# Tutorial for Github

## What is Github?

GitHub is a code hosting platform for version control and collaboration using Git. You can use it to store your homework code, cooperate with others
on the same project and make a version control of your code.

## What is git?

Git is free and open source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

## Set up Git

### Step 1: Sign up for Github with an email account

https://Github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

### Step 2: Download and install git on your computer

- For macOS

  (1) Open the terminal in your mac

  (2) Check if git is installed in your mac

```
git
```

```
grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local
 branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

  If you got information above, it means that your mac has installed git.

  (3) If the terminal indicates that your mac doesn't install git

  First install homebrew in a macOS Terminal using command:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Then use homebrew to install git via command:

```
brew install git
```

Or you can check other install methods on https://git-scm.com/download/mac

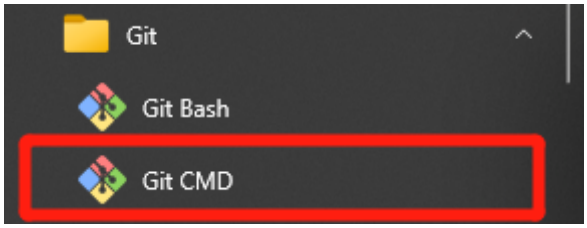
Or you can install Xcode on https://developer.apple.com/xcode/ via "Xcode"->"Preferences"->"Downloads"->"Command Line Tools"->"Install"

- For windows

  download the package on https://git-scm.com/download/win and install the package

## Step 3: Create SSH on your computer

- Open Git Bash




- Set a Git username

```
git config --global user.name "test"
```

- Set an email address in Git

```
git config --global user.email "email@example.com"
```

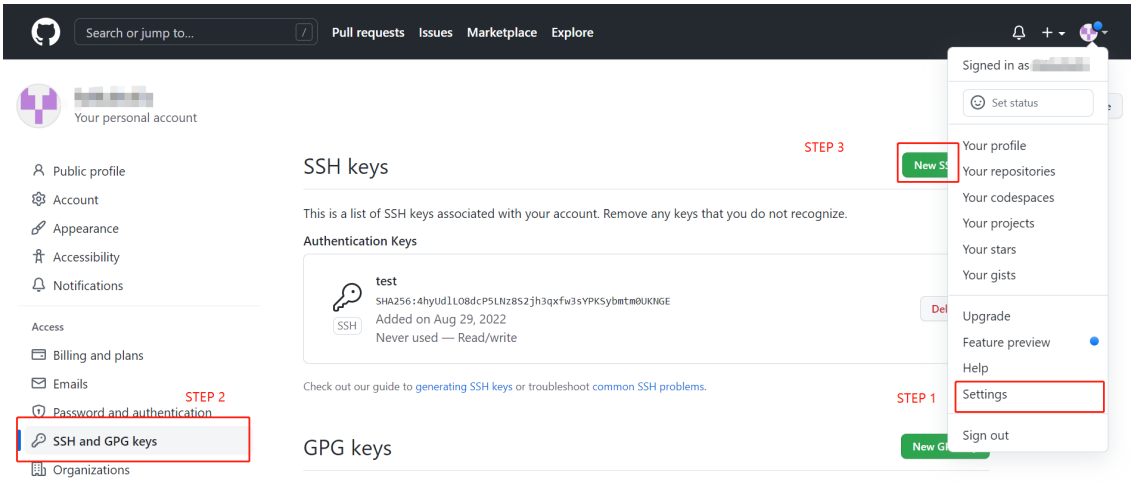
- Generate SSH key

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

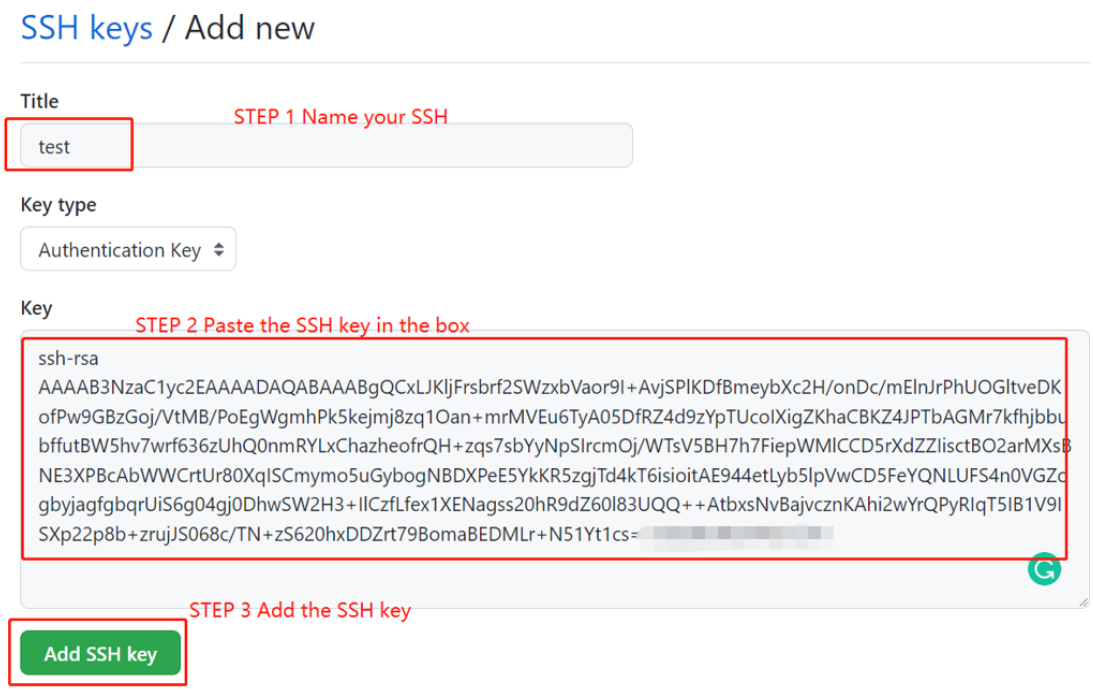
## Step 4: Create SSH on Github

- Add the generated public key to Github

  (1) Open Github->setting->SSH and GPG keys-> New SSH key

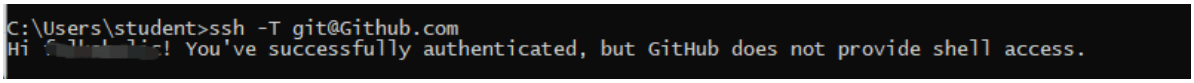



(2) Copy the generated SSH key from "../.ssh/id_rsa.pub" and then add the SSH keys to Github



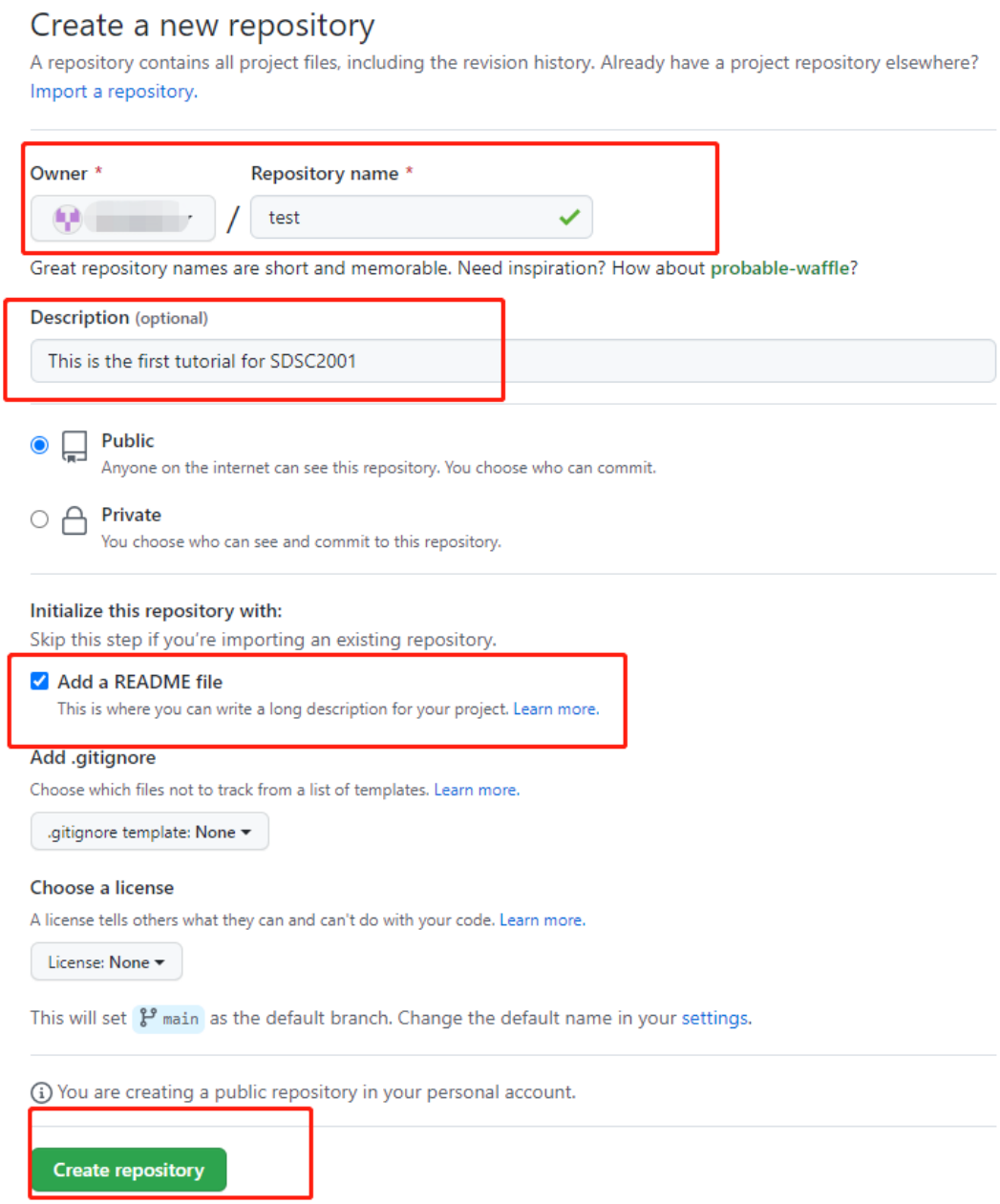

- Test the connection between Git and your Github

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




# Create a repository on Github

- In the upper-right corner of any page, use the drop-down menu, and select New repository
- In the Repository name box, name your repository
- In the Description box, add a description
- Select Add a README file
- Click Create repository




# The principle of Github

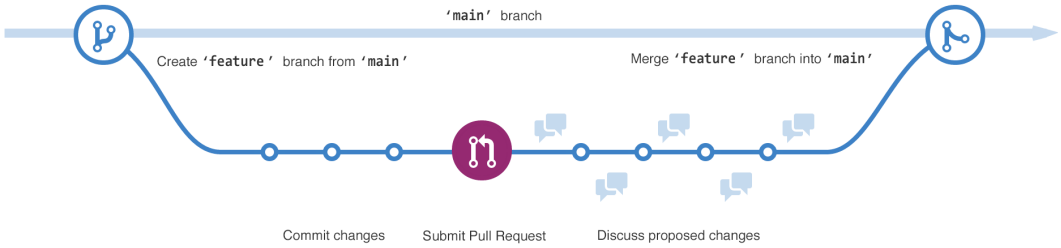
## Branch

Branching lets you have different versions of a repository at one time.

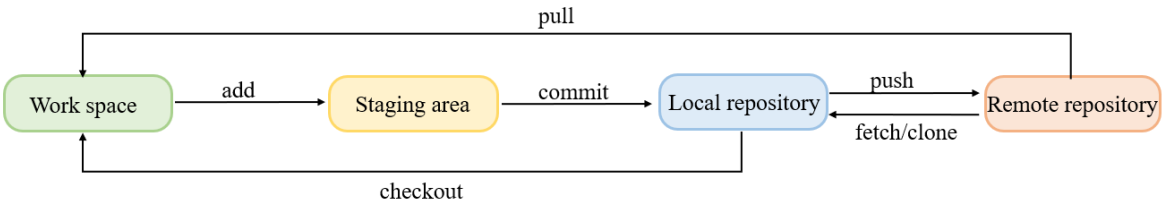
By default, your repository has one branch named `main` that is considered to be the definitive branch. You can create additional branches (called `feature`) off of `main` in your repository.

You can use branches to have different versions of a project at one time. The work done on different branches will not show up on the main branch until you merge it.

When you create a branch off the `main` branch, you're making a copy of `main` as it was at that point in time. If someone else made changes to the `main` branch while you were working on your branch, you could pull in those updates.




## Basic operations




- "git init": Initialize repository
- "git add":  Add  files to the staging area
- "git commit": Add the files in the staging area to the local repository
- "git push": Push the local repository to the remote repository
- "git checkout": Switch branches
- "git pull": Download remote code and merge to work space
- "git fetch": Download remote repository to local repository
- "git clone": Download remote repository to local repository

# Upload your code to your Github

- Enter your workspace by command or Git GUI
- Create new git repository in directory

```
git init
```

- Add all files to the staging area

```
git add .
```

- Add one file to the staging area

```
git add filename.filetype
```

- Add the files in the staging area to the local repository

```
git commit -m "notation about this commit"
```

- Rename current branch as main

```
git branch -M main
```

- Connect to the remote repository

```
git remote add origin https://github.com/xxxxx/test.git
```

- Before uploading a project to the remote repository, pull the repository to avoid conflicts

```
git pull --rebase origin main
```

- Push the local repository to the remote repository

```
git push -u origin main
```

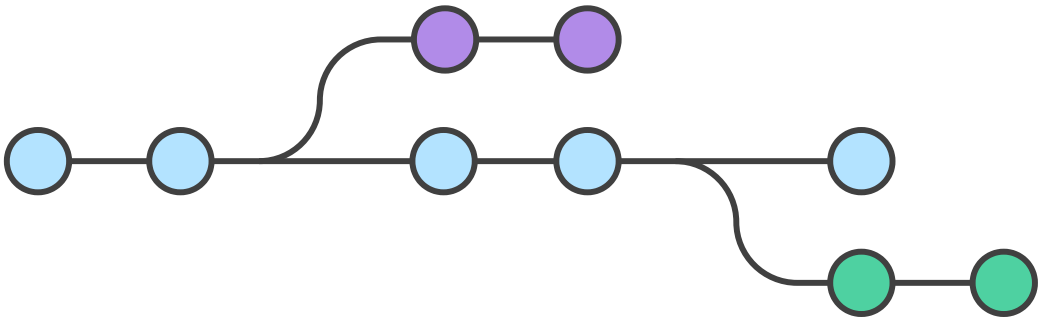
- Check your repository in the github

## Download the code base from Github

- Enter your workspace by command or Git GUI
- Download the remote repository via command:

```
git clone https://github.com/xxxxx/test.git
```

## Version control



A project may be completed by a group, git can make sure that all group members share a non-conflicting version of your repository.

- Check current branch

```
git status
```

```
C:\Users\student\Desktop\tutorial_1>git status
On branch master
```

- Create a new branch

```
git branch branchname
```

- Switch branch

```
git checkout branchname
```

```
C:\Users\student\Desktop\vc>git checkout b1
Switched to branch 'b1'
```

- Add a file in new branch

```
git add test2.txt
```

- Add the new file in the staging area to the local repository

```
git commit -m "example of version control"
```

- Check branches

```
git branch
```

```
C:\Users\student\Desktop\vc>git branch
* b1
  main
```

- Switch branch

```
git checkout main
```

```
C:\Users\student\Desktop\vc>git checkout main
Switched to branch 'main'
Your branch is up to date with 'b1/main'.
```

- Merge branch

```
git merge branchname
```

```
C:\Users\student\Desktop\vc>git merge b1
Updating 31d4c8b..89b47d7
Fast-forward
 test2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test2.txt
```

- Check branches

```
git branch
```

```
C:\Users\student\Desktop\vc>git branch
  b1
* main
```

- Delete the branch and check

```
git branch -d branchname
git branch
```

```
C:\Users\student\Desktop\vc>git branch -d b1
Deleted branch b1 (was 89b47d7).

C:\Users\student\Desktop\vc>git branch
* main
```

- Push the local repository to the remote repository

```
git push -u origin main
```

- Check update on github