

Introduction to GIT II

Cloud Infrastructure Engineering

**Nanyang Technological University
& Skills Union - 2022/2023**

Course Content

- Quick Check-In
- Git 2FA
- Git commit signature verification
- Initialize new project from local
- Create new branch and some changes
- Pull Request
- Git merge

Self Study Check-In



Q1: How to secure your GitHub Account?



Q2: Explain about 2FA?



Q3: What is the difference between “git clone” with
“git pull”?



Q4: What is the difference between “git pull” with
git push?



Git 2FA



What is 2FA and Git 2FA

Two-factor authentication (2FA) is an **extra layer of security** used when logging into websites or apps. With 2FA, you have to log in with your **username** and **password** and provide **another form of authentication** that only you know or have access to.

For GitHub, the second form of authentication is a code that's generated by an application on your mobile device or sent as a text message (SMS). After you enable 2FA, GitHub generates an authentication code any time someone attempts to sign into your account on GitHub.com. The only way someone can sign into your account is if they know both your password and have access to the authentication code on your phone.

Activity

Instructor demonstrate how to set up Git 2FA

Activity

Let's spend 5 - 10 mins for Learners to set up 2FA on their account

Activity(Recap)

- Create new repository with README.md file
- Pull that repository to your computer
- Create some changes to README.md file
- Push the changes to your repository

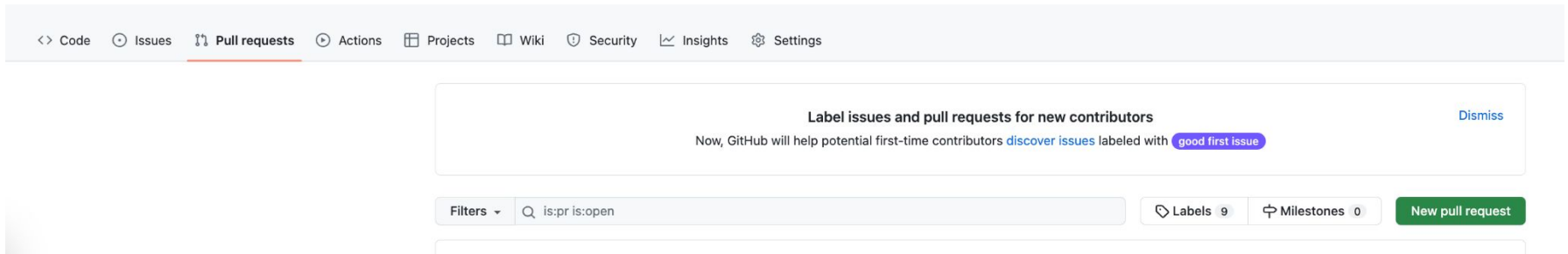
Pull Request & Git Merge



Pull Request

A pull request is an event in Git where a contributor asks a other member of a Git repository to review code they want to merge into a project.

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.



Step-by-step

- Create new repository with a professional name and include README.md file
- `git clone`
- `cd` into your repository
- `git checkout -b update-readme-1`
- Create some changes on README.md file. **SAVE.**
- `git status`
- `git add .`
- `git commit -m "Modify README.md"`
- `git push --set-upstream origin update-readme-1`

Create a pull request

Delete branch after merge PR

- `git checkout main`
- `git pull`

Activity

Instructor demonstrate how to create pull request.

- Create new branch
- Create the changes on the new branch
- Push to git
- Open pull request

Activity

Learners open new pull request on their own repository.



If there is no changes made on the new branch
(main branch and new branch is the same),
can pull request be opened?

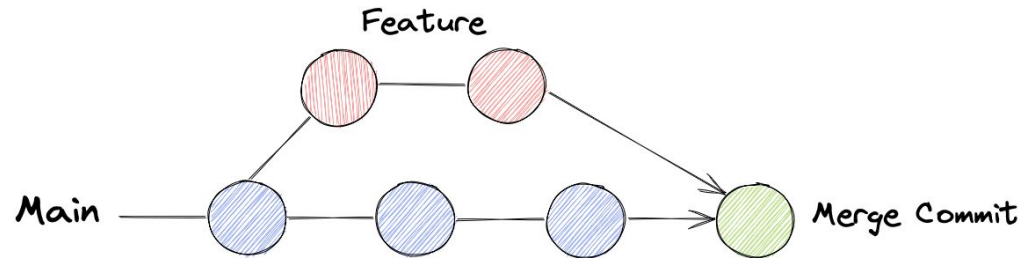
Break 10 mins



“git merge”

The "merge" command is used to integrate changes from another branch.

While Git can perform most integrations automatically, some changes will result in conflicts that have to be solved by the user.



Activity

Use the same repo as previous activity

- **git checkout main => to make sure you are in the main branch**
- **git pull => to make sure you have the latest changes in the main branch**

Activity

Run the code in the following order(start from main branch)

- **git checkout -b my-first-feature**
- **git checkout main**
- **git checkout -b my-second-feature**
- **git branch**

```
PS C:\Users\jaze\OneDrive\Desktop> git clone https://github.com/jaezeu/merge-tutorial.git
Cloning into 'merge-tutorial'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
PS C:\Users\jaze\OneDrive\Desktop> cd .\merge-tutorial\
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git pull
Already up to date.
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git checkout -b my-first-feature
Switched to a new branch 'my-first-feature'
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git checkout -b my-second-feature
Switched to a new branch 'my-second-feature'
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git branch
  main
  my-first-feature
* my-second-feature
```

Activity

While in the “my-second-feature” branch, make some changes and push to github

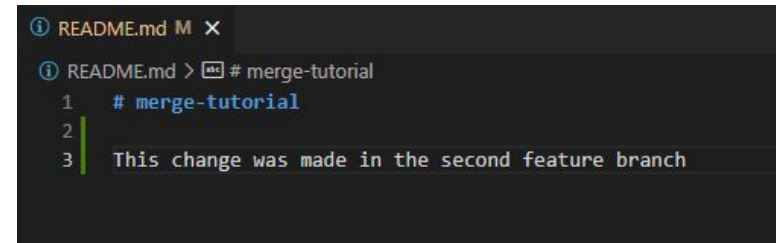
- **git status**
- **git add .**
- **git commit -m “added second feature”**
- **git push --set-upstream origin my-second-feature**

```
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git status
On branch my-second-feature
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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git add .
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git commit -m "add second feature"
[my-second-feature 1a68870] add second feature
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git push
fatal: The current branch my-second-feature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin my-second-feature

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetUpRemote' in 'git help config'.
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git push --set-upstream origin my-second-feature
```

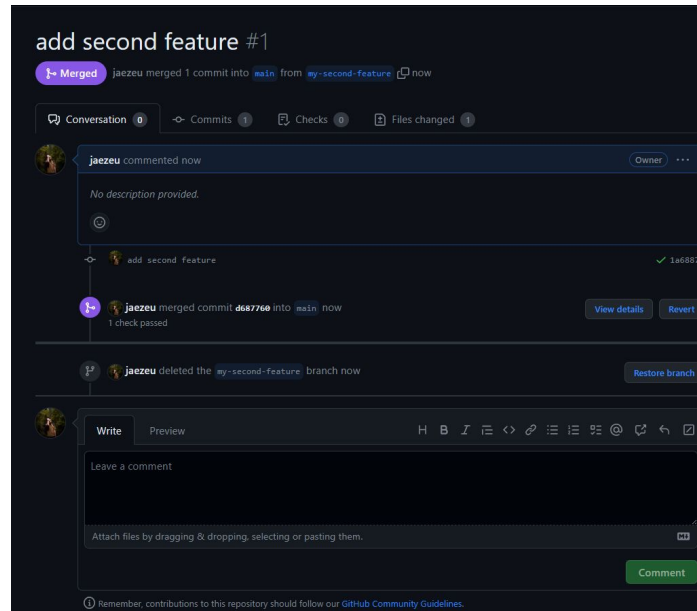


The screenshot shows a code editor with a dark theme. The file name 'README.md' is at the top. The content of the file is as follows:

```
# merge-tutorial
1 # merge-tutorial
2
3 This change was made in the second feature branch
```

Activity

Create a Pull request to merge your “**my-second-feature**” branch to the main branch and **Delete the branch**



Activity

- git checkout main
- git pull

```
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 626 bytes | 62.00 KiB/s, done.
From https://github.com/jaezeu/merge-tutorial
   94b5fb8..d687760  main       -> origin/main
Updating 94b5fb8..d687760
Fast-forward
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial>
```

Activity

- `git status` => verify that the previous slide is completed and you are in the main branch
- `git checkout my-first-feature`
- `git merge main`
-

```
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git checkout my-first-feature
Switched to branch 'my-first-feature'
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial> git merge main
Updating 94b5fb8..d687760
Fast-forward
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\jazee\OneDrive\Desktop\merge-tutorial>
```

Activity

While in the “my-first-feature” branch, make some changes and push to github

- **git status**
- **git add .**
- **git commit -m “added first feature”**
- **git push --set-upstream origin my-first-feature**

```
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git status
On branch my-first-feature
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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git add .
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git commit -m "first feature added"
[my-first-feature e66c4de] first feature added
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git push
fatal: The current branch my-first-feature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin my-first-feature

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetUpRemote' in 'git help config'.
PS C:\Users\jaze\OneDrive\Desktop\merge-tutorial> git push --set-upstream origin my-first-feature
```

```
① README.md X
① README.md > [add] # merge-tutorial
1  # merge-tutorial
2
3  This change was made in the second feature branch
4
5  This change was made in the first feature branch
```

Markdown cheatsheet

[markdown-cheatsheet/README.md at master · tchapi/markdown-cheatsheet · GitHub](#)



gitignore

[Git - gitignore Documentation \(git-scm.com\)](https://git-scm.com/docs/gitignore)



Git Resume

<https://resume.github.io/>



What's Next?

<https://education.github.com/git-cheat-sheet-education.pdf>



Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.
- Check the AWS account after learner clean up.

Useful Links



Activity

\$ git checkout main

\$ git merge feature-1

Activity

Instructor demonstrate how to merge in github website and in terminal

- \$ git branch feature1
- \$ git checkout feature1
 - create changes on README.md
- \$ git add .
- \$ git commit -m "Changes on readme.md"
- **\$ git push --set-upstream origin feature1**
- \$ git checkout main
- \$ git merge feature1
- **\$ git push**

Git commit signature



Git commit signature

When you sign a commit, you **use a private key to create a digital signature** for the commit. This signature is then **attached to the commit** and **can be verified by anyone** that can access your repository.

Signing Git commits helps with two important aspects of the software delivery:

- **Security** – By signing your commits, you can help protect your repository from tampering or modification by unauthorized parties.
- **Authenticity** – Git commit signing lets you verify the identity of the person who made the commit, and it helps ensure that the commit is genuine.

You can sign commits and tags locally, to give other people confidence about the origin of a change you have made. If a commit or tag has a **GPG**, **SSH**, or **S/MIME** signature that is cryptographically verifiable, GitHub marks the commit or tag "Verified" or "Partially verified."

Setting up our Git to sign commits

Add changes on README.md

Delfrinando Pranata authored and Delfrinando Pranata committed 18 hours ago



91e3f25



Initial commit



del-skillsunion committed 18 hours ago

Verified



1a4d4d2



Activity

Instructor demonstrate how to set up Git commit signature using SSH followed this instruction:

<https://phoenixnap.com/kb/generate-ssh-key-windows-10>

Generate new SSH Key

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Adding new SSH Key to github

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

Activity

```
$ ssh-keygen -t ed25519 -C "your_email@email.com"  
$ pbcopy < ~/.ssh/id_ed25519.pub
```

For Windows / WSL

```
$ clip < ~/.ssh/id_ed25519.pub
```

Add key to GitHub (<https://github.com/settings/keys>)

Activity

Let's spend 15 - 20 mins for Learners to set up Git sign commits using SSH

Activity

Change your remote's URL from HTTPS to SSH with the git remote set-url command.

- `$ git remote set-url origin git@github.com:OWNER/REPOSITORY.git.`

Activity

After learners finish ssh setup, lets try to commit a changes and see the result on github.

- Create some changes to README.md file
- Push the changes to your repository