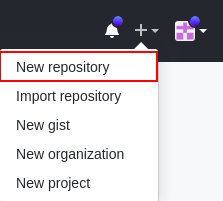
**Create a git repository**

To create a git repository, you need to have a Github account. Follow the steps below to create a github account and a git repository:

* Open [Github](https://github.com/join). If you don't already have a Github account, create one by entering a username, email, and password. If you already have a Github account proceed to the next step.
* Log in to your account from the [Github](https://github.com/) login page.
* Click the **+** sign in the top-right corner of the page and click then on **New repository**.



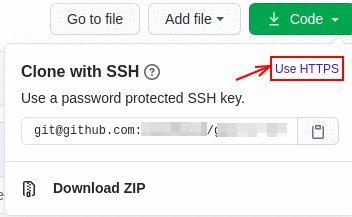
* Enter your repository name in the field **Repository name** and add a project description in the **Description** field.
* You can either select **public** or **private** to restrict repository accessibility. If **public**,anyone can see the repository but you still choose who can commit to it. If **private**, you choose who can see and commit to the repository.
* Check the option **Initialize this repository with a README** to initialize the repository with a README file. Leave all the other values to their default.
* Click the **Create repository** button.

**Git operations**

You now need to create a local copy of this remote repository on your machine. We'll do this by cloning the repository. The syntax for this is:

git clone [URL]

For the URL, you can either choose an **SSH** or an **HTTPS** link as a URL. We will use HTTPS to clone the Git repository. Click on **Clone or download** and select HTTPS. Copy the **HTTPS** link by clicking on the Copy button beside the link.



Next, go to your **linux-instance** terminal and replace [URL] from the above syntax with the link you copied. The command should now look similar to:

git clone https://github.com/[username]/[git-repo].git

Here, **username** is the Git username and **git-repo** is the name of the remote repository you created.

This creates a directory with the same name as your repository, initializes a **.git** directory inside it, pulls down all the data for that repository, and creates a working copy of the latest version.

You can now list the files using the **ls** command and find your new repository. Move into your repository using **cd** command. There, you'll see the project files, which are ready to be worked on or used.

cd directory\_name

Replace the **directory\_name** with your repository's name that you just initialized.

If you want to clone the repository into another directory of your choice, you can do that by passing the name of the directory. This automatically creates a new directory with the specified name and initializes the repository inside it.

**Syntax:**

git clone [URL] directory\_name

**Configure Git**

Git uses a username to associate commits with an identity. It does this by using the **git config** command. Set Git username with the following command:

git config --global user.name "Name"

Replace **Name** with your name. Any future commits you push to GitHub from the command line will now be represented by this name. You can use **git config** to even change the name associated with your Git commits. This will only affect future commits and won't change the name used for past commits.

Let's set your email address to associate them with your Git commits.

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

Replace **user@example.com** with your email-id. Any future commits you now push to GitHub will be associated with this email address. You can also use **git config** to even change the user email associated with your Git commits.

**Edit the file and add it to the repository**

Now, edit the README file by using nano editor:

nano README.md

Add any text within the file, or you can use the following text:

I am editing the README file. Adding some more details about the project description.

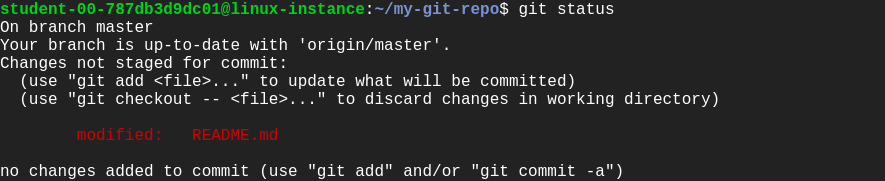
Save the file by pressing Ctrl-o, Enter key, and Ctrl-x.

We can check the status using the following command:

git status

The git status command shows the different states of files in your working directory and staging area, like files that are modified but unstaged and files that are staged but not yet committed.

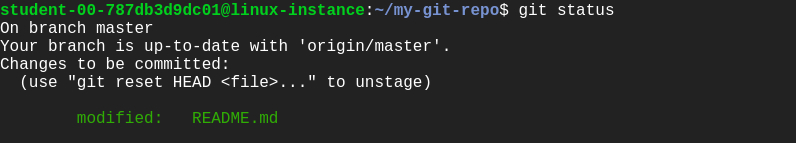
You can now see that the README.md file shows that it's been modified.



Now, let's add the file to the staging area using the following command:

git add README.md

Use the **git add** command to add content from the working directory into the staging area for the next commit. When the git commit command is run, it looks at this staging area. So you can use git add to craft what you'd like your next commit snapshot to look like.



Let's now commit the changes. A Git commit is like "saving" your work.

Commit the changes using the following command:

git commit

This now opens an editor that asks you to type a commit message. Every commit has an associated commit message. A commit message is a log message from the user describing the changes.

Enter the commit message of your choice or you can use the following text:

I am editing the README file.

Once you've entered the commit message, save it by clicking Ctrl-o and the Enter key. To exit click Ctrl-x.

The **git commit** command captures a snapshot of the project's currently staged changes. It stores the current contents of the index in a new commit along with the commit message.

You've successfully committed your file!

Now, push the committed changes from your local repository to a remote repository on the **master** branch by using:

git push origin master

Next, enter your Github username/email ID and password to prompt the associated remote repository to push the changes.

**Note:** If you enabled two-factor authentication in your Github account you won't be able to push via HTTPS using your account’s password. Instead you need to generate a personal access token. This can be done in the application settings of your Github account. Using this token as your password should allow you to push to your remote repository via HTTPS. Use your username as usual. For more help to generate a personal access token, click [here](https://help.github.com/en/github/authenticating-to-github/creating-a-personal-access-token-for-the-command-line#creating-a-token).

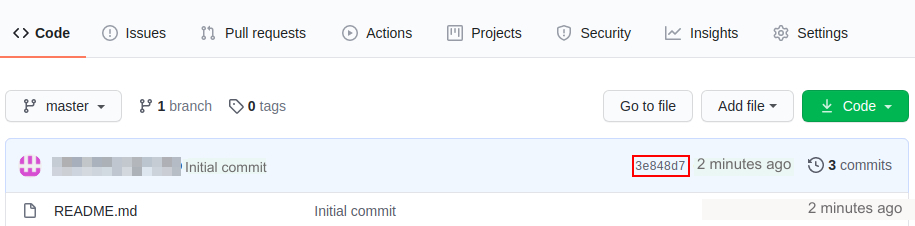
Click *Check my progress* to verify the objective.

Edit the file and add it to repository

Check my progress

You can check the changes made to the local README.md file on the remote repository on [Github](https://github.com/). You can see the last time when the README.md file was added/updated.

You can also see the commit ID just above the list of files in the repository. Click on the Commit ID to get more details related to the commit.



**Create a new file and commit it to the repository**

You now need to create a new file **example.py** on the local git repository in the working directory. To do this, use the following command:

nano example.py

Add the following Python script to the **example.py** file:

def git\_opeation():

print("I am adding example.py file to the remote repository.")

git\_opeation()

Save the file by pressing Ctrl-o, Enter key, and Ctrl-x.

Now, repeat the same procedure by adding a file to the staging area for next commit:

git add example.py

Commit the changes:

git commit

Enter a commit message and save it by pressing Ctrl-o and the Enter key. To exit click Ctrl-x.

We will push these changes later in the lab.

Click *Check my progress* to verify the objective.

Create a new file and commit it to the repository

Check my progress

**Add an empty file to the repository through web UI**

Now, let's create an empty file on the remote repository using the Github website.

1. Go to your repository on the Github website and click on the **Add file** button, then click on **Create new file**. This will open a new page.



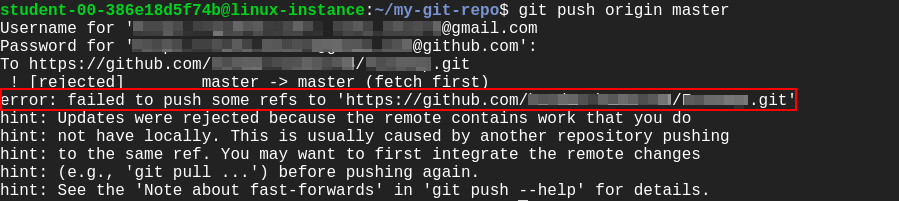
1. Enter the file's name in the box beside your repository's name. Leave the contents of the file empty.
2. Scroll down and enter a commit message in the first box under **Commit new file** section.
3. Leave the rest on its default value and click the **Commit new file** button.

You've successfully committed a new file through the website.

Now, let's push the changes made on the local repository that weren't pushed. Switch back to your terminal and enter the following command:

git push origin master

Output:



The last command throws an **error**. This is because the files added or change done on a remote repository (the Github website) isn't present yet on your local repository, but we're trying to push something from the local repository to the remote repository. To push changes from the local repository, we need to first update the local repository from the remote repository.

Let's now pull the current snapshot/commit in the remote repository to the local repository:

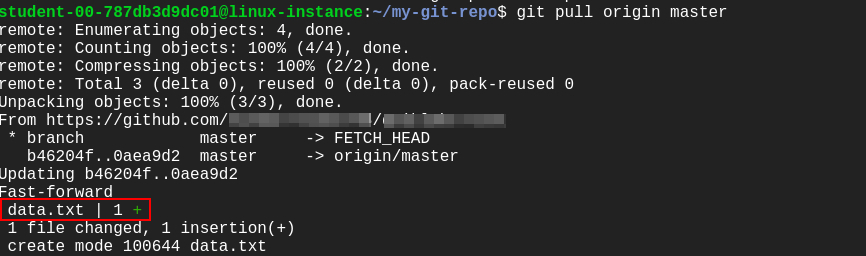
git pull origin master

This opens an editor that asks you to enter a commit message for the merge operation (remote repository to local repository).

You can simply accept the default message or type your own message. To continue, save the file by pressing Ctrl-o, Enter key, and Ctrl-x.

The git pull command is used to fetch and download content from a remote repository and update the local repository to match that content.

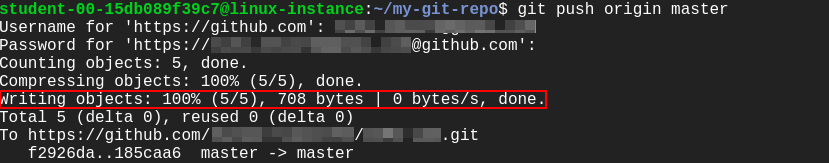
**Output:**



Now try pushing the changes again.

git push origin master

**Output:**



This shows that your local repository is now up-to-date with your remote repository and you successfully pushed the changes to the remote repository.

Click *Check my progress* to verify the objective.

Add an empty file to the repository through web UI

Check my progress

**Congratulations!**

Great job! You've successfully set up a Github account, logged in, created a repository, made changes on the local machine, and pushed changes back to the remote repository. You've learned how to share local changes from a local to remote repository and vice-versa. These skills will help you as an IT specialist to work with a remote user on the same repository.