INFORMATION & NOTICE

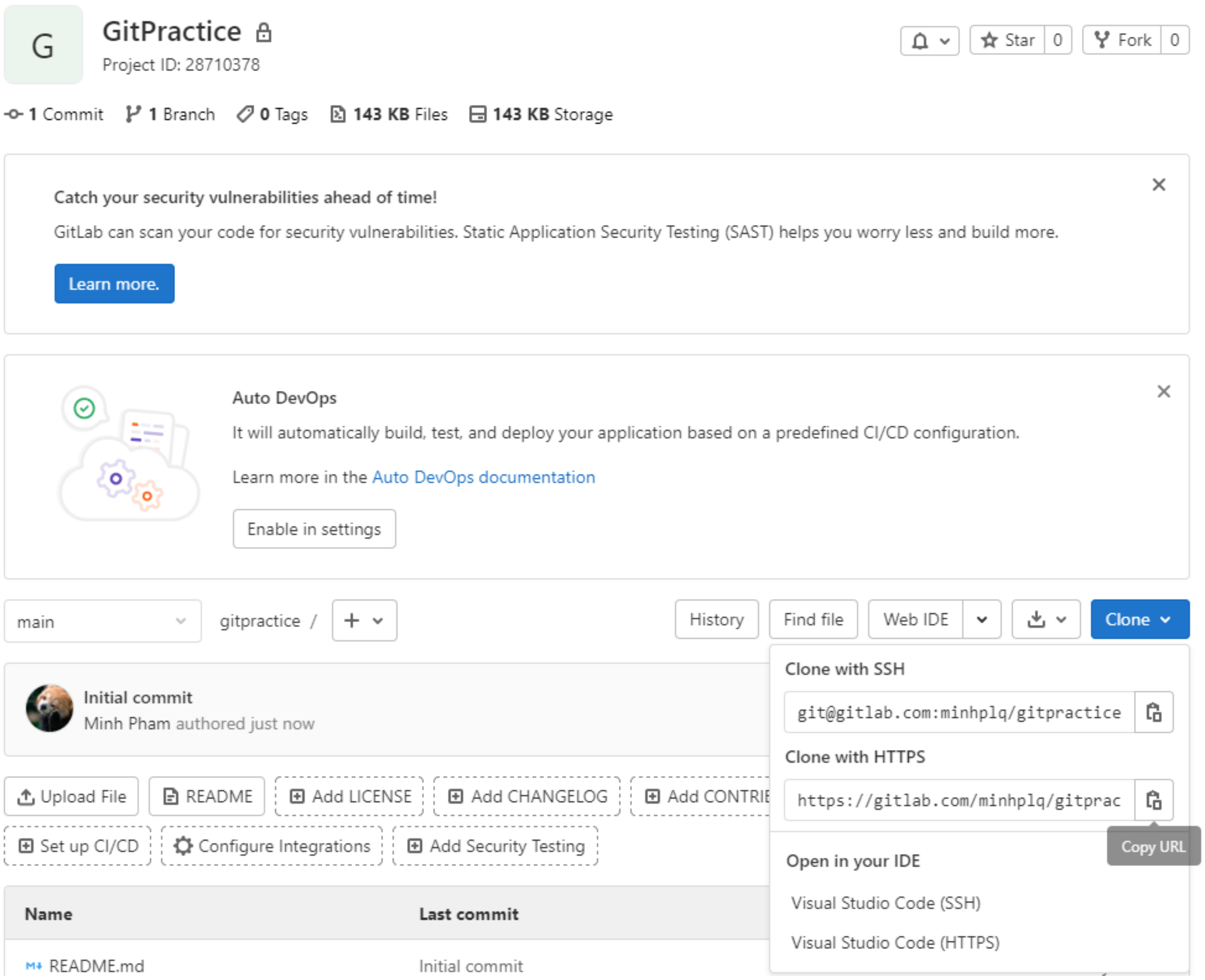
- *You must install Git and setup environment before doing*
- *Follow rules of Git*

Step 1: Go to your GitLab link and create a repository with name *GitPractice*

New Project -> Create blank project

Step 2: Get your Repository

Clone ->



GitPractice

Project ID: 28710378

1 Commit · 1 Branch · 0 Tags · 143 KB Files · 143 KB Storage

Catch your security vulnerabilities ahead of time!

GitLab can scan your code for security vulnerabilities. Static Application Security Testing (SAST) helps you worry less and build more.

Auto DevOps

It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.

Learn more in the Auto DevOps documentation

Initial commit

Minh Pham authored just now

Clone with SSH

`git@gitlab.com:minhplq/gitpractice`

Clone with HTTPS

`https://gitlab.com/minhplq/gitprac`

Open in your IDE

Visual Studio Code (SSH)

Visual Studio Code (HTTPS)

Name	Last commit
README.md	Initial commit

Congratulations! Go to the next page to start!

PRACTICES

In this practice, you will be one in characters below:

Character	Role	Working On	Branch name
WORKER1	Setup trunk for project	Working1	setup_trunk
WORKER2	Do something related A feature	Working2	feature_a
WORKER3	Set up trunk for project	Working3	setup_trunk
REVIEWER	Do something related B feature	Working4	feature_b
	Resolve Merge request	Gitlab	NA
	Can Push on Main directly	NA	NA

Just an example (^_^)

Let pay your attention, use right **Working on** and **Branch name** for each Character

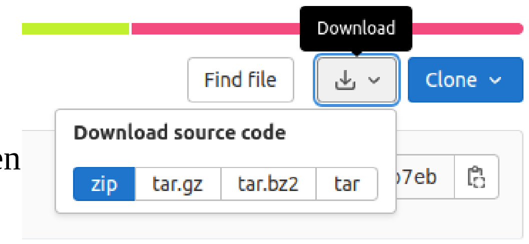
You are WORKER1 *(You will work on **Working1** folder)*

1. **[Clone]**

   Git from your Repository to your PC with name: **Working1**

2. **[Add & Push]**

   - Open https://gitlab.com/minhplq/trunk and download data from it. *(It just an example of trunk, don't mention it)*
   - Create a branch with name **setup_trunk** then switch to it
   - Extract the downloaded data then add into **setup_trunk** branch
   - Push onto remote Repository



3. **[Delete & Commit]**

   Create **build** folder. Then push on remote Repository

Take a note here, after adding the empty folders above, build folder also is deleted. That means, Git stores are based on file, not folder!

You are WORKER2 *(You will work on* ***Working2*** *folder)*

4. **[Rebase]**
   - Clone your Repository to your PC with name: **Working2**
   - Create a branch with name **feature/a** then switch to it
   - Get data from ***setup_branch*** branch through ***Rebase***
5. **[Revert]**
   - Open ***buildWin.bat*** and modify (anything you want)
   - Open ***buildAndroid.bat*** and modify (anything you want)
   - Don't commit, revert all to HEAD (the same after Rebase)
6. **[Modify]**

   Open ***applyPatches.bat*** and put the text below at end of file, then Push on Repository

```
echo nothing to do here
```

You are WORKER3 *(You will work on **Working3** folder)*

7. **[Ignore]**

   - Clone your Repository to your PC with name: **Working3**
   - Switch to ***setup_trunk*** branch
   - Create folder ***./build, ./bin, ./debug, ./data***
   - Create file ***./build/build.txt, ./bin/bin.tmp, ./debug/debug.cache, ./data/csv/data.csv, ./data/text/another_data.txt***

   another_data.txt

   ```
   This is a data file
   ```

   - Ignore: folders: ***./build, ./bin, ./debug***

     files: ****.tmp, *.cache***, ****.txt*** except files in ***/data***

   Push on remote Repository

You are WORKER1 *(You will work on **Working1** folder)*

8. **[Submodule]**

   Link the trunk with submodules below then push on remote Repository

Where	Repository
/submodules/ssh-private-key	https://gitlab.com/gitlab-examples/ssh-private-key
/submodules/docker	https://gitlab.com/gitlab-examples/docker

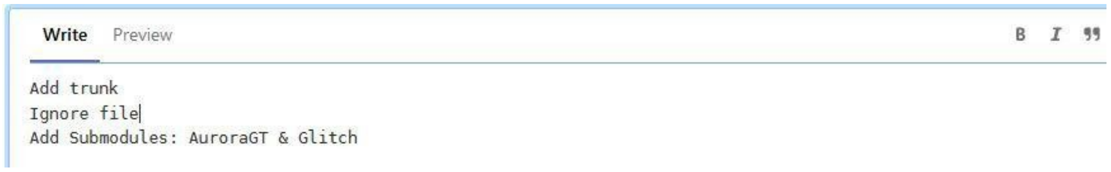
9. **[Create Tag]**

   Create a Tag with name: **0.0.1** and message "***setup trunk***"

10. **[Merge request]**

    Create a merge request from source branch is ***setup_trunk*** to target branch is ***master***:

    - Title: Setup trunk
    - Description:



```
Add trunk
Ignore file
Add Submodules: AuroraGT & Glitch
```

    - Assignee: assign to yourself

You are REVIEWER *(You will work on **Working4** folder and **GitLab**)*

11. **[Accept merge request]**

    Go to Gitlab link and accept the ***merge request*** above

12. **[Resolve conflict 1]**

    - Clone your Repository to your PC with name: **Working4**
    - Create a branch with name **feature/b** then switch to it
    - Open ***applyPatches.bat*** and put the text below at end of file, then commit to local Repository

    ```
    echo nothing to do here
    echo this line: nothing to do here too
    ```

    - Merge from ***feature/a*** branch

    Well done! A conflict will be occured. Resolve it through *“**use their**”* solution. Then push on remote Repository

13. **[Update]**

    Modify file ***another_data.txt*** in ***./data/text***

    Change source code below (end of file), then push on remote Repository

    from

    ```
    This is a data file
    ```

    to

    ```
    This is a text data file
    ```

14. **[Merge to master]**

    Merge ***feature/b*** branch into ***master*** branch and push on remote Repository

You are WORKER2 *(You will work on **Working2** folder)*

15. [Move]

- Move all of file from ***./data/csv*** and ***./data/text*** to ***/data/file,*** delete ***./data/csv*** and ***./data/text*** *folder*.

  Then push on remote Repository
- Create a merge request from source branch is ***feature/a*** to target branch is ***master.*** With title, description that thinking is right for ***feature/a*** branch. Of course, assign to yourself
- Go to Gitlab link and accept the ***merge request*** **(**Of course, you are Reviewer)

16. [Create Patch]

- Get data from ***master*** through ***Rebase***
- Add source code below into end of file: ***/data/file/another_data.txt***

```
Add another data line
```

- Create a **patch** after modifying and revert everything has just changed

Note: keep the patch to do next practice

You are REVIEWER *(You will work on **Working4** folder)*

17. **[Apply Patch]**

    Get the patch from ***WORKER2,*** apply to ***feature/b*** and commit on local Repository

18. **[Revert specific commit]**

    Revert changes from commit of practice ***13.[Update]***

    Merge ***feature/b*** branch into ***master*** branch and push on remote Repository

Congratulations! You have just completed the Practice