SI100B Python Final Project Part0 - **Git**

Zixuan Liu

Contents

上海科技大学
ShanghaiTech University

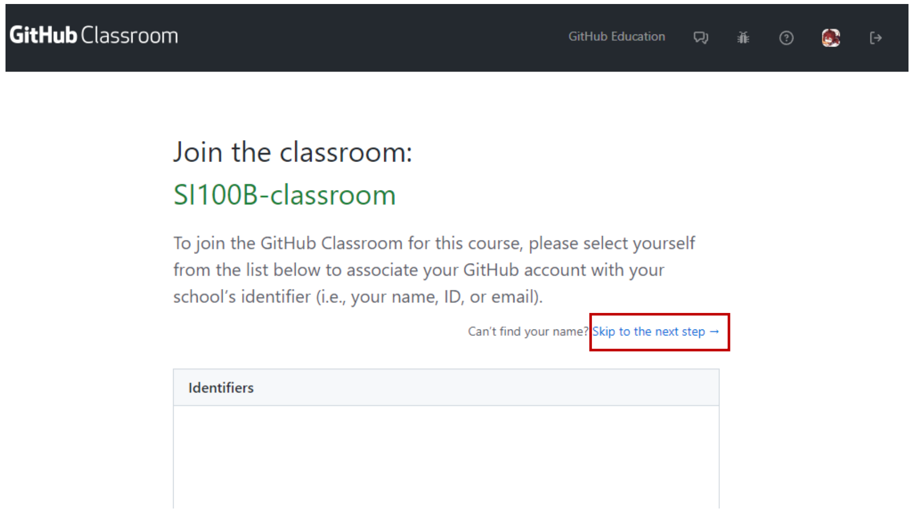
Accessing to GitHub Classroom

Accessing to GitHub Classroom

- A GitHub account
  - https://github.com/
- Sign in GitHub Classroom with your GitHub account
  - https://classroom.github.com/

Accessing to GitHub Classroom

- Paste the invitation URL https://classroom.github.com/a/C3B3odAQ
- Skip to the next step.



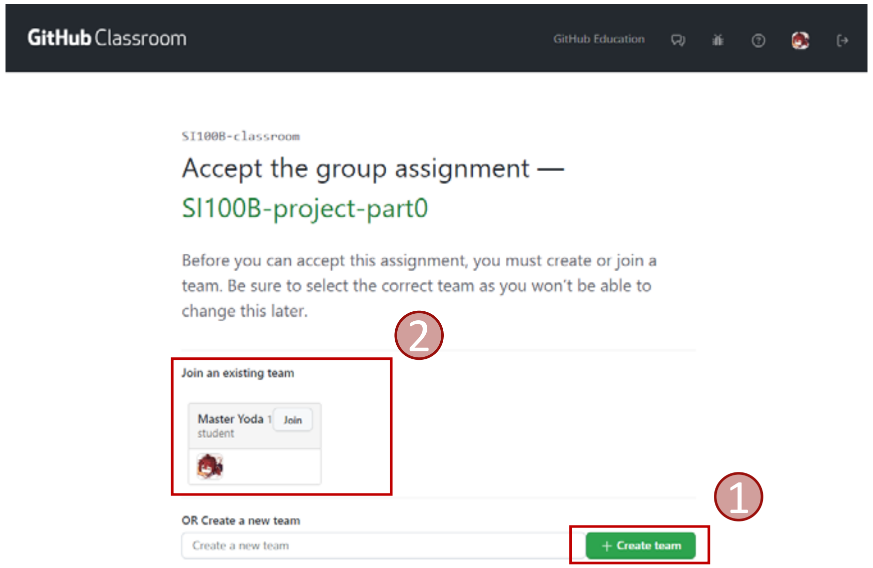
Join the classroom:
SI100B-classroom

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step →

Identifiers

- One in a group creates a team
- The other two team members join.



SI100B-classroom

Accept the group assignment — SI100B-project-part0

Before you can accept this assignment, you must create or join a team. Be sure to select the correct team as you won't be able to change this later.

Join an existing team

Master Yoda 1 student Join

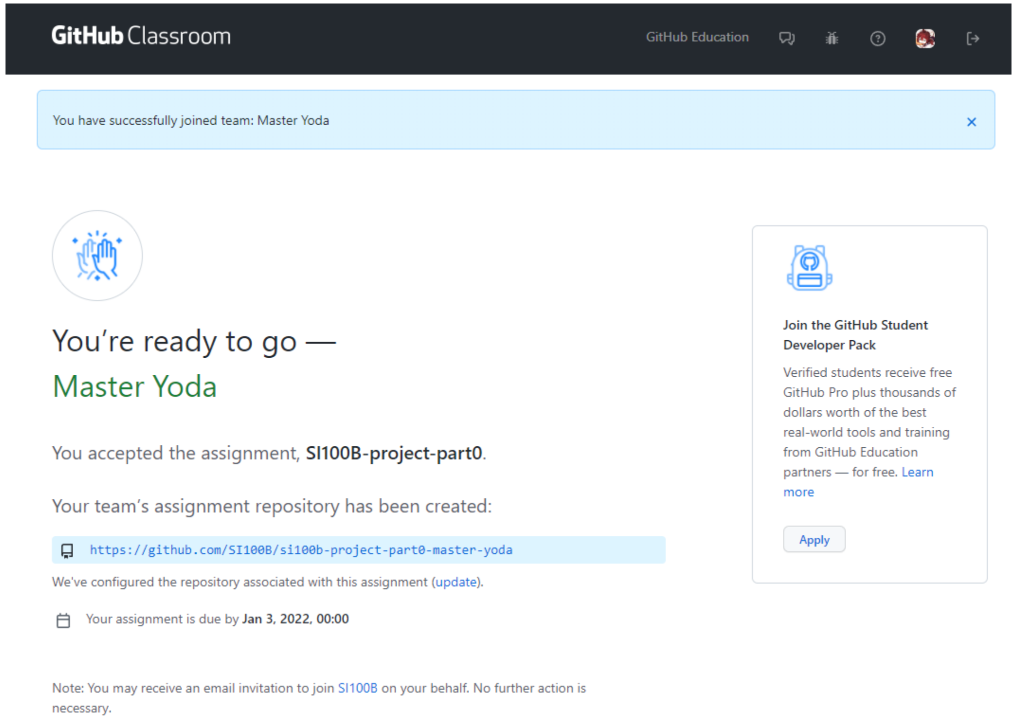
OR Create a new team

Create a new team

+ Create team

Accessing to GitHub Classroom

- Ready to go
- Please fill in this
- https://docs.qq.com/sheet/DT2tQY3pobktMeWli?tab=BB08J2



GitHub Classroom

You have successfully joined team: Master Yoda

You're ready to go — Master Yoda

You accepted the assignment, **SI100B-project-part0**.

Your team's assignment repository has been created:

https://github.com/SI100B/si100b-project-part0-master-yoda

We've configured the repository associated with this assignment (update).

Your assignment is due by **Jan 3, 2022, 00:00**

Note: You may receive an email invitation to join SI100B on your behalf. No further action is necessary.

Join the GitHub Student Developer Pack

Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. Learn more

About Git

Remember: Git is EASY to use!

Hope you would think so after the tutorial. If not, it’s all my fault…

About Git - Short History

In 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool’s free-of-charge status was revoked. This prompted the Linux development community (and in particular Linus Torvalds, the creator of Linux) to develop their own tool based on some of the lessons they learned while using BitKeeper.

林納斯•托瓦茲
Linus Torvalds

林納斯·托瓦茲於2014年LinuxCon Europe大會演講

BITKEEPER

About Git - What is Git?

- **About Version Control**
- **Snapshots, Not Differences**
- **Nearly Every Operation Is Local**
    - -> Fast and powerful
- **Git Has Integrity**
    - Everything in Git is checksummed before it is stored and is then referred to by that checksum.
- **Git Generally Only Adds Data**
    - It is hard to get the system to do anything that is not undoable or to make it erase data in any way.
- **The Three States (talk later)**
    - You modify files in your working tree.
    - You selectively stage just those changes you want to be part of your next commit, which adds **only** those changes to the staging area.
    - You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

Installing & Using Git

Installing Git

- **Installing on Windows**

  Download https://git-scm.com/download/win

- **Installing on macOS**

  Install the Xcode Command Line Tools

  Try to run *git –version*

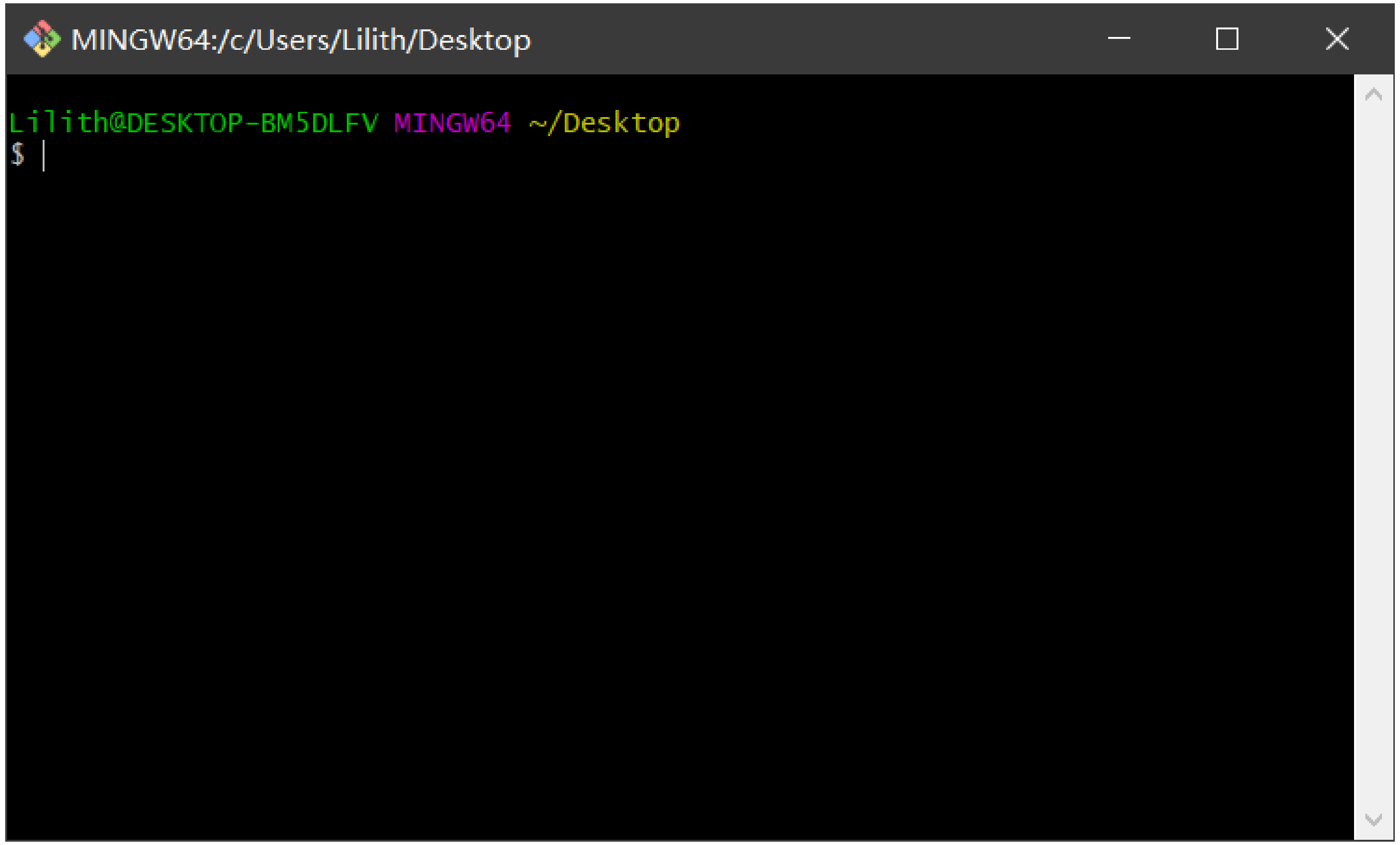
  Or download https://git-scm.com/download/mac

- **Installing on Linux**

  sudo apt install git-all

Using Git

Let’s do with Git Bash.



MINGW64:/c/Users/Lilith/Desktop

```
Lilith@DESKTOP-BM5DLFV MINGW64 ~/Desktop
$ 
```

Using Git - First-time Git Setup

You'll want to do a few things to customize your Git environment. You should have to do these things only once on any given computer; you can also change them at any time.

- We will use *git config* command
  - To view the origin settings: *git config --list --show-origin*
  - To set your indentity: *git config --global user.name "John Doe "*
    git config --global user.email johndoe@example.com
  - (You may also set your default editor)
  - Check your settings: *git config --list*
    Or: *git config user.name*
    git config user.email

上海科技大学 ShanghaiTech University

Using Git − Getting Help

- *git help <verb>*
- *git <verb> --help*
- *git <verb> -h*

Examples

- *git help config*
- *git config --help*
- *git config –h*

- We will do more latter.

Using Git – 2 Ways to Get a Repository

1. You can take a local directory that is currently not under version control, and turn it into a Git repository.
   - *git init*

1. You can **clone** an existing Git repository from elsewhere.
   - *git clone [some url or ..]*
   - Or you can do with SSH in GitHub, then you don't have to enter password every time.
     It is smooth, but need some preparation.
     See https://docs.github.com/cn/authentication/connecting-to-github-with-ssh

To get the project, we use the second way: **clone**.

Using Git –

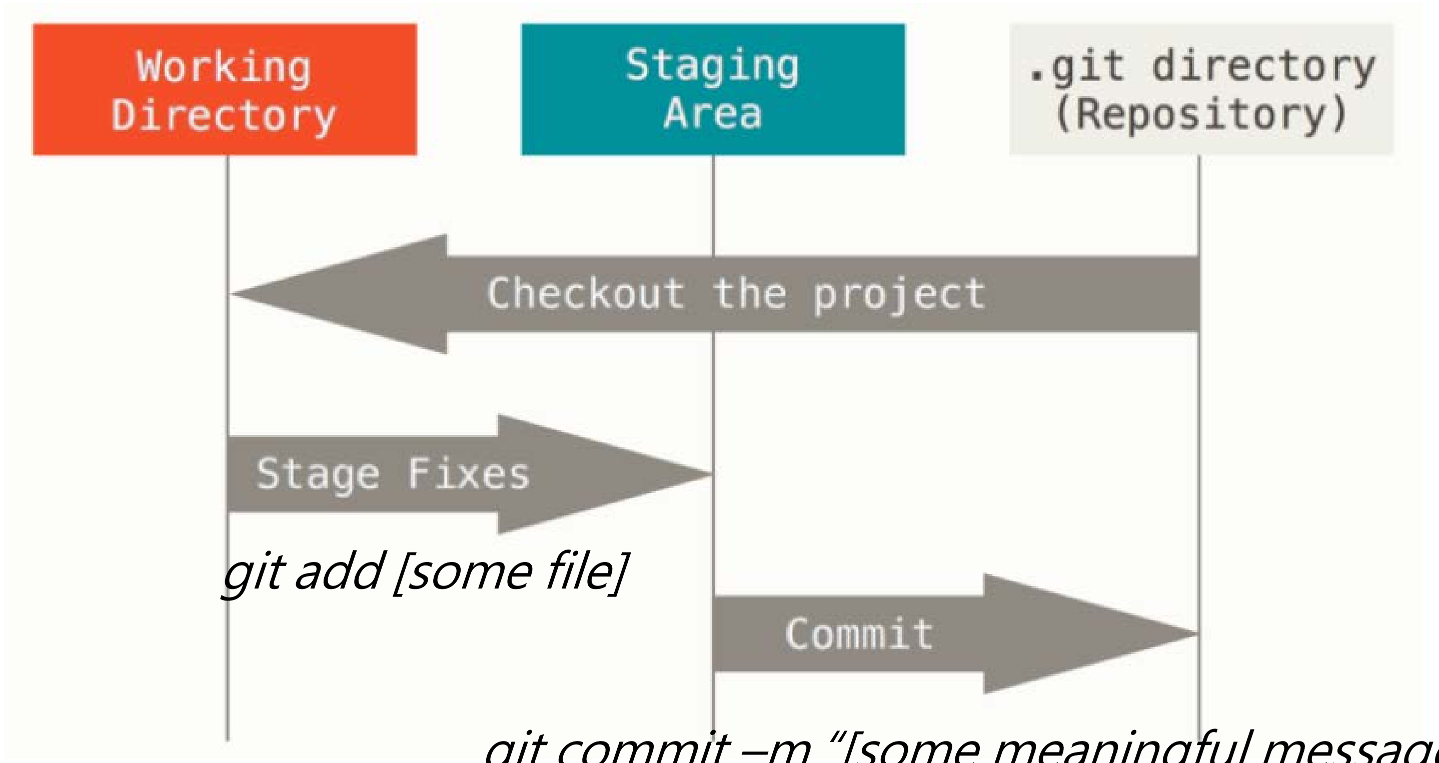
Recording Changes to the Repository

The Three States

Here is the main thing to remember about Git if you want the rest of your learning process to go smoothly.

- **Modified** means that you have changed the file but have not committed it to your database yet.
- **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
- **Committed** means that the data is safely stored in your local database.

Checking the Status of Your Files: *git status*



git add [some file]

git commit –m "[some meaningful message]"

Using Git –

Recording Changes to the Repository

Checking the **Status** of Your Files: *git status*

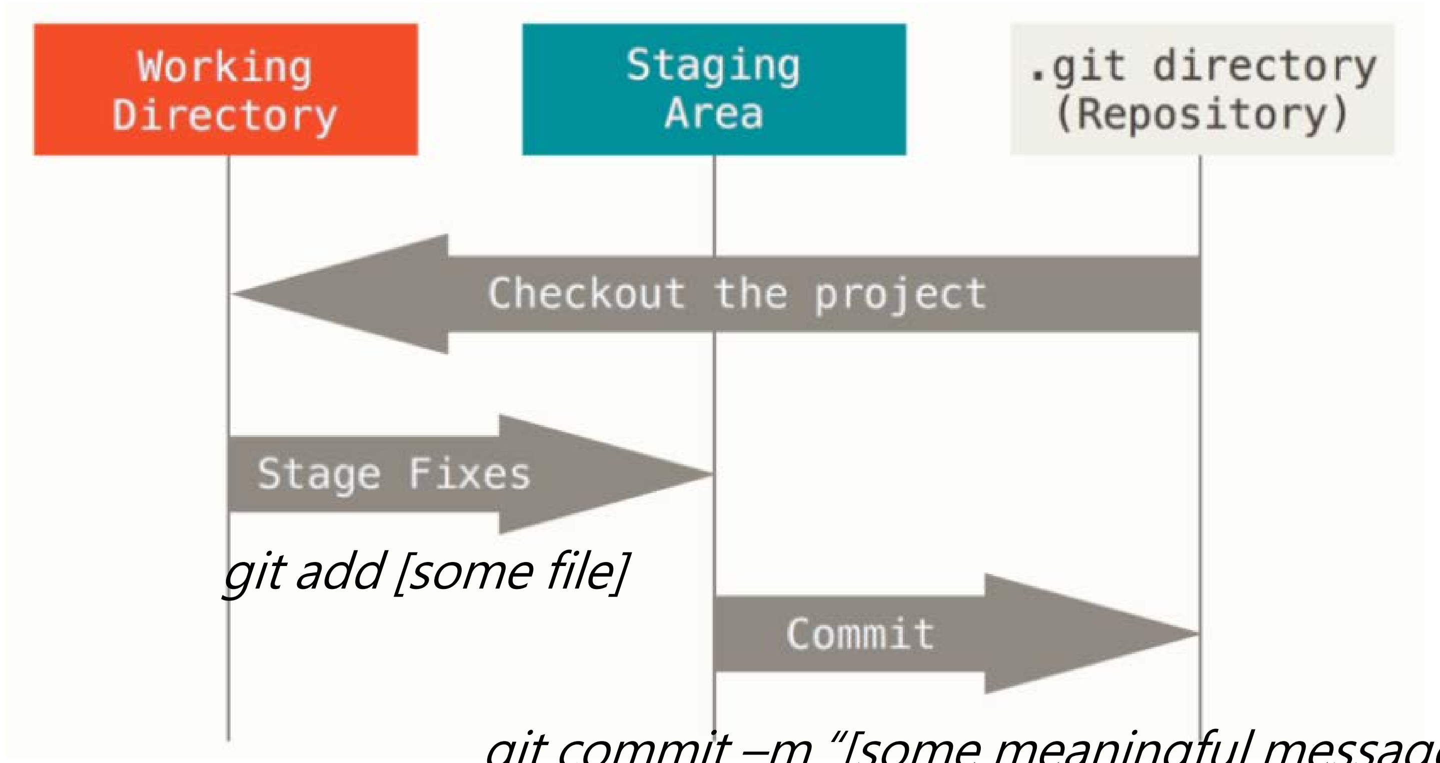
Another useful trick: the .*gitignore* file

To see what you've changed:

git diff

git diff –staged

I usualy do this with vscode graphically.



git add [some file]

git commit –m "[some meaningful message]"

Using Git - Viewing the Commit History

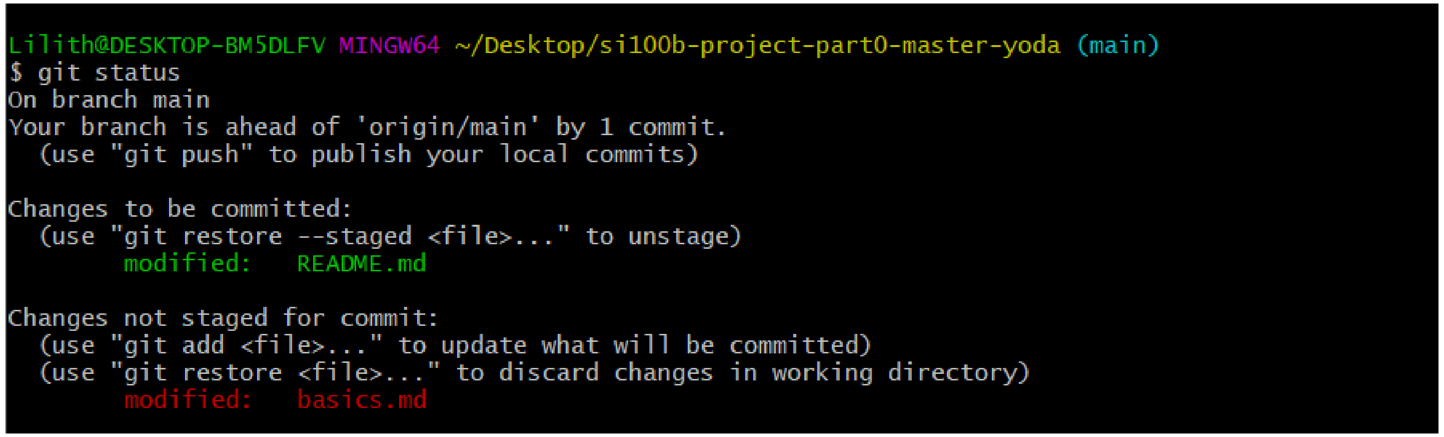
- *git log*
- *git log -p -2*
- *git log --stat*
- *git log --oneline*
- *git log --graph --oneline*

Using Git – Undoing Things (can be dangerous)

- Be careful, this is one of the few areas in Git where you may lose some work if you do it wrong.
- If you want to redo that commit, make the **additional changes** you forgot, stage them, and commit again using the *--amend* option. (useful when there is a typo in the last commit)
  - *e.g.*

```
$ git commit -m 'Initial commit'
$ git add forgotten_file
$ git commit --amend
```

- *git restore / git restore --staged* Can be extremely dangerous!



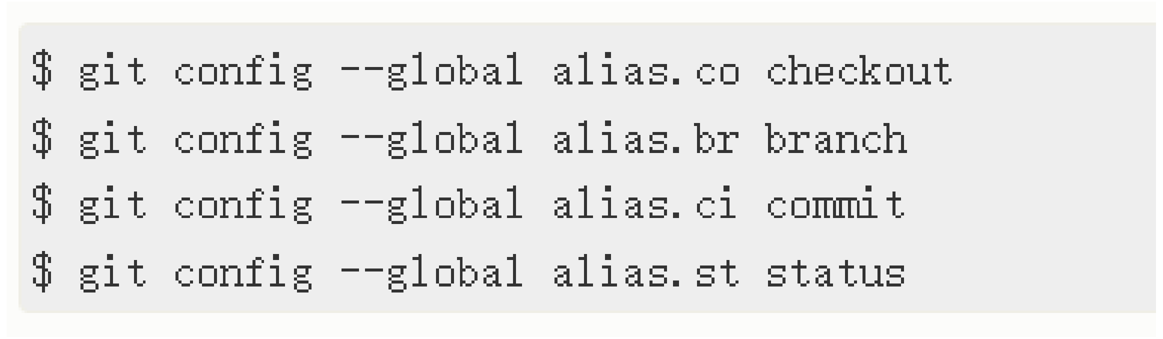
```
Lilith@DESKTOP-BM5DLFV MINGW64 ~/Desktop/si100b-project-part0-master-yoda (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   basics.md
```

Using Git - Working with Remotes

- To be able to collaborate on any Git project, you need to know how to manage your remote repos.
- Use *git remote, git remote -v* to see your remote.
- Use *git pull/fetch* to pull down all the data from that remote project that you don't have yet.
    - *check out the difference*
- Use *git push <remote> <branch>* to push your work to the remote.
    - e.g. *git push origin main*

Using Git - Git Aliases

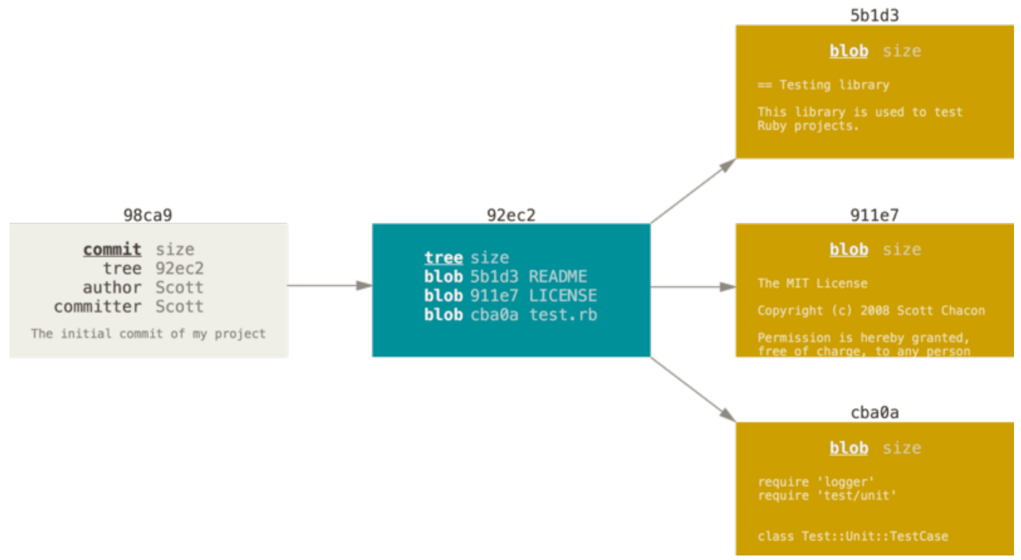
- e.g.

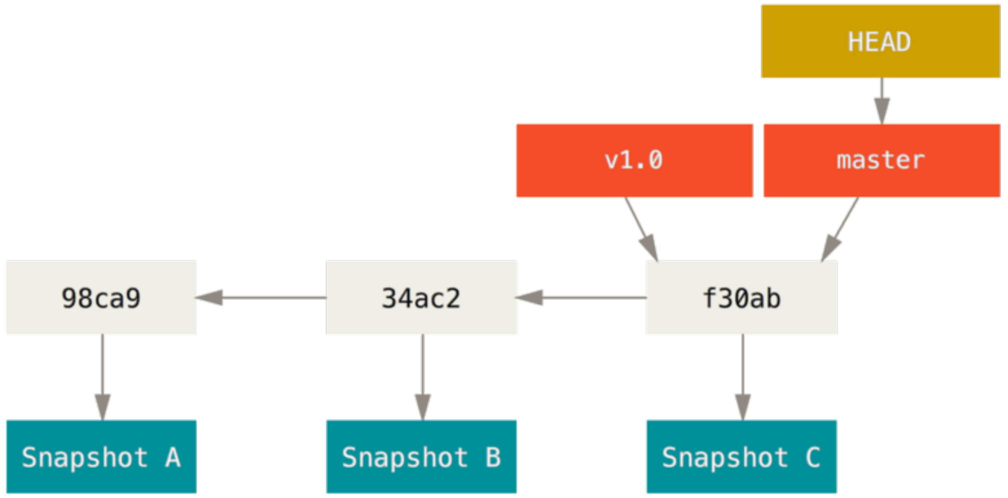


```
$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.st status
```

Using Git - Branching

- Git doesn’t store data as a series of changesets or differences, but instead as a series of **snapshots**.



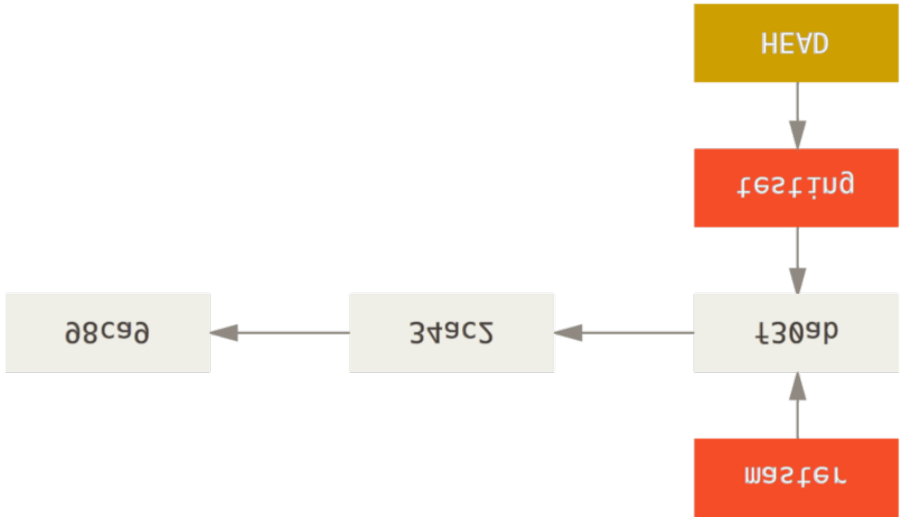


Using Git - Branching

- **Creating a New Branch**:

  git branch [new branch name]

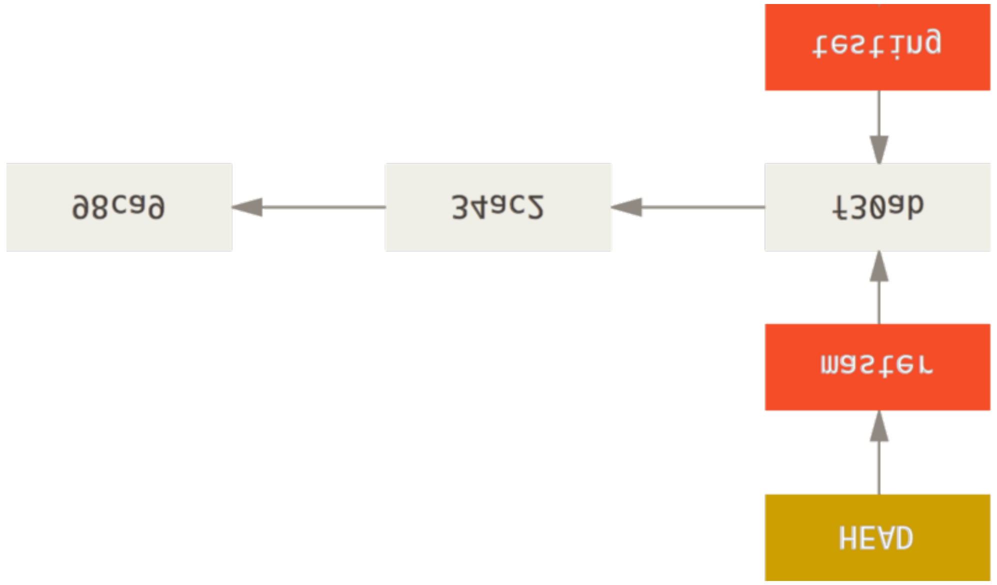
  e.g. *git branch test*



- **Switching Branches**

  git checkout [branch name]

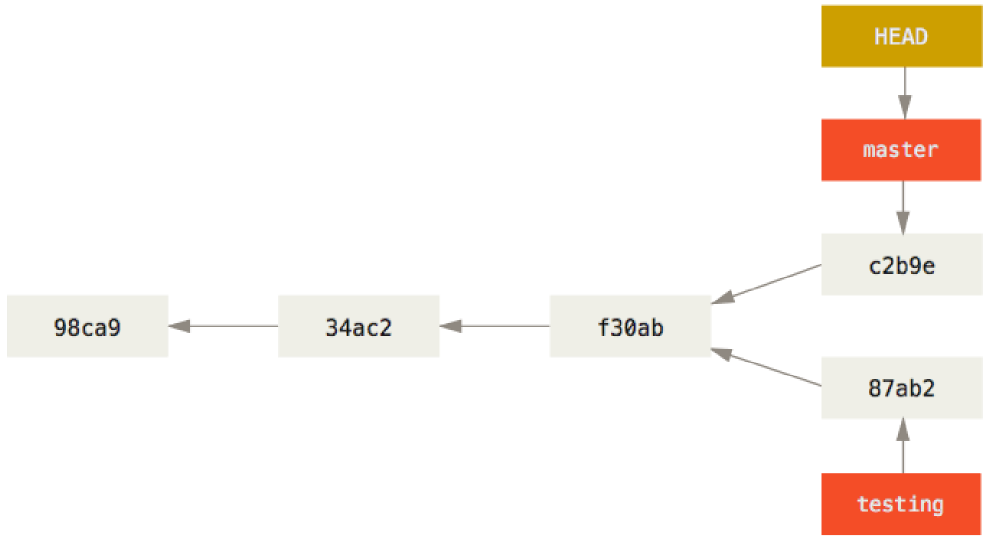
  e.g. *checkout test*



上海科技大学 ShanghaiTech University

Using Git - Branching

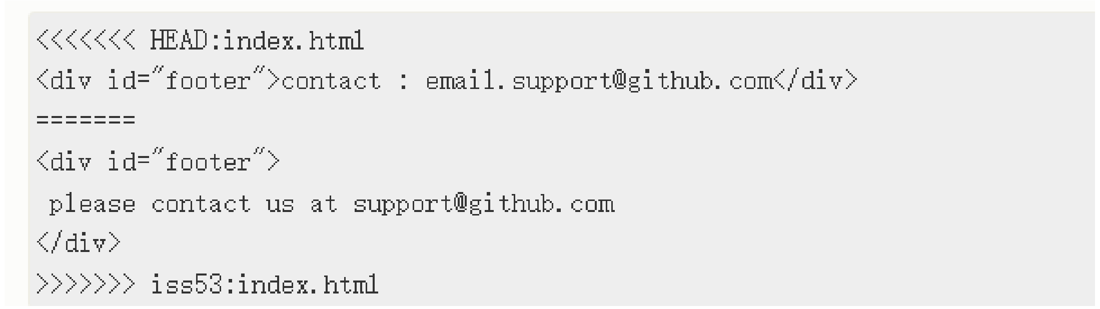
- After commits on different branches:



- May use *git log --all --graph --oneline* to view.

Using Git - Merging

- Use *git checkout [branch1]* and *git checkout [branch2]* to merge **branch2** into **branch1**.
- If there is a conflicts haven’t be resolved by git, resolve them manually.
  - Git adds standard **conflict-resolution markers** to the files that have conflicts, so you can open and edit them manually.
  - e.g. the sections between <<<<<<<, =======, >>>>>>>.



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

- Please check out a more detailed and concrete example
  - https://git-scm.com/book/zh/v2/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E6%96%B0%E5%BB%BA%E4%B8%8E%E5%90%88%E5%B9%B6

Thanks for your attention!

reference: https://git-scm.com/book/zh/v2