2.1 Install Git on Linux

This section will guide you to:

* Install Git on Linux

This lab has two subsections, namely:

1. Verifying the installation
2. Installing Git

**Step 2.1.1:** Verifying the installation

Please note: Git is already installed in your lab. Execute the below command to check the version. Execute **Step 2.1.2** in case you don’t get any results for **git –version**

**$ git --version**

****

**Step 2.1.2:** Installing Git:

Execute the following commands on the terminal to install Git.

**$ sudo apt-get update**

**$ sudo apt-get install git**



2.2 Create a Github Repository

This section will guide you to:

* Create a github repository

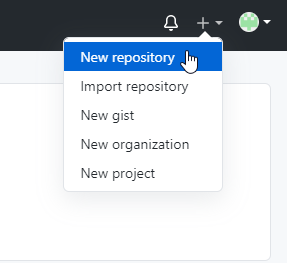
This lab has three subsections, namely:

1. Creating a new github repository
2. Editing the README file
3. Uploading a file to the repository

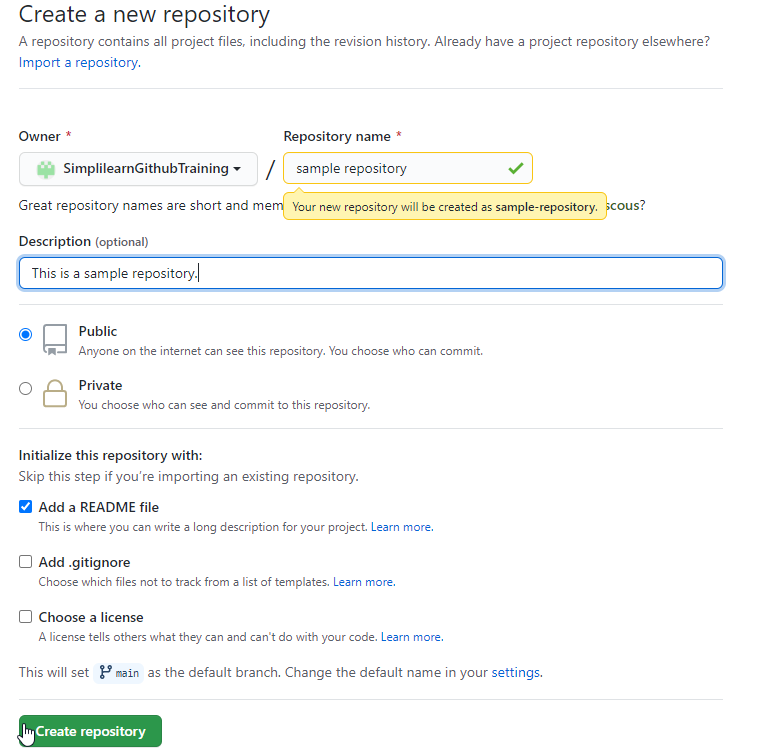
**Please Note**: As you have already done the DevOps Certification Training, you might already have a GitHub account and a repository. If not please refer to the GitHub demo in the DevOps Certification Training to create a GitHub account and follow the steps below to create a repository.

**Step 2.2.1:** Creating a new github repository

* Open the browser in your lab, go to **github.com** and login to your account
* Click on the **+** icon from the upper-right corner of the page and select **New repository** from the drop-down menu

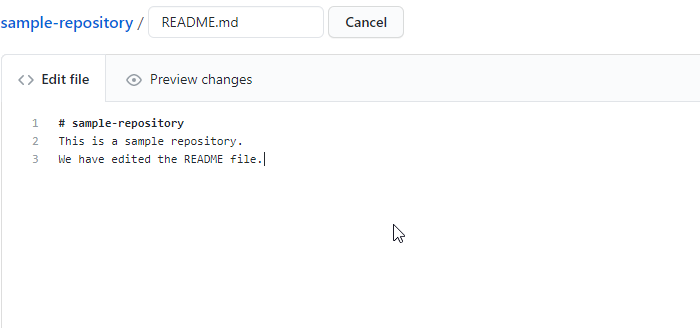


* Enter the **Repository name.** Entering **Description** is optional.
* Choose **Public** for the repository type
* Select **Initialize this repository with a README** to include a README file for the repository
* Click on the **Create Repository** button

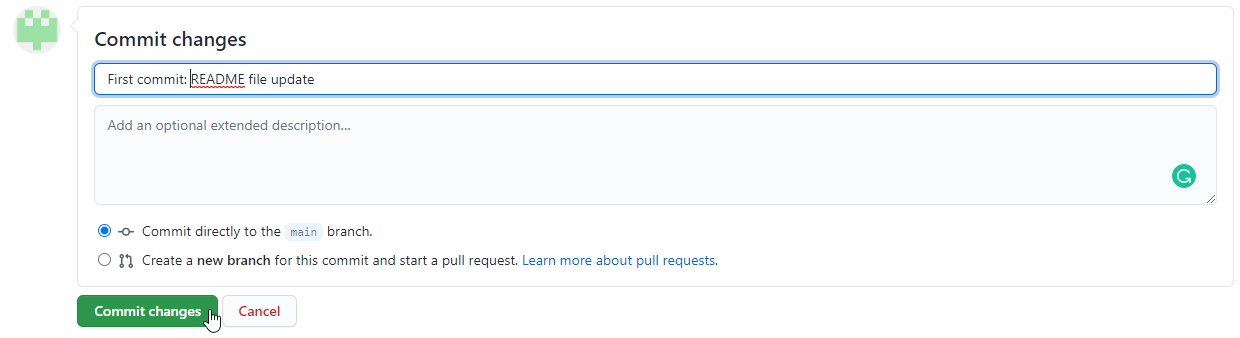


**Step 2.2.2:** Editing the README file

* Click on the **Edit** button to edit the README file



* Type a commit message to describe the changes made in the file (**Example**: First commit: README file update)



**Step 2.2.3:** Uploading a file to the repository

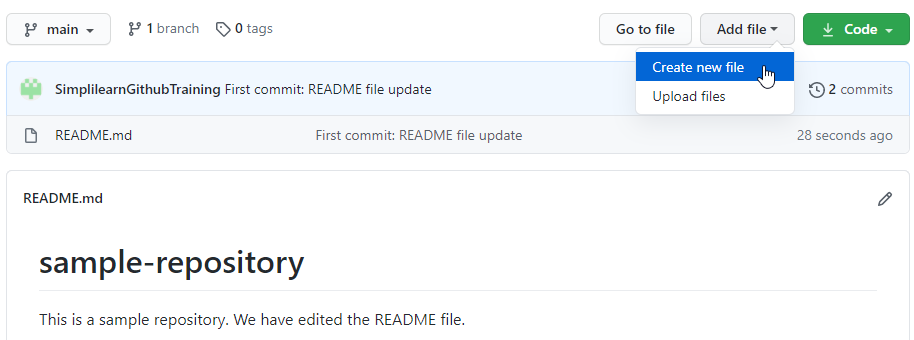
* Create a new file in your lab inside your home directory similar to **/home*/*<YourUsername>** with the name **GitandGithubCourseFile.txt** by executing the below command in the terminal

**vi GitandGithubCourseFile.txt**

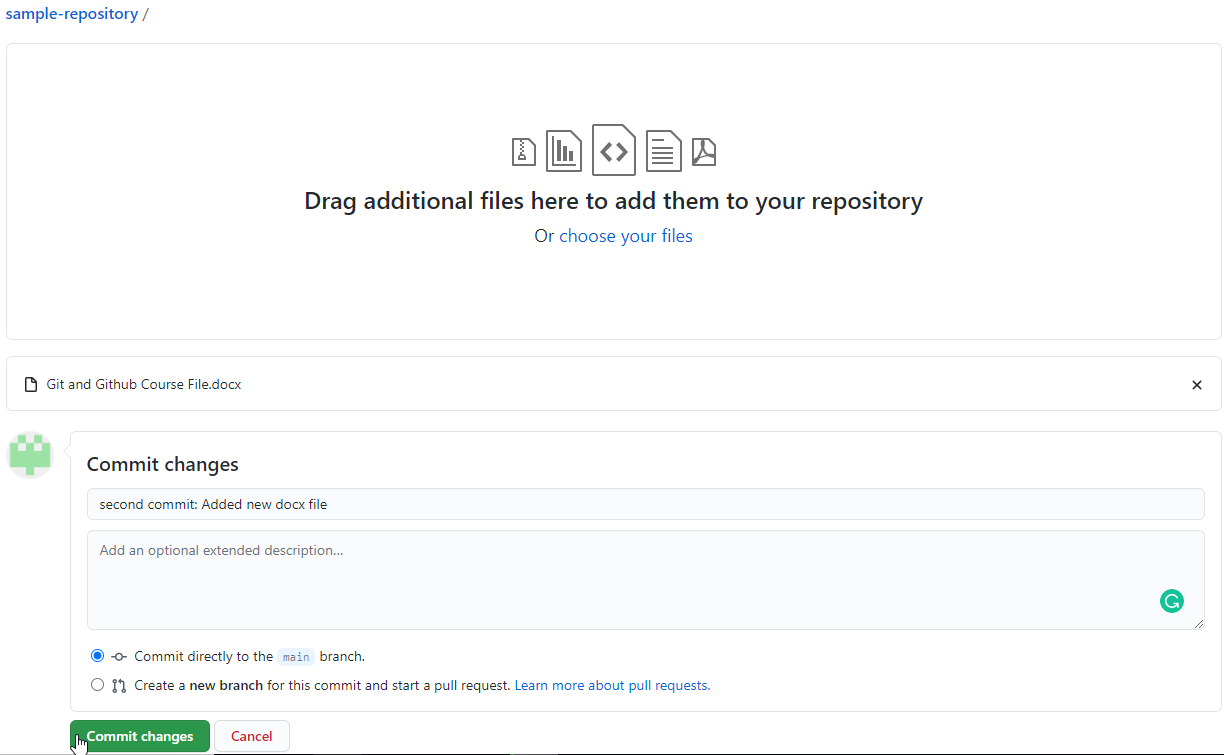
* Add the below line in **GitandGithubCourseFile.txt** file and save it.

*This is a test file*

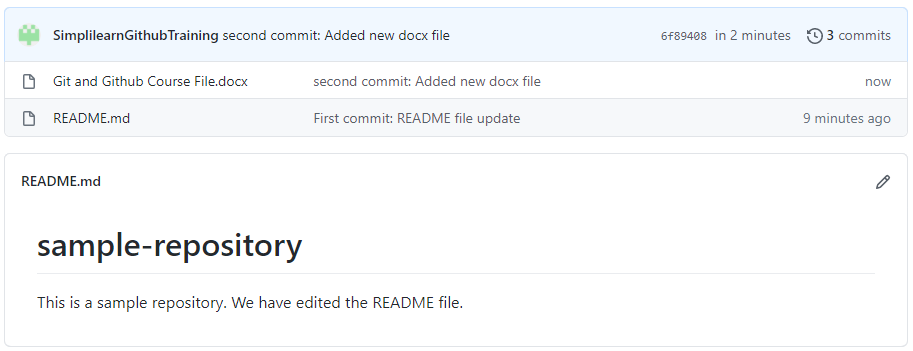
* Go to your Github repository and to add a file to the repository, click on **Add file**, and then select **Upload files.**



* Select the files from **choose your files**
* Enter a commit message (**Example**: second commit: Added new docx file), and then click on the **Commit changes** button



* Check the newly added file with the commit message



2.3 Configure Git

This section will guide you to:

* Configure Git

This lab has three subsection, namely:

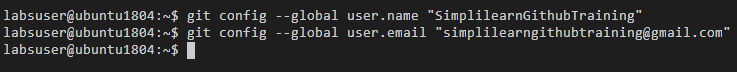
1. Configuring Git with username and email id
2. Confirming the username and email id
3. Enabling credentials storage globally

**Step 2.3.1:** Configuring the username and email id. (Execute the following commands in the terminal)

**$ git config --global user.name "USERNAME"**

**$ git config --global user.email "USERMAIL"**

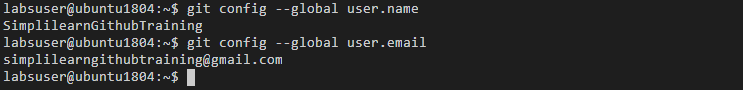
Note: In place of *USERNAME* and *USEREMAIL* use your github account’s username and email id.



**Step 2.3.2:** Confirming the username and email id

**$ git config --global user.name**

**$ git config --global user.email**



**Step 2.3.3:** Enabling credentials storage globally

**$ git config --global credential.helper store**



2.4 Clone a GitHub Repository

This section will guide you to:

* Clone a github repository

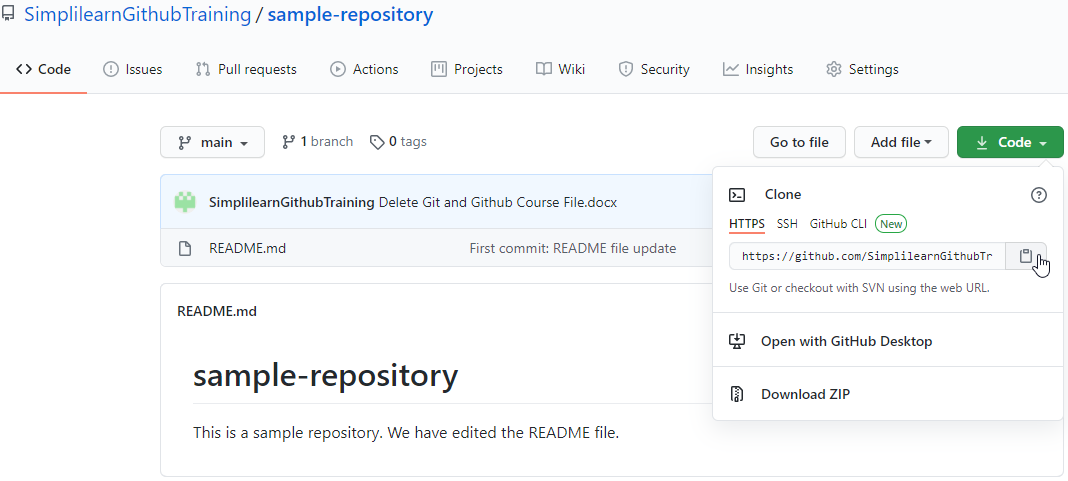
This lab has one subsections, namely:

1. Cloning a GitHub repository

**Please Note**: You should already have a GitHub repository which you created in assisted practice 2.2. If not, first complete the demo 2.2 and then proceed with this demo.

**Step 2.4.1:** Cloning a GitHub repository

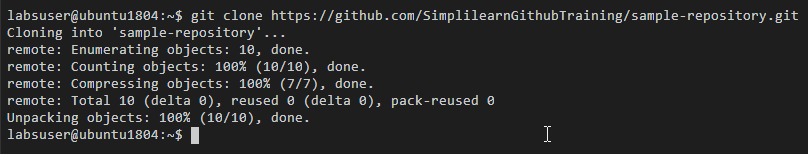
* Open the browser in your lab, Go to **github.com** and login to your account
* Navigate to the repository you wish to clone and click on the **Code** button
* Copy the *url* provided under **HTTPS**



* Open the **Terminal** on the local machine and use the following command to clone the repository:

***$ git clone URL***

where URL is the copied url from the repository



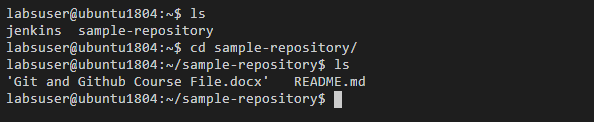
* Use the following commands to check the cloned repository:

***$ ls***

***$ cd Repository\_Name***

where Repository\_Name is the name of cloned repository

***$ ls***



2.5 Create a Git Alias

This section will guide you to:

* Create a Git alias

This lab has one subsections, namely:

1. Creating an alias

**Step 2.5.1:** Creating an alias

* Open the terminal and execute the following commands to create the respective aliases:

***$ alias gs='git status'***

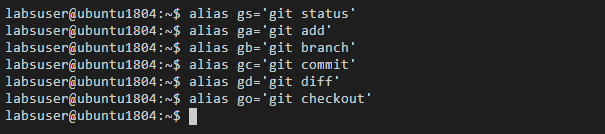
***$ alias ga='git add'***

***$ alias gb='git branch'***

***$ alias gc='git commit'***

***$ alias gd='git diff'***

***$ alias go='git checkout'***



3.1 Demonstrate the Centralized Git Workflow

This section will guide you to:

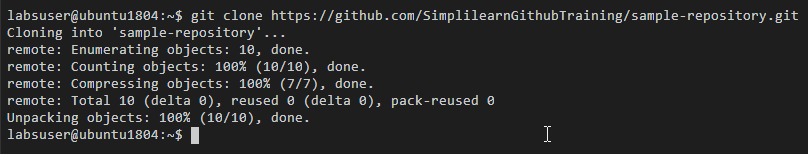
* Demonstrate the git workflow

This lab has five subsections, namely:

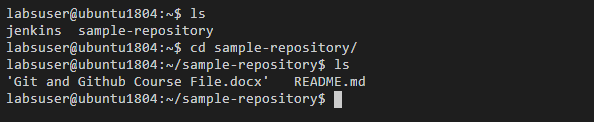
1. Cloning a github repository
2. Adding a file to the cloned repository
3. Checking the status of the repository
4. Adding the file to the staging area
5. Committing the changes and pushing the files to the github repository

**Step 3.1.1:** Cloning a github repository

* Clone a github repository in the local machine **(Refer to demo 2.14 in LMS)**



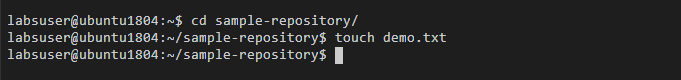
* Go to the repository folder using the **cd** command



**Step 3.1.2:** Adding a file to the cloned repository

* Use the following command to add a **demo.txt** file:

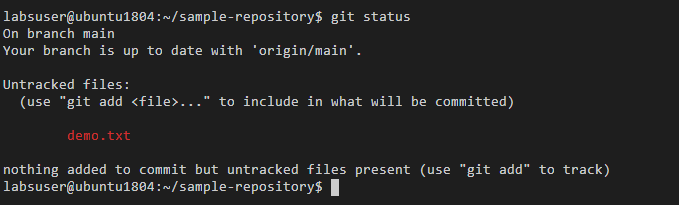
**$ touch demo.txt**



**Step 3.1.3:** Checking the status of the repository

* Use the following command to check the status of the repository:

**$ git status**



**Note:** It shows that you are on the branch master, and an **untracked file,** which is available in the repository. An untracked file is the one which is not added to Git and Git is not able to track it.

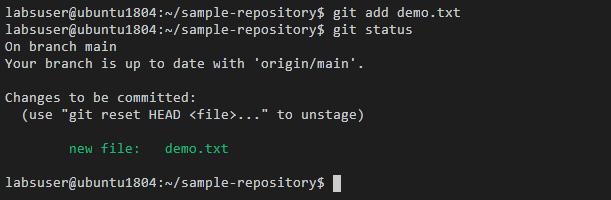
**Step 3.1.4:** Adding the file to the staging area

* Use the following command to add the file to the staging area:

**$ git add demo.txt**

* Check the repository status again

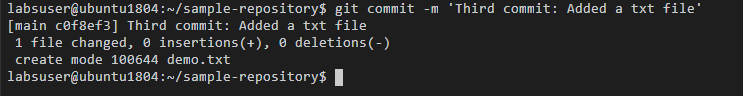
**$ git status**



**Step 3.1.5:** Committing the changes and pushing the files to the github repository

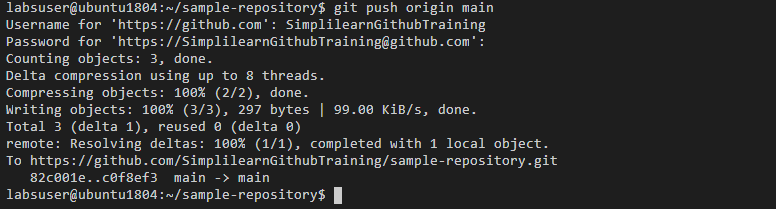
* Use the following command to commit the changes in the repository:

**$ git commit -m ‘Third commit: Added a txt file’**



* Use the following command to push the file to the main branch:

**$ git push origin main**



3.2 Tracking File Changes

This section will guide you to:

* Tracking the changes in a file

This lab has four subsections, namely:

1. Updating a file from the repository
2. Tracking the changes in the file
3. Adding the file to the staging area
4. Comparing git commits to track file changes

**Please note**: Make sure you have successfully completed the assisted practice 3.1.

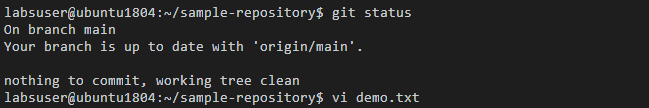
**Step 3.2.1:** Updating a file from the repository

* Check the status of the repository

**$ git status**

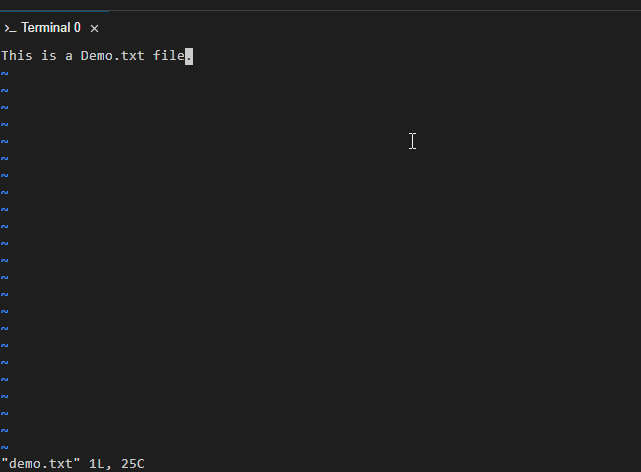
* Open a file from the local repository in **vi editor** and make some changes in the file

**$ vi demo.txt**



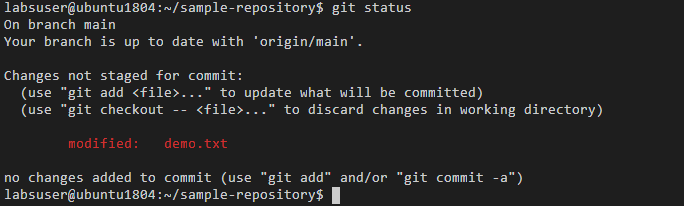
* Add some sample content in the file **demo.txt**, for instance:

**This is a Demo.txt file**



* Save the file and exit by pressing **esc** key and **shift+: wq**
* Check the status of the repository again

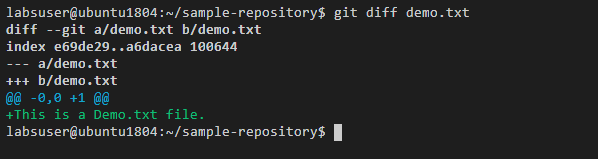
**$ git status**



**Step 3.2.2:** Tracking the changes in the file

* Use the following command to compare the file in the working directory with its last staging area:

**$ git diff demo.txt**



**Note:** The **+** statement is showing the change in the file.

**Step 3.2.3:** Adding the file to the staging area

* Use the **git add** command to add the file to the staging area

**$ git add demo.txt**

* Use the **git diff** command to track any changes

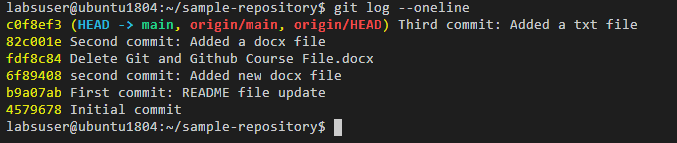
**$ git diff demo.txt**



**Step 3.2.4:** Comparing git commits to track file changes

* Use the following command to check the recent log of commits with **--oneline** flag

**$ git log --oneline**



* Use the **git commit** command to commit the changes from the staged area

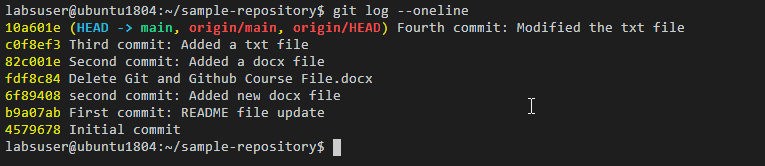
**$ git commit -m "Fourth commit: Modified the txt file"**

* Use the **git push** command to push the file to the main branch

**$ git push origin main**

* Use the **git log** command again to see the latest commit

**$ git log --oneline**



3.3 Rolling Back to Previous Commits

This section will guide you to:

* Rollback to a previous commit using reset command
* Rollback to a previous commit using revert command

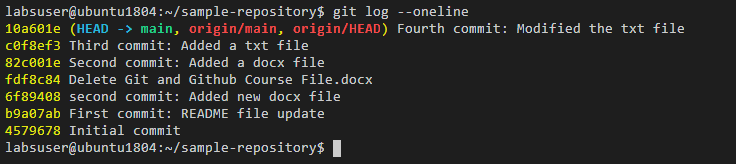
This lab has two subsections, namely:

1. Rolling back to a previous commit using **reset** command
2. Rolling back to a previous commit using **revert** command

**Please Note**: Make sure you have successfully completed assisted practice 3.2

**Step 3.3.1:** Rolling back to a previous commit using **reset** command

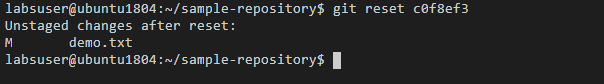
* In the terminal, use the **git log --oneline** command to check the recent commits log



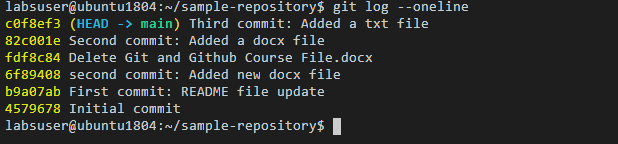
* Use the following command to rollback to a particular commit

**$ git reset COMMIT\_VALUE**

where COMMIT\_VALUE is the number shown in front of each commit



* Use the **git log** **–oneline** command to again check the recent commits log



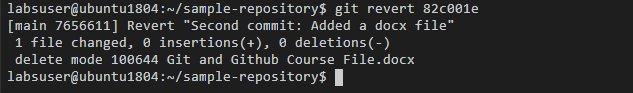
**Note:** Notice that the Fourth commit is deleted and the last commit showing on the log is Third commit.

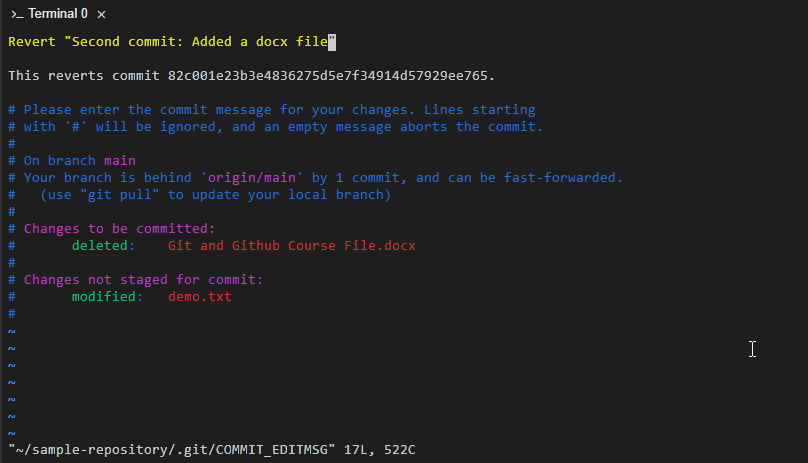
**Step 3.3.1:** Rolling back to a previous commit using **revert** command

* Use the following command to rollback to a particular commit

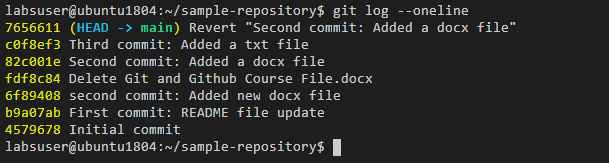
**$ git revert COMMIT\_VALUE**

where COMMIT\_VALUE is the number shown in front of each commit





* Use the **git log** command to again check the recent commits log



**Note:** Notice that a new commit is showing that reflects the contents before the last commit.

3.4 Cleaning the Working Directory

This section will guide you to:

* Clean the working directory

This lab has one subsection, namely:

1. Cleaning the working directory

**Please note**: Make sure you have successfully completed the assisted practice 3.3

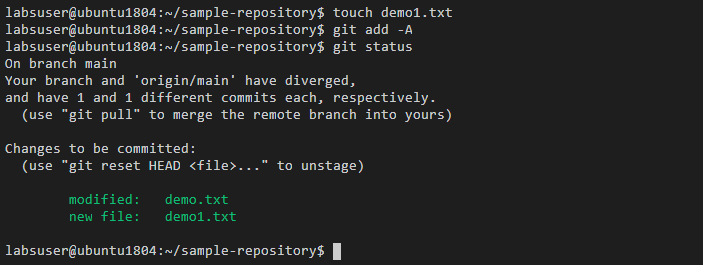
**Step 3.4.1:** Cleaning the working directory

* In the terminal, Add a file in the repository and keep it in staging area

**$ touch demo1.txt**

**$ git add -A**

**$ git status**



* Use the following command to perform dry run on cleaning the untracked files

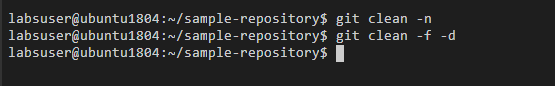
***$ git clean -n***

* Use the following command to force clean the untracked files

***$ git clean -f***

* Use the following command to clean the untracked directories

***$ git clean -f -d***



3.5 Adding Changes to the Last Commit

This section will guide you to:

* Add changes to the last commit

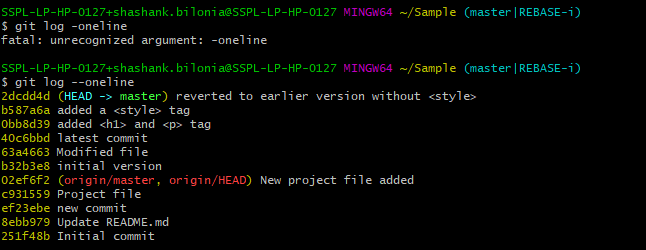
This lab has six subsections, namely:

1. Checking the list of files in the repository
2. Opening the file in the vi editor to implement the necessary changes
3. Executing **git status** to check the status of the repository**.** It shows that the file is being modified
4. Using **git add** command to add the files to the staging area
5. Executing **git commit –amend** to modify the most recent commit
6. Executing **git log –oneline** to check the recent commit with the modified message

**Please note**: Make sure you have successfully completed the assisted practice 3.4

**Step 3.5.1:** Checking the log of the recent commits

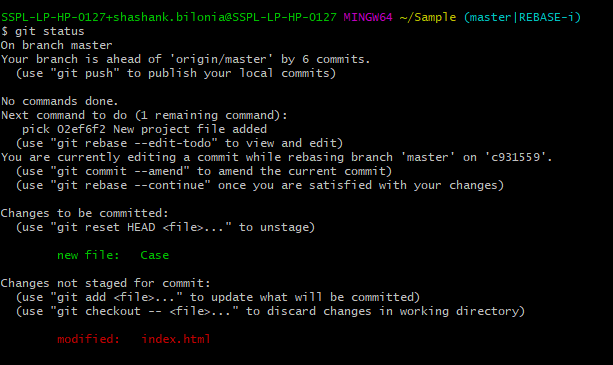
* Execute **git log --oneline** to know the latest commit



**Step 3.5.2:** Opening a file in the **vi editor** to make necessary changes



**Step 3.5.3:** Executing **git status** to check the status of the repository**.** It shows that the file is being modified



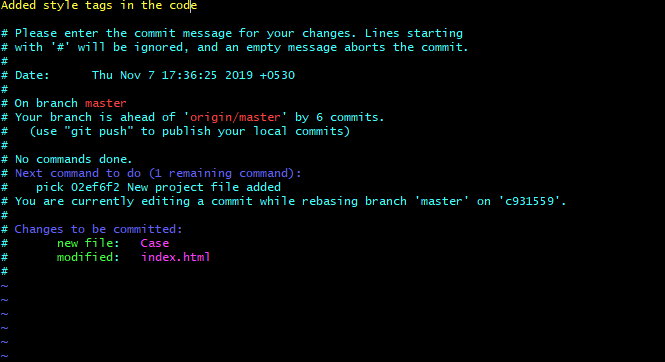
**Step 3.5.4:** Using **git add .** command to add the files to the staging area

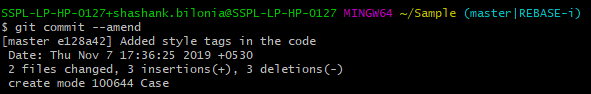


**Step 3.5.5:** Executing **git commit –amend** to modify the most recent commit

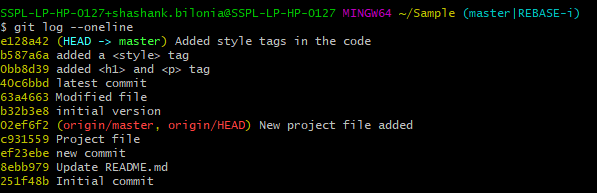


* Editing the commit message in the text editor that shows up after executing the **“amend”** command. For instance, the commit message can be **“Added style tags in the code”**





**Step 3.5.6:** Executing **git log –oneline** to check the recent commit with the modified message



3.6 Deleting Files in Git

This section will guide you to:

* Delete a file in Git

This lab has five subsections, namely:

3.6.1 Checking the list of files in the repository

3.6.2 Removing a file from the repository

3.6.3 Checking the list of files again

3.6.4 Executing git status, to show that the removal of the file is still in staging and needs to be committed to the repository

3.6.5 Committing the changes

**Please note**: Make sure you have successfully completed the assisted practice 3.5

**Step 3.6.1:** Checking the list of files in the repository

* In the terminal, execute the following command to list all the files

***$ git ls-files***



**Step 3.6.2:** Removing a file from the repository

* Use the following commandto remove the file from the repository

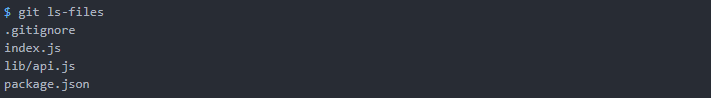
***$ rm <YourFileName>***



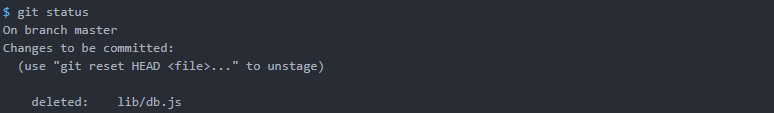
**Step 3.6.3:** Checking the list of files again.

* Use git ls command to check that the file is removed from the list

***$ git ls-files***



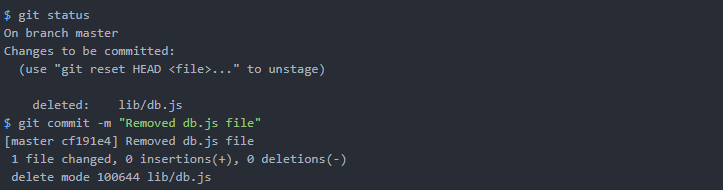
**Step 3.6.4:** Executing **git status**, where you will notice that the removal of the file is still in staging and needs to be committed to the repository



**Step 3.6.5:** Committing the changes

* Execute the **git add .** Command
* Use git commit command to commit the changes

**$ git commit -m “Removed file”**



3.7 Ignoring Files in Git

This section will guide you to:

* Create .gitignore file

This lab has two subsections, namely:

3.7.1 Creating the .gitignore file

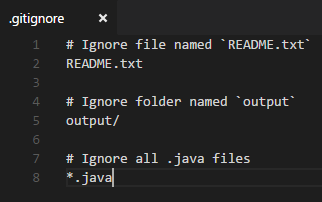
3.7.2 Checking the git status

**Please note**: Execute the following commands in a sample Git repository that you had created in the previous demos.

**Step 3.7.1:** Creating the **.gitignore** file. Also, define the rules that you need to keep and save the file.

Example:

1. For readme in Github, I will only need a README.md file. So, I have ignored if there is a README.txt file.
2. Also, I don’t need a folder called “output.”
3. And, there will be no Java files added to the repository. So, I have ignored all of the Java files.



* In the terminal, execute the following commands to create the .gitignore file

***touch .gitignore***

***echo “README.txt”>>.gitignore***

***echo “output/”>>.gitignore***

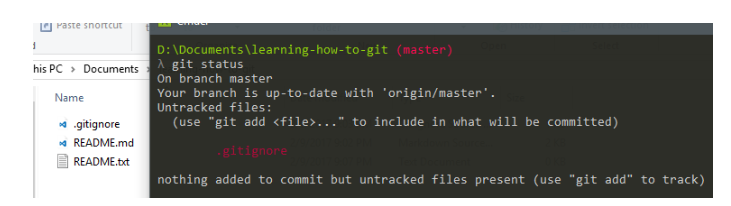
***echo “.java”>>.gitignore***

**Step 3.7.2:** Checking the git status. Now, if you try to add those files to the Git repository, it will be ignored.

* Use the following commands to create the .gitignore file

***git status***

Notice that I try to add README.txt, but it is not listed when I try to check it with “git status” command.



3.8 Renaming Files in Git

This section will guide you to:

* Rename files in Git

This lab has three subsections, namely:

3.8.1 Executing **git mv <old file name> <new file name>** to rename the file

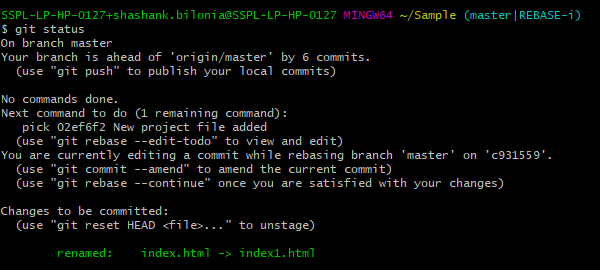
3.8.2 Checking the status of the repository

3.8.3 Commit the changes

**Please note**: Execute the below commands in a sample git repository that you have created in the previous demos.

**Step 3.8.1:** In the terminal, execute **git mv <old file name> <new file name>** to rename the file



**Step 3.8.2:** Checking the status of the repository by executing **git status**

**Step 3.8.3:** Commit the changes by executing **git commit -m “file renamed”**