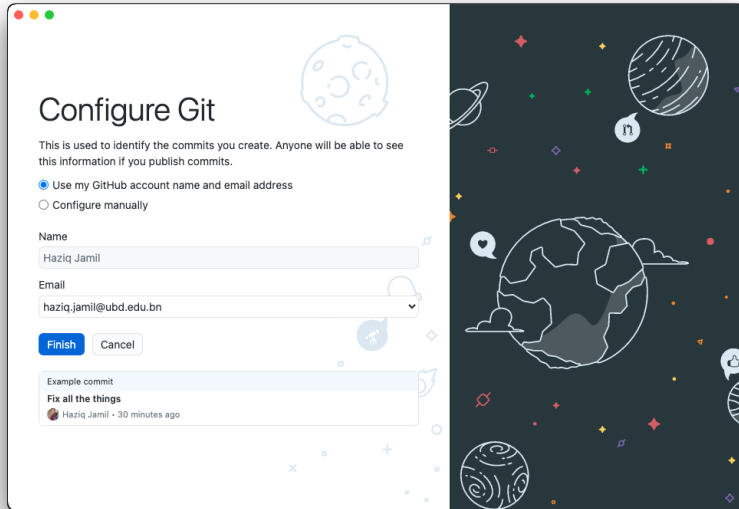
Follow along

`https://docs.github.com/en/get-started/quickstart/hello-world`

Install GitHub Desktop



Configure Git



The image shows a 'Configure Git' dialog box with a space-themed background illustration. The illustration features a large Earth in the center, a moon at the top, and various planets and stars. The dialog box has a white background with a light blue border. It contains the following elements:

- Title:** Configure Git
- Description:** This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.
- Options:**
 - ☒ Use my GitHub account name and email address
 - ☐ Configure manually
- Name:** Haziq Jamil
- Email:** haziq.jamil@ubd.edu.bn
- Buttons:** Finish (blue), Cancel (light blue)
- Example commit:**
 - Fix all the things
 - Haziq Jamil • 30 minutes ago

Git sitrep

```
usethis::git_sitrep()
## Git config (global)
## • Name: <unset>
## • Email: <unset>
## • Global (user-level) gitignore file: <unset>
## • Vaccinated: FALSE
## See `?git_vaccinate` to learn more
## Defaulting to 'https' Git protocol
## • Default Git protocol: 'https'
## • Default initial branch name: <unset>
## GitHub
## • Default GitHub host: 'https://github.com'
## • Personal access token for 'https://github.com': <unset>
## • To create a personal access token, call `create_github_token()`
## • To store a token for current and future use, call `gitcreds::gitcreds_set()`
## Read more in the 'Managing Git(Hub) Credentials' article:
## https://usethis.r-lib.org/articles/articles/git-credentials.html
## Git repo for current project
## No active usethis project
```

Configure Git (cont.)

Remark

Do this only if sitrep still shows wrong info.

The following will tell Git who you are, and other common configuration tasks.

```
usethis::use_git_config(  
  user.name = "Haziq Jamil",  
  user.email = "haziq.jamil@ubd.edu.bn"  
  # push.default = "simple",  
  # pull.rebase = FALSE  
)
```

This can also be done via the terminal with,

```
$ git config --global user.name "Haziq Jamil"  
$ git config --global user.email "haziq.jamil@ubd.edu.bn"  
$ git config --global push.default simple
```

Typical GitHub workflow

I want to start a project that involves some code.

1. Go to GitHub.com and create a new repo.
 - Can initialise accordingly (.gitignore and/or README)
2. Clone to local repo
3. Add code inside
4. Commit and push

Not really going to spend much time branching and creating pull requests

Version control best practices

- Commit early, often, and with complete code.
- Write clear and concise commit summary messages.
- Test code before you commit.
- Use branches.
- Communicate with your team.

Git and GitHub resources

- Git's Pro Git book, Chapters Getting Started and Git Basics will be most useful if you are new to Git and GitHub
- Git cheatsheet by Atlassian
- GitHub's interactive tutorial
- Free online course from Udacity
- Happy Git with R by Jenny Bryan