

Git project structure



Questions:

Question

What do we need to do after **modifying a file tracked by Git?**

- ☒ We need to stage the file, so that the changes will be included in the next commit.
- ☐ We need to commit the file, so that the changes will become part of the staging area.
- ☐ We need to add the file to the Git directory.
- ☐ We need to change to a different working tree.

[Skip](#)

[Submit](#)

Staging area (index)

A file maintained by Git that contains all of the information about what files and changes are going to go into your next commit

Question

What should your commit message look like?

- ☐ A jumble of words
- ☐ A description of no more than 50 characters
- ☒ A short description of the change (up to 50 characters), followed by one or more paragraphs giving more details of the change (if needed).
- ☐ Always write as much as you can about the changes.

✓ Correct

Question

When committing new files or changes with `git commit`, the user is asked to provide a commit message. What will happen if an empty commit message is entered?

- ☐ It will make it difficult to track bugs without commit messages.
- ☐ The info shown with the `git log` command will show no commit message.
- ☒ The commit will be aborted.
- ☐ The commit will ignore untracked files or files that weren't staged.

✓ Correct

Best way to write git commit message:

```
user@ubuntu:~$ cat example_commit.txt
Provide a good commit message example
```

```
The purpose of this commit is to provide an example of a hand-crafted,
artisanal commit message. The first line is a short, approximately 50 character
summary, followed by an empty line. The subsequent paragraphs are jam-packed
with descriptive information about the change, but each line is kept under 72
characters in length.
```

```
If even more information is needed to explain the change, more paragraphs can
be added after blank lines, with links to issues, tickets, or bugs. Remember
that future you will thank current you for your thoughtfulness and foresight!
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
```

```
#
```

```
# On branch master
```

```
#
```

```
# Changes to be committed:
```

```
# new file:   super_script.py
```

```
# new file:   cool_config.txt
```

```
#
```

```
user@ubuntu:~$ █
```

Few commands:

...or create a new repository on the command line

```
echo "# blog-website-django" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/nimesh44/blog-website-django.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/nimesh44/blog-website-django.git
git branch -M main
git push -u origin main
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.