ATTACK DEFENSE

by PentesterAcademy

Name	Git: Learn Basics with Git Cola
URL	https://attackdefense.com/challengedetails?cid=2035
Type	DevOps Basics: Version Control System

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

Git is a distributed version control system (VCS) for managing the software source code. Git can be hosted on the cloud or on the local system.

A Kali machine and a Gitlab instance are provided in this lab. Git cola, an open-source GUI based Git client is installed on the Kali machine and a repository is hosted on the Gitlab server.

The Gitlab instance is accessible on hostname 'gitlab' and uses the following credentials:

Username	Password
root	welcome123

The user is provided with access to Kali GUI and Gitlab web UI.

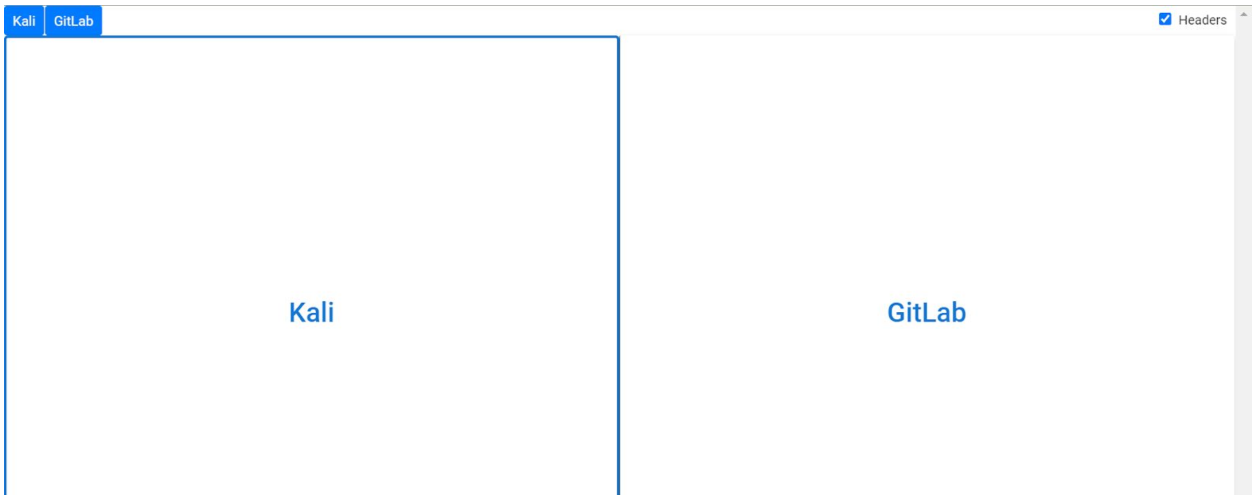
Objective: Follow the manual and learn how to use the Git with Git Cola!

Instructions:

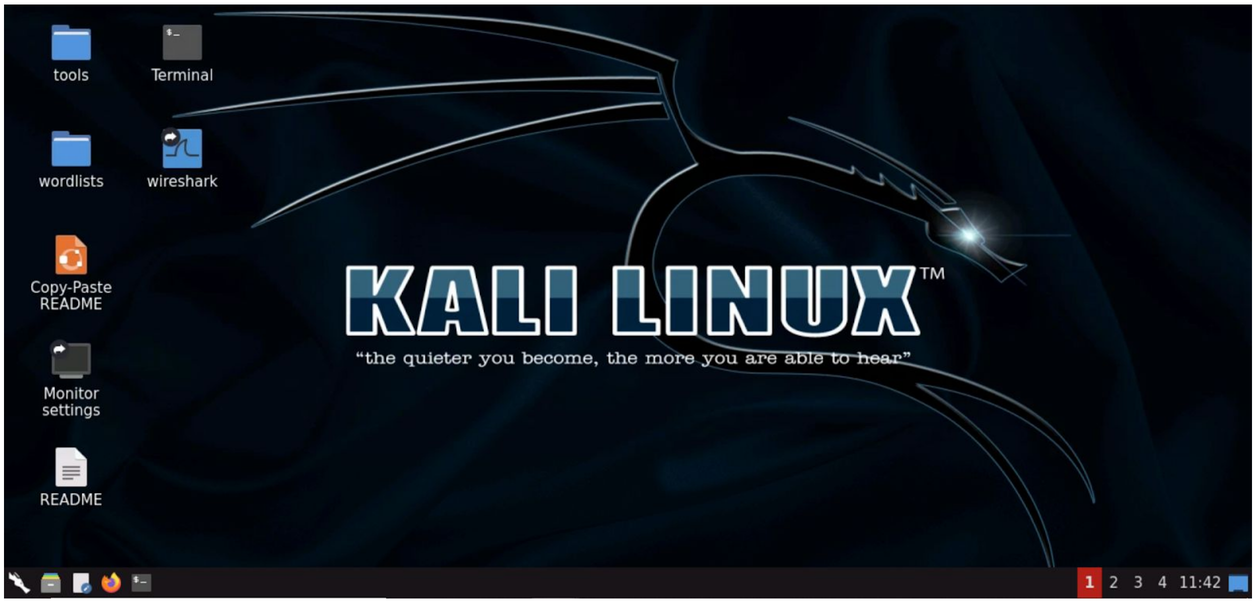
- The Gitlab instance is accessible with the name "gitlab".

Lab Setup

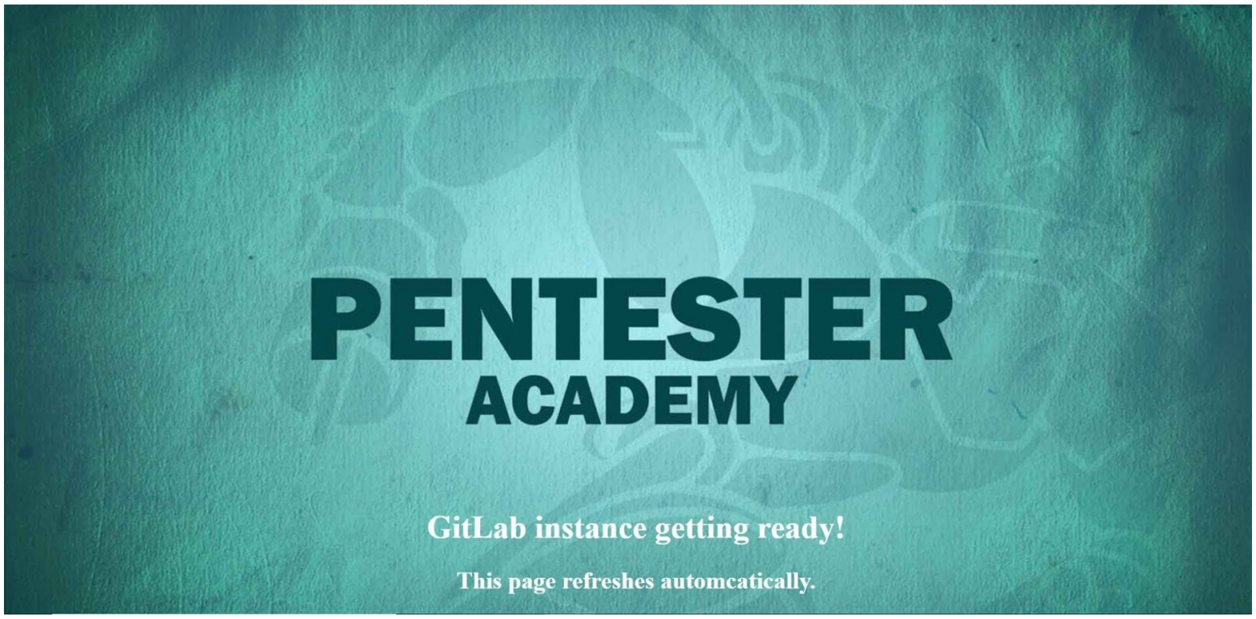
On starting the lab, the following interface will be accessible to the user.



On choosing (clicking the text in the center) left panel, **a Kali GUI instance** will open in a new tab.



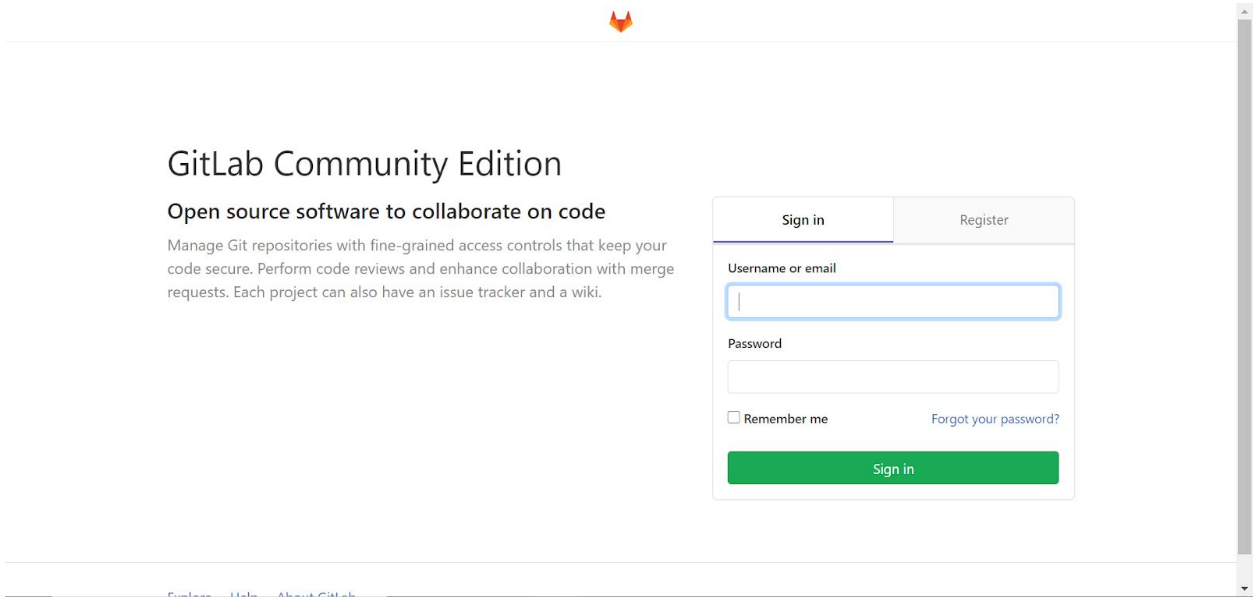
Similarly on selecting the right panel, a web UI of **Gitlab instance** will open in a new tab. However, if the Gitlab instance is not ready then one will see the following page.


PENTESTER ACADEMY

GitLab instance getting ready!

This page refreshes automcatically.

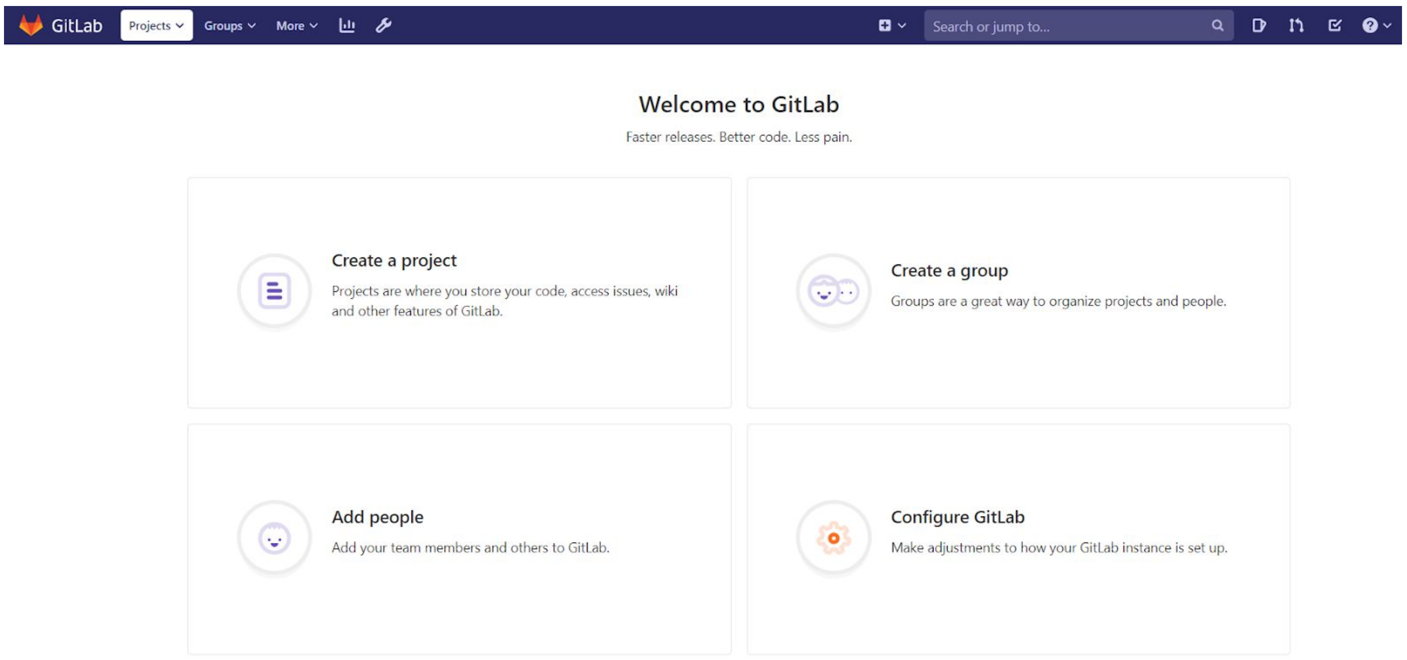
This page reloads automatically and shows the intended page as soon as the setup is ready. The loaded Gitlab page will look like this:


GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

The public projects of the root user can be checked at http://<gitlab-url>/root/



Welcome to GitLab

Faster releases. Better code. Less pain.

Create a project

Projects are where you store your code, access issues, wiki and other features of GitLab.

Create a group

Groups are a great way to organize projects and people.

Add people

Add your team members and others to GitLab.

Configure GitLab

Make adjustments to how your GitLab instance is set up.

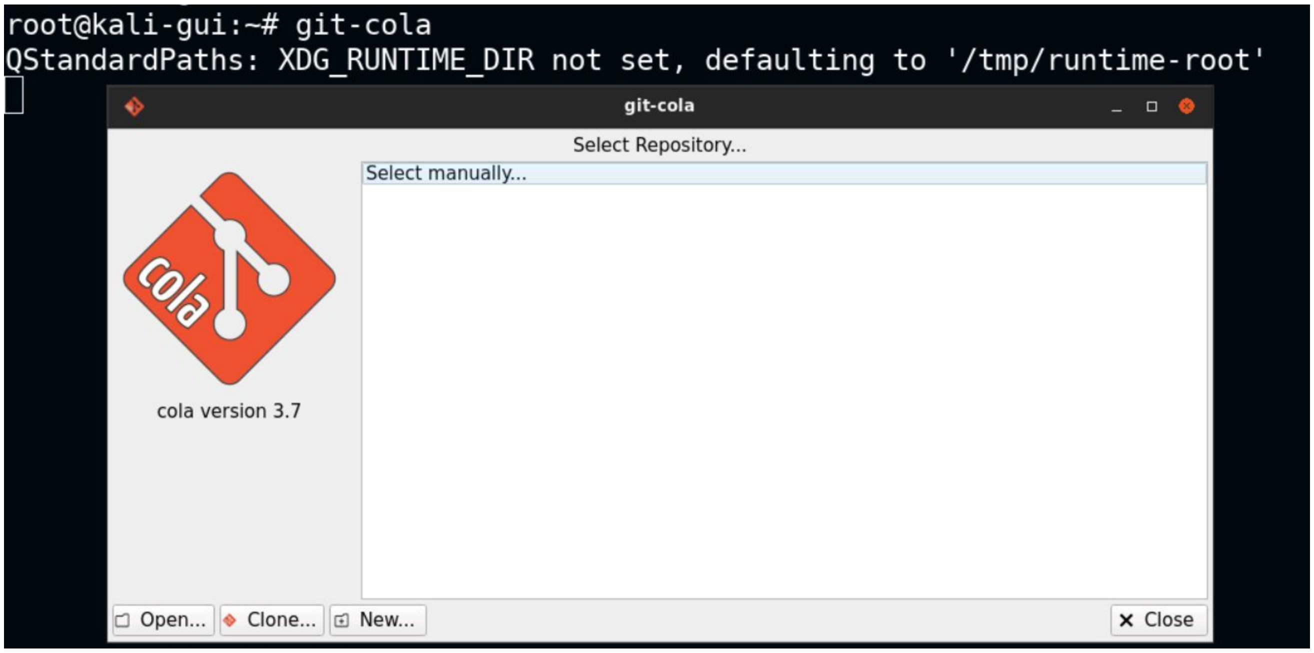
Now, both interfaces are ready and the user can start with the lab.

Solution

1. Create a Repository

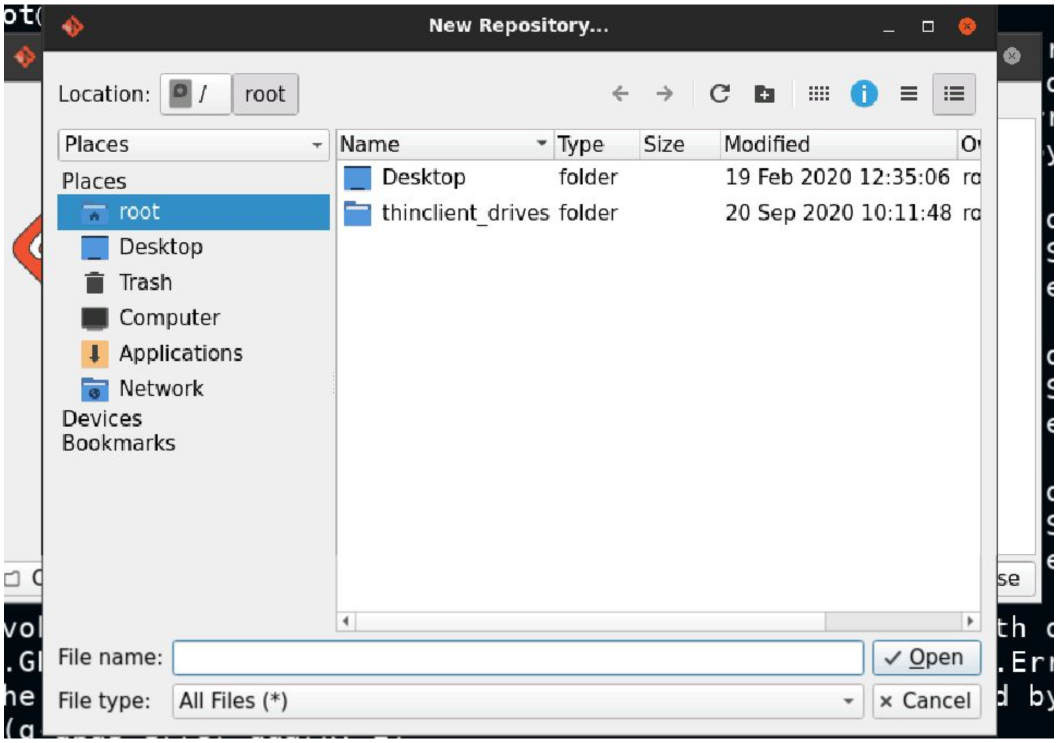
Step 1: Start Git-cola by opening a terminal and type the following command

Command: git-cola



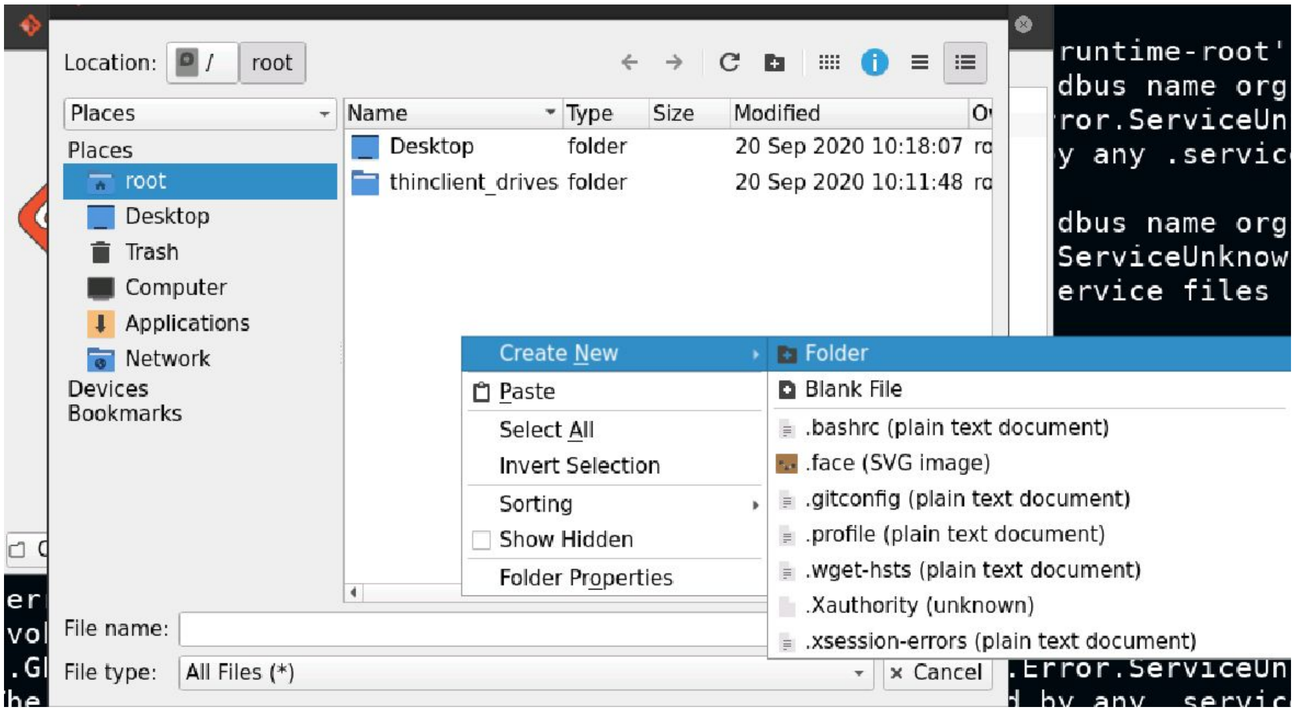
```
root@kali-gui:~# git-cola
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Click on the "New" button

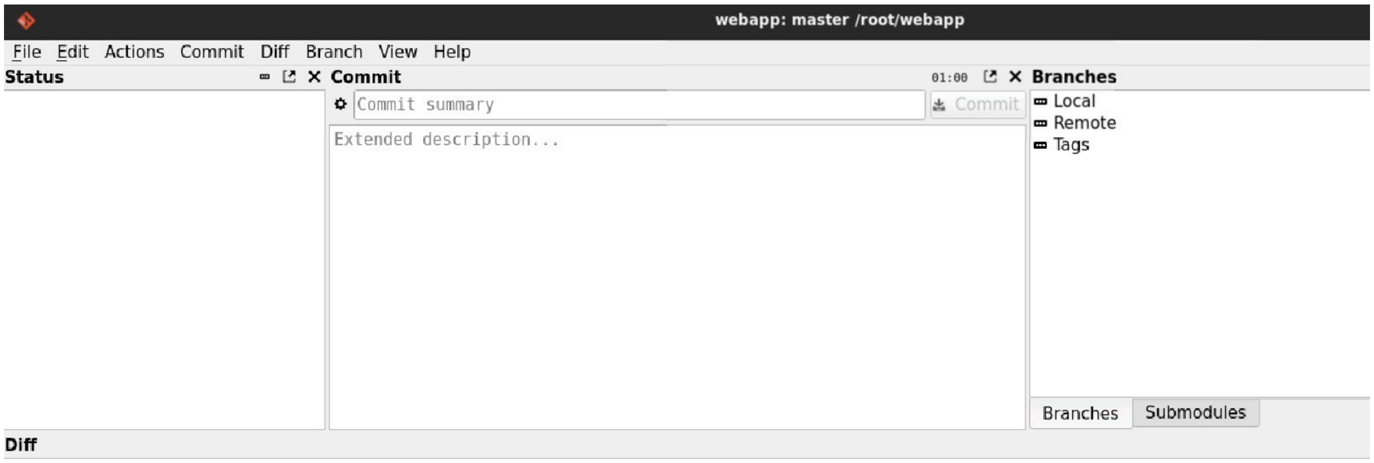


Name	Type	Size	Modified
Desktop	folder		19 Feb 2020 12:35:06
thinclient_drives	folder		20 Sep 2020 10:11:48

Right-click on the page and create a new folder named “webapp”

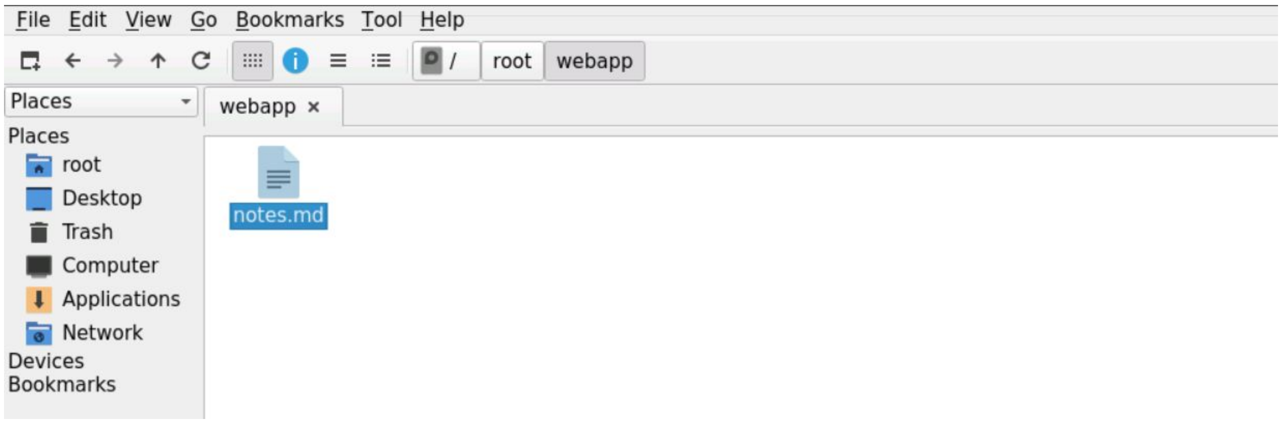


Select the newly created folder and press “Open” button.

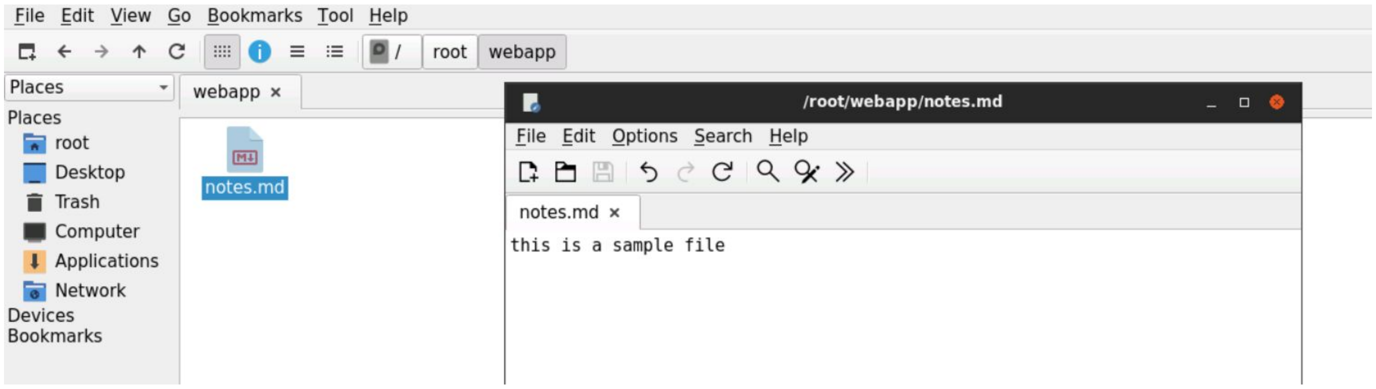


2. Add a new file in the repository

Step 1: Open the webapp local repository's directory and create a new file as "notes.md"

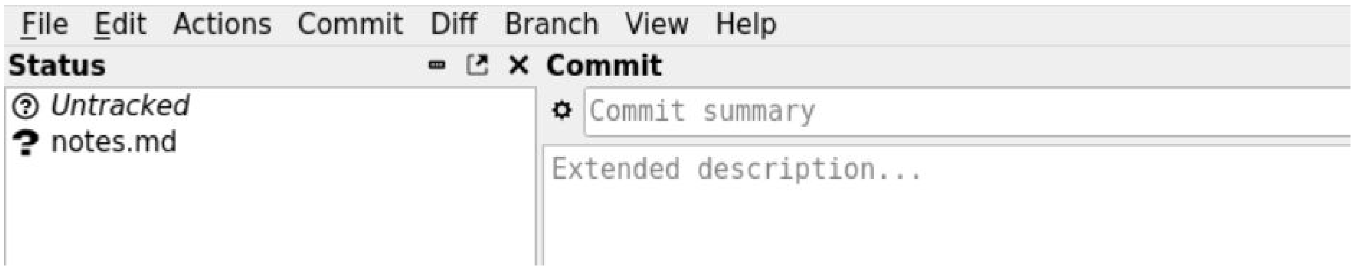


Step 2: Open the file to enter data and save the file.



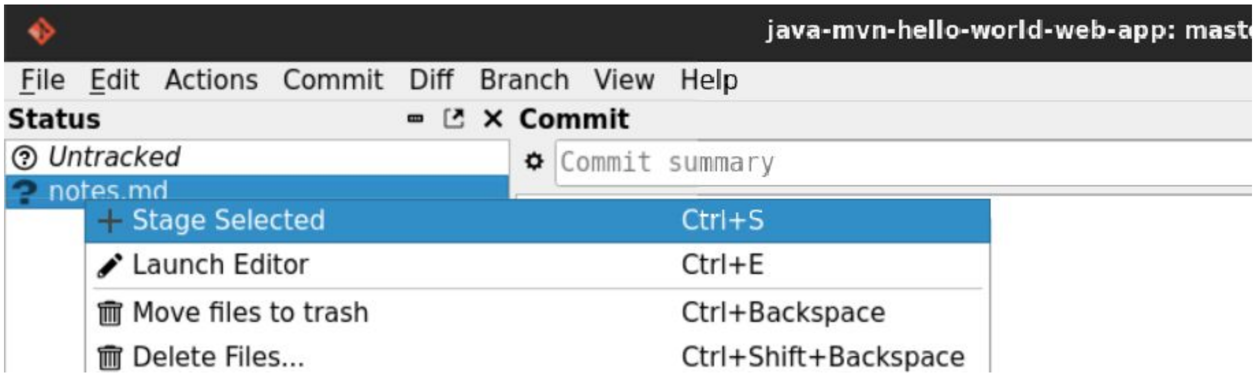
```
this is a sample file
```

Step 3: Check the Git cola and press CTRL + R to refresh the repo

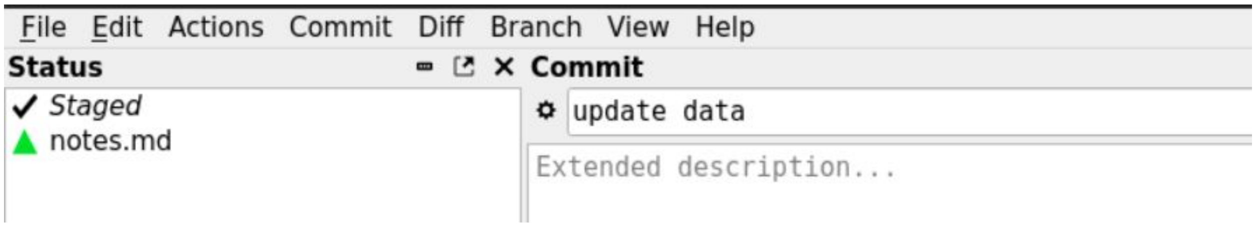


The notes.md file has to be tracked in order to upload them to the remote repository.

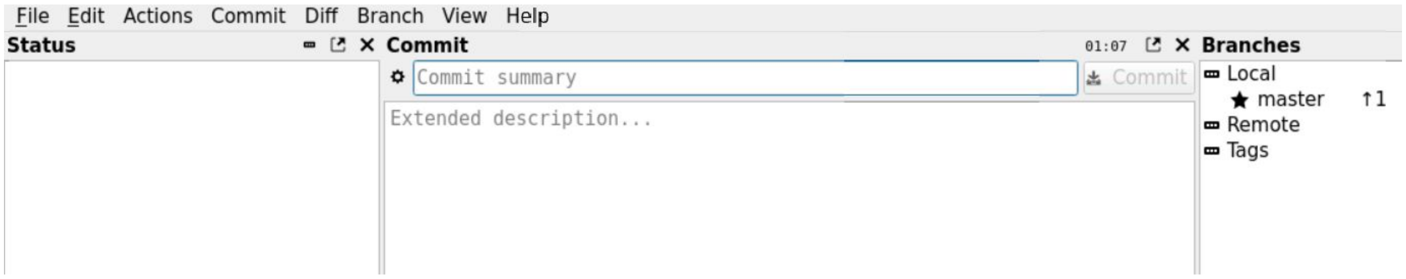
Step 4: Right-click on the “notes.md” and select “Stage Selected” to track the file


java-mvn-hello-world-web-app: mast
File Edit Actions Commit Diff Branch View Help
Status
Commit
Untracked
notes.md
Commit summary
Stage Selected Ctrl+S
Launch Editor Ctrl+E
Move files to trash Ctrl+Backspace
Delete Files... Ctrl+Shift+Backspace

Step 5: Enter a description of the notes.md in “Commit Summary” text box



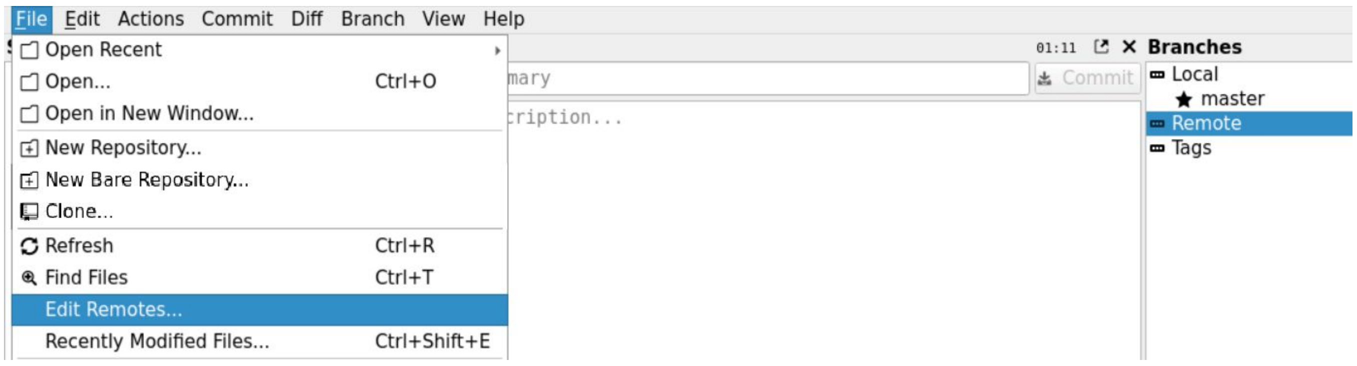
Click on the Commit Drop-down menu and select Commit.



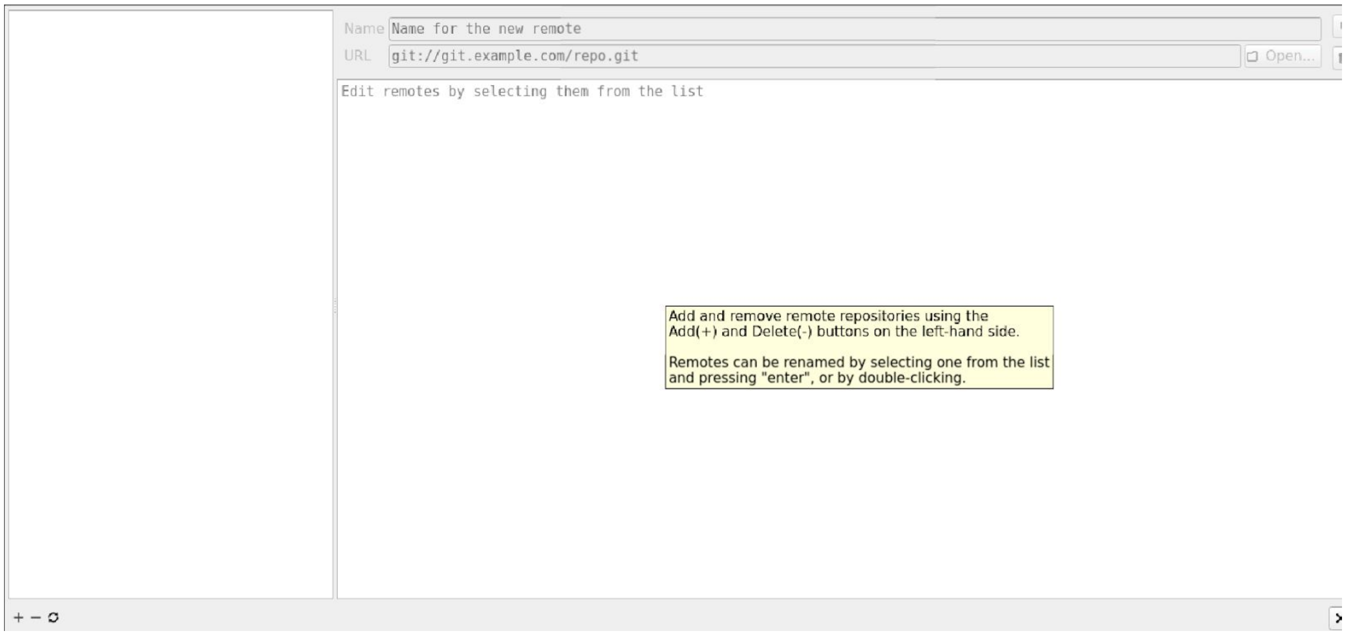
The file has been committed in the local repository

3. Push the data to the remote repository

Step 1: Add a remote repository by selecting the “edit Remote” under File option



Click on the option



Name for the new remote

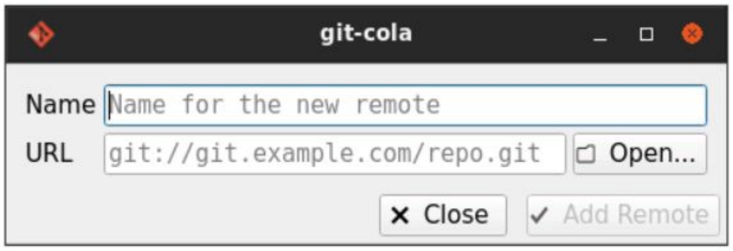
git://git.example.com/repo.git

Edit remotes by selecting them from the list

Add and remove remote repositories using the Add(+) and Delete(-) buttons on the left-hand side.

Remotes can be renamed by selecting one from the list and pressing "enter", or by double-clicking.

Step 2: Click on the + (plus) button.

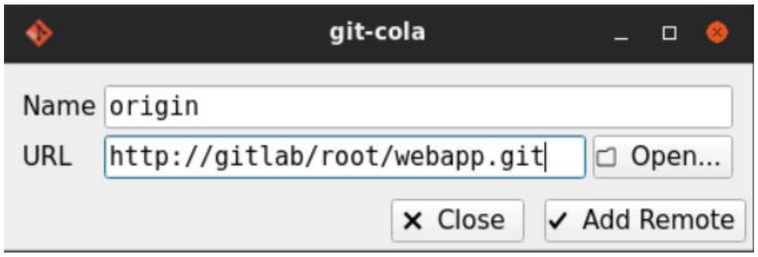


Name for the new remote

git://git.example.com/repo.git

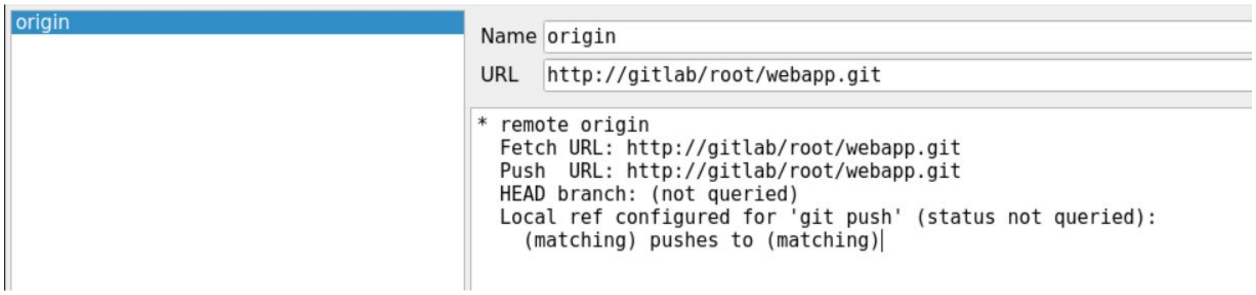
Step 3: Enter the Remote Repo name and URL

Remote Repo Name: origin
Remote Repo URL: http://gitlab/root/webapp.git



Name: origin
URL: http://gitlab/root/webapp.git

Click on “Add Remote”

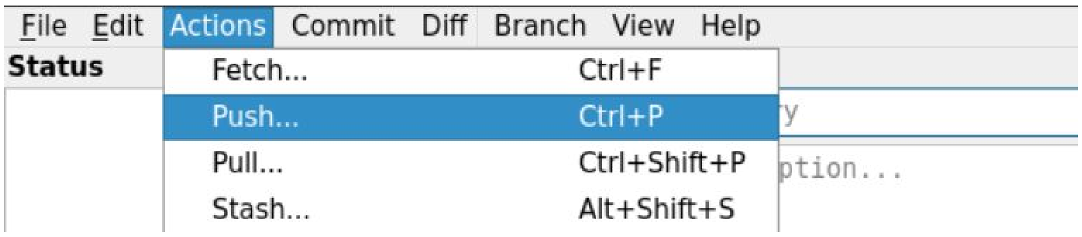


Name: origin
URL: http://gitlab/root/webapp.git

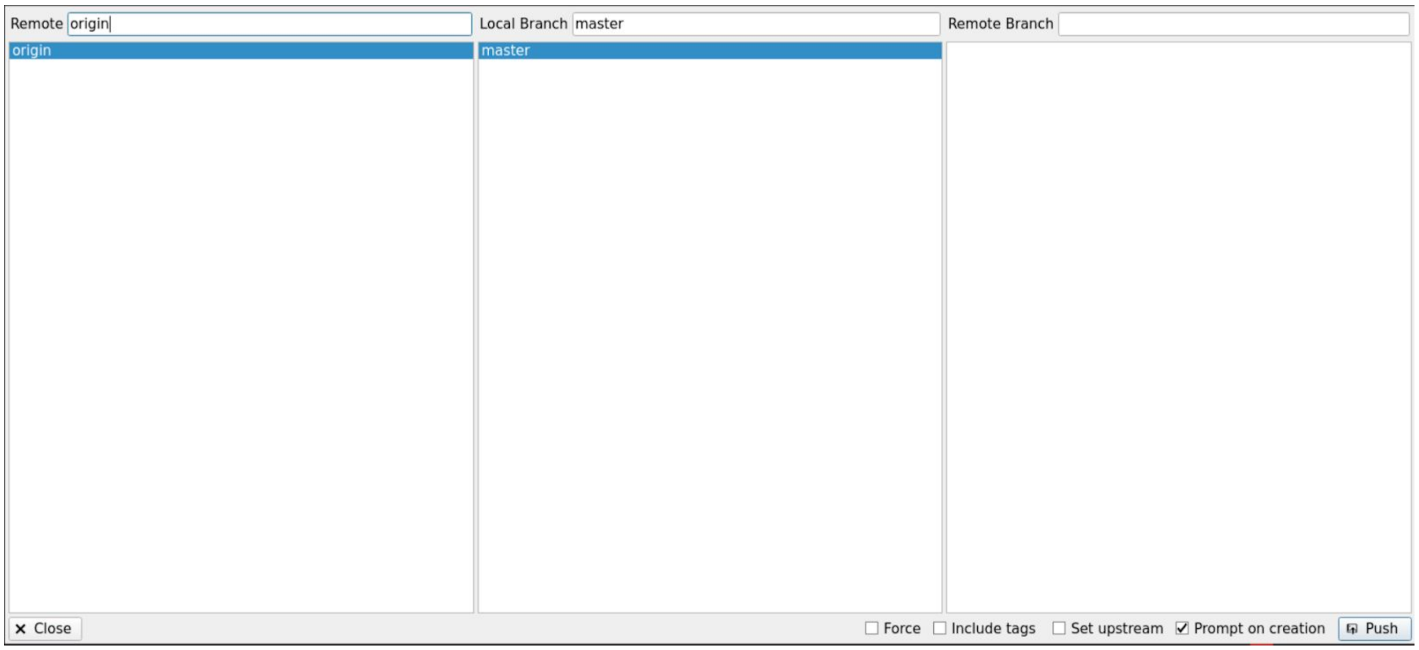
```
* remote origin
  Fetch URL: http://gitlab/root/webapp.git
  Push  URL: http://gitlab/root/webapp.git
  HEAD branch: (not queried)
  Local ref configured for 'git push' (status not queried):
    (matching) pushes to (matching)
```

Close the ‘Edit Remotes’ window.

Step 4: Click on the action Drop-down button



Click on the Push.



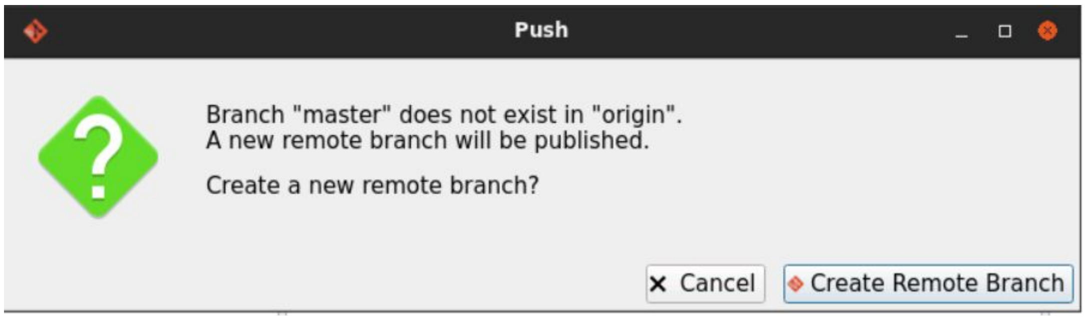
Remote origin | Local Branch master | Remote Branch

origin

master

Close | Force | Include tags | Set upstream | ☑ Prompt on creation | Push

Click on the “Push” button

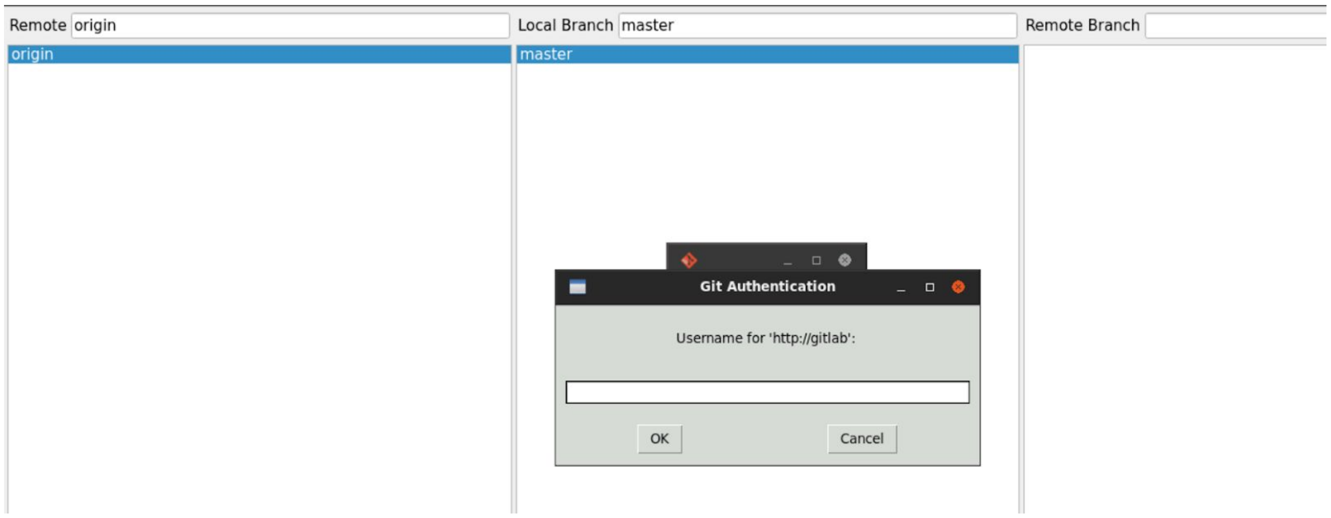


Push

Branch "master" does not exist in "origin".
A new remote branch will be published.

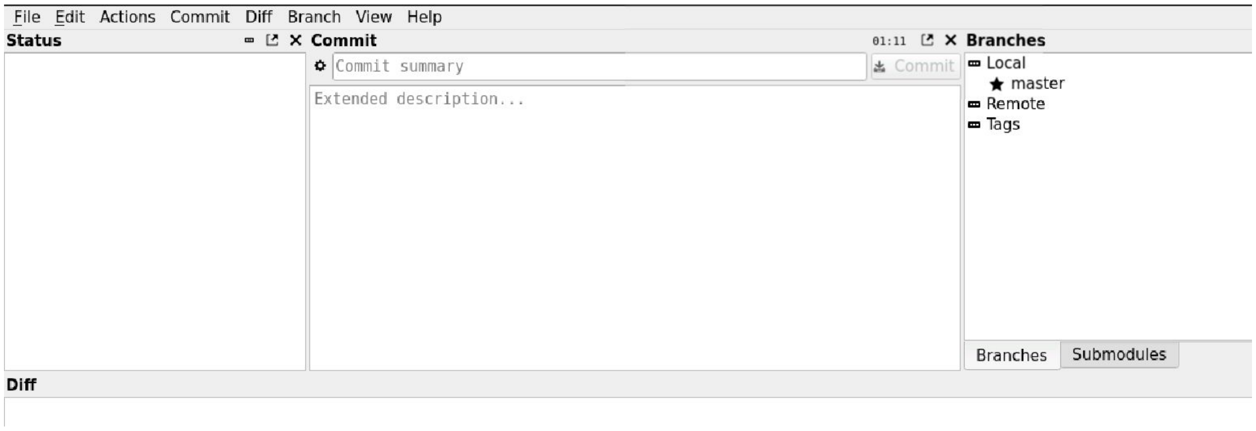
Create a new remote branch?

Click on “Create Remote Branch”



Username for 'http://gitlab':

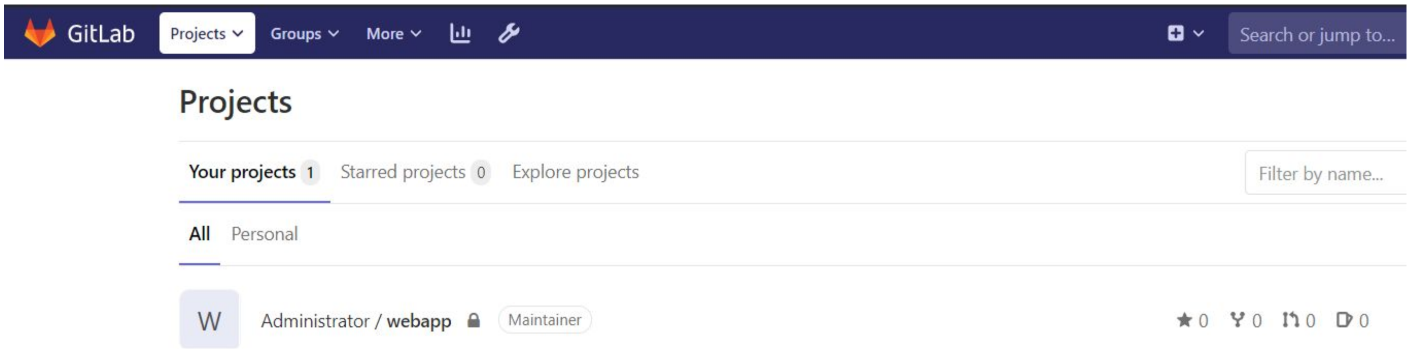
Enter the username and password in the prompt. The credentials have been provided in the challenge description.



The push has been successful. Navigate to the Gitlab Website and check the changes. Login using the credentials provided in the challenge description.

Credentials:

- **Username:** root
- **Password:** welcome@123



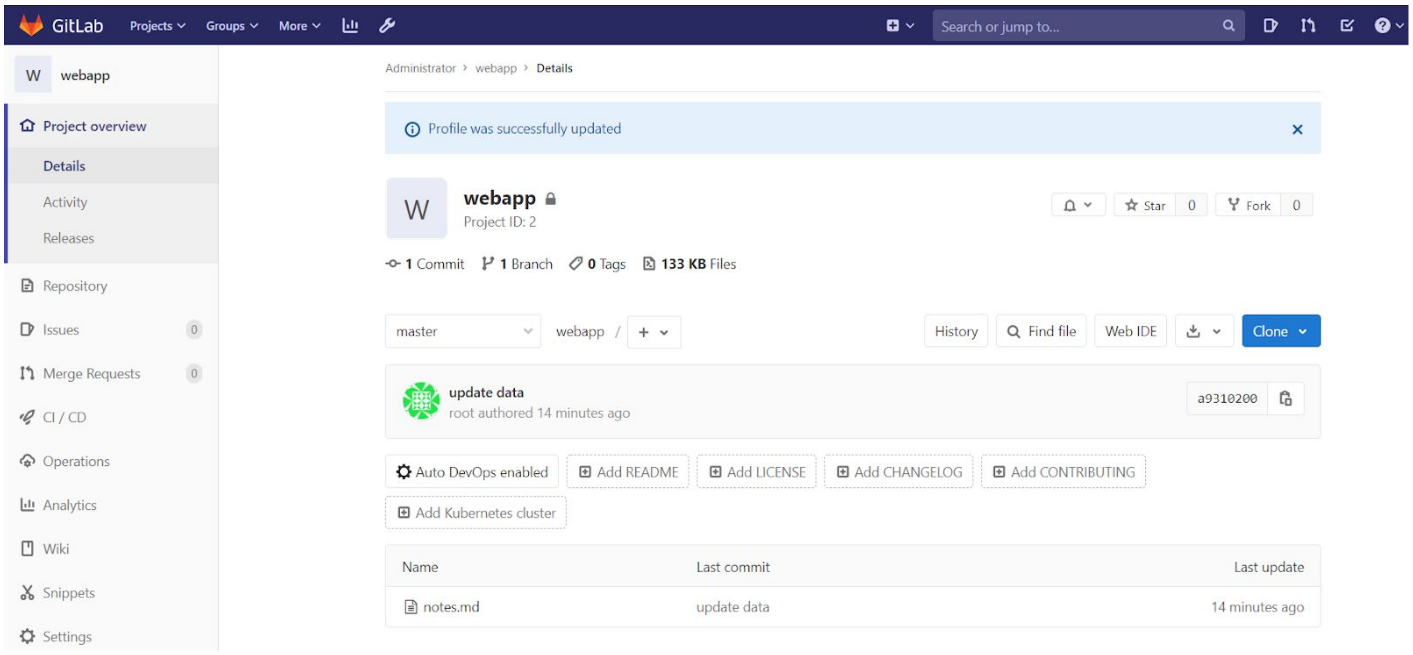
Projects

Your projects 1	Starred projects 0	Explore projects	Filter by name...

All Personal

Administrator / webapp Maintainer ★ 0 0 0 0

Click on the “webapp” repository.



Profile was successfully updated

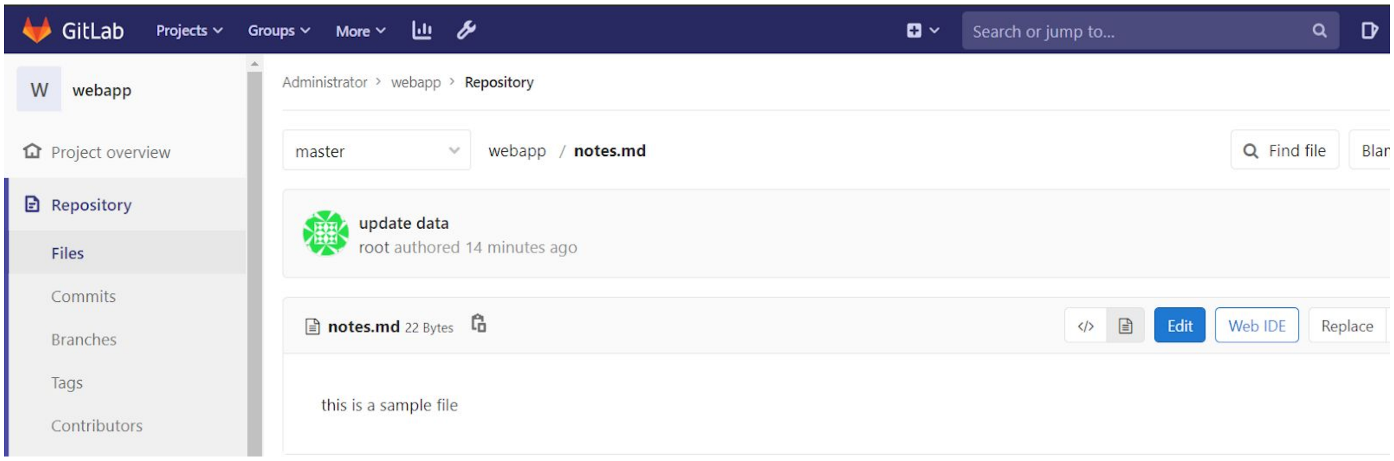
webapp
Project ID: 2

1 Commit 1 Branch 0 Tags 133 KB Files

update data
root authored 14 minutes ago a9310200

Name	Last commit	Last update
notes.md	update data	14 minutes ago

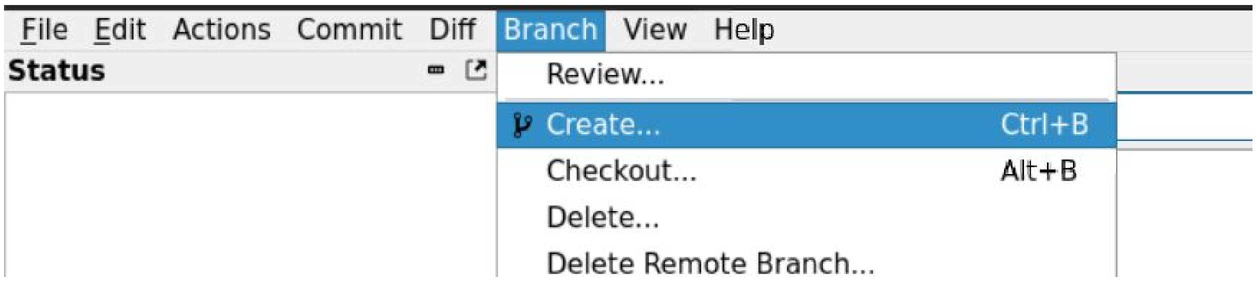
The notes.md is present in the remote repository, click on the notes.md and check its contents



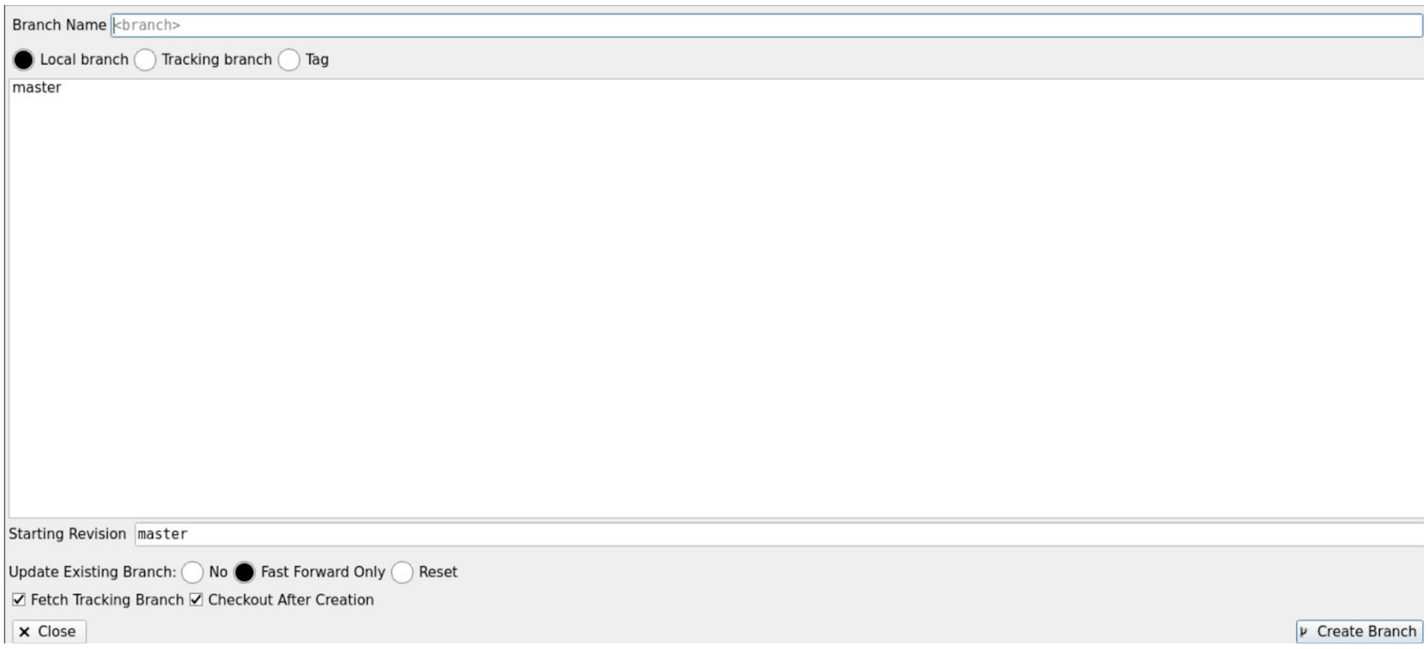
this is a sample file

4. Create a new branch

Step 1: Navigate back to the kali instance and click on the Branch drop-down



Step 2: Click on the Create button.


Branch Name <branch>

[✓ Local branch] Tracking branch Tag

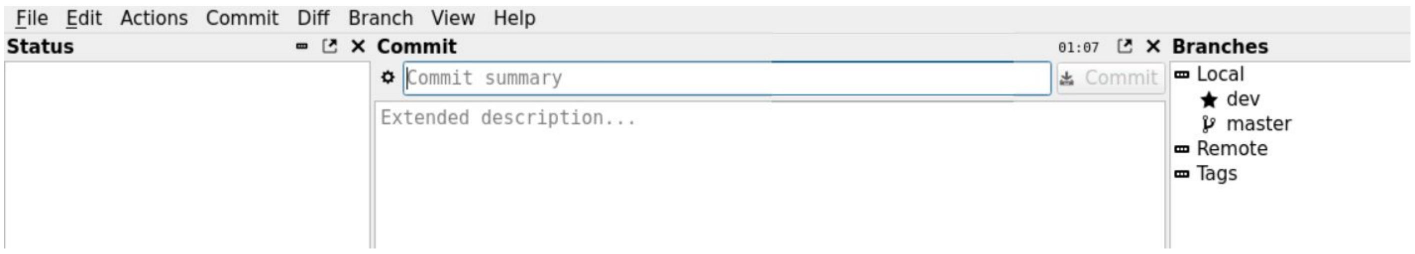
master

Starting Revision master

Update Existing Branch: No [✓ Fast Forward Only] Reset

☑ Fetch Tracking Branch ☑ Checkout After Creation

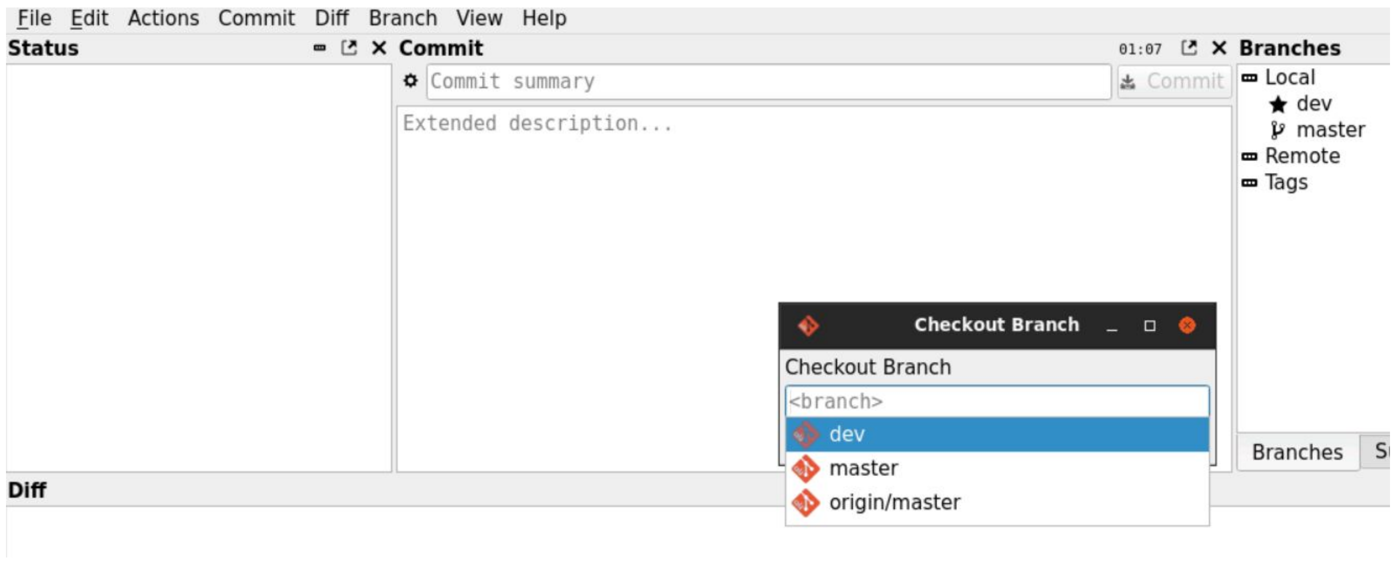
Step 3: Enter “dev” in branch name and click on “Create Branch” located at the bottom right corner of the page.



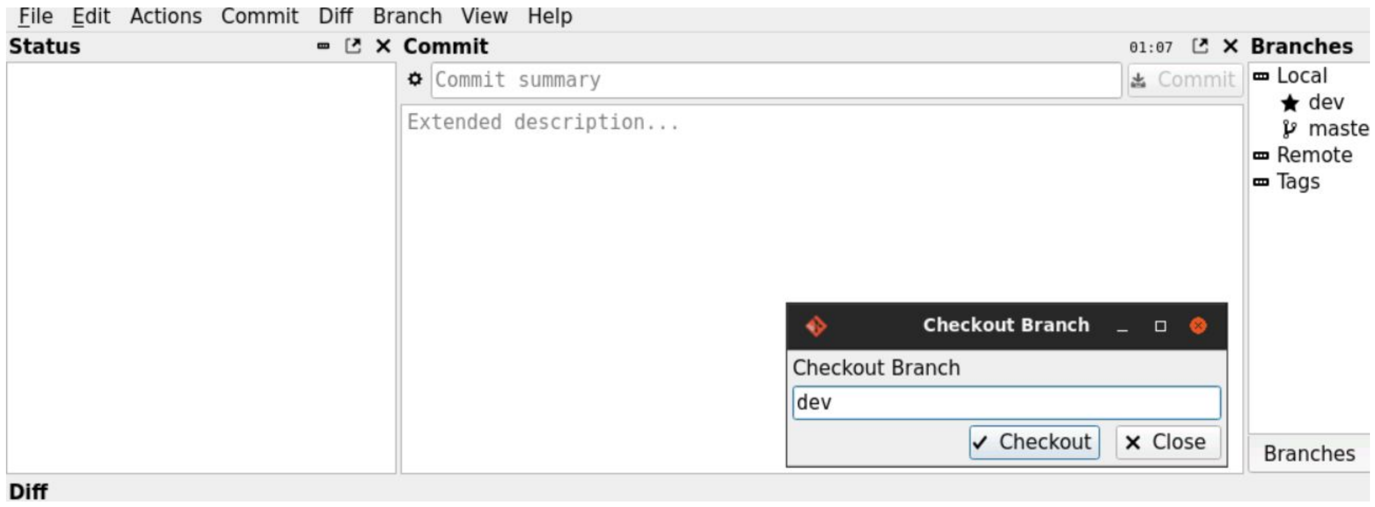
The dev branch has been created successfully.

5. Switch to Dev branch

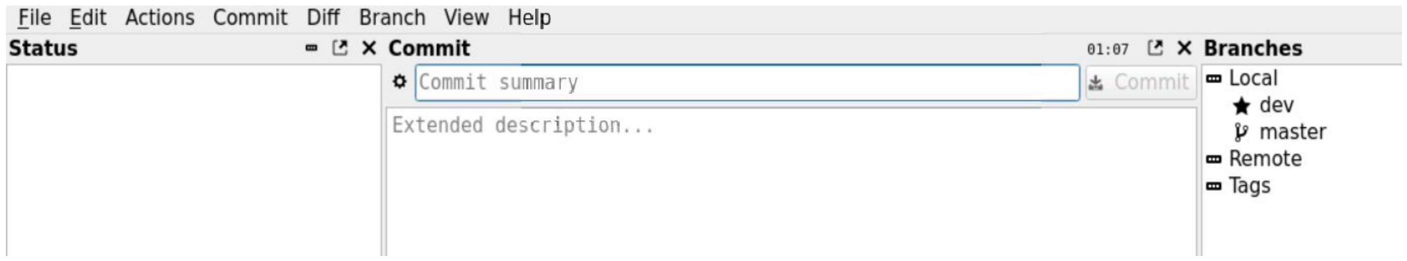
Step 1: Click on the Branch drop-down and select Checkout



Step 2: Select the dev branch



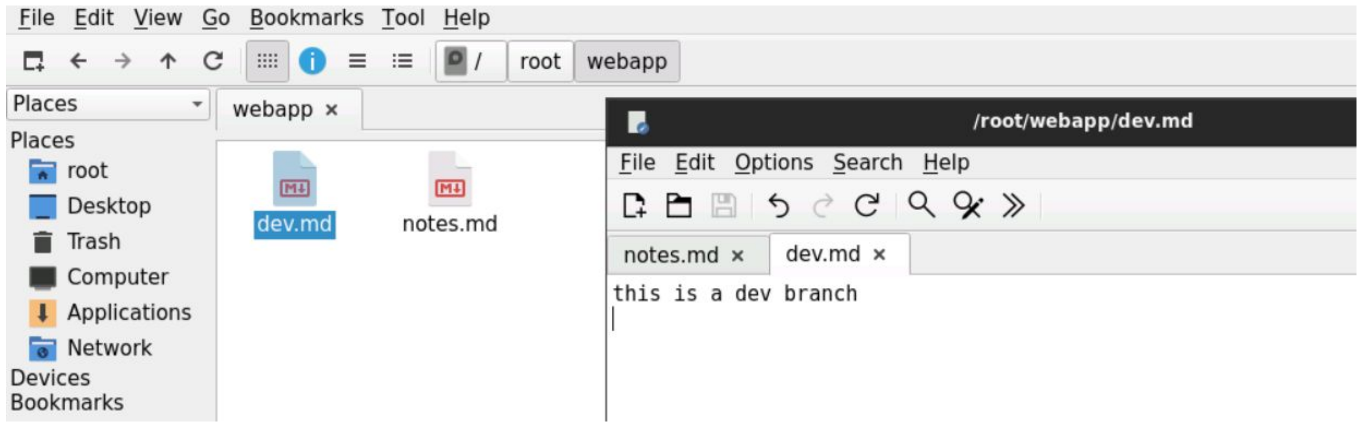
Click on the Checkout button.



The dev branch has been selected successfully.

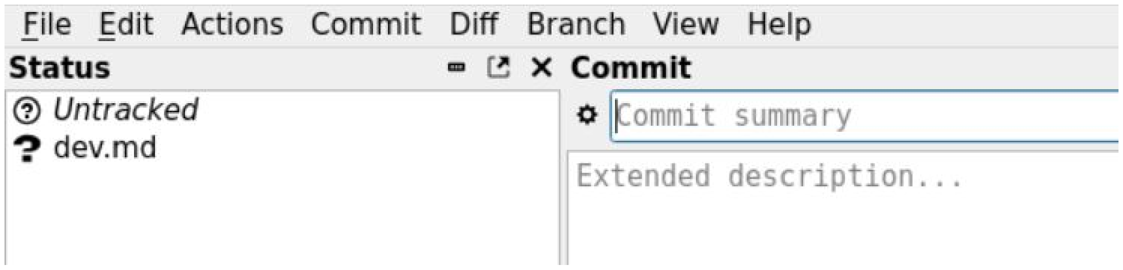
6. Create a new file in the dev branch

Step 1: Open the webapp directory and create a new file with any content in the file and save the file as "dev.md"

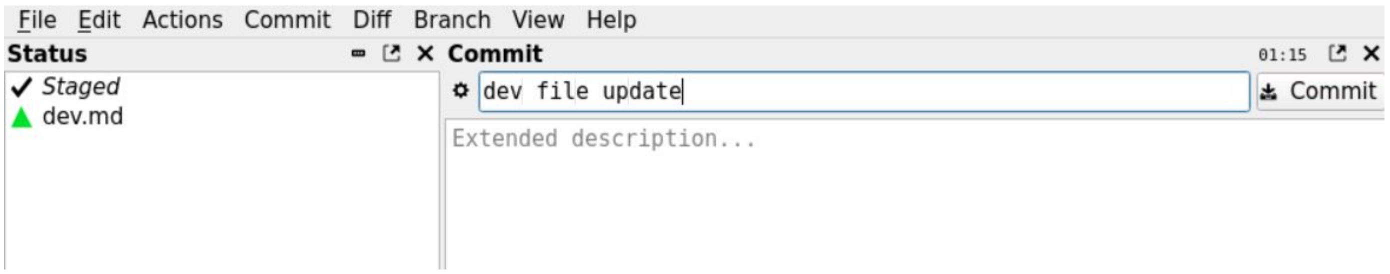


```
this is a dev branch
```

Step 2: Check the git cola



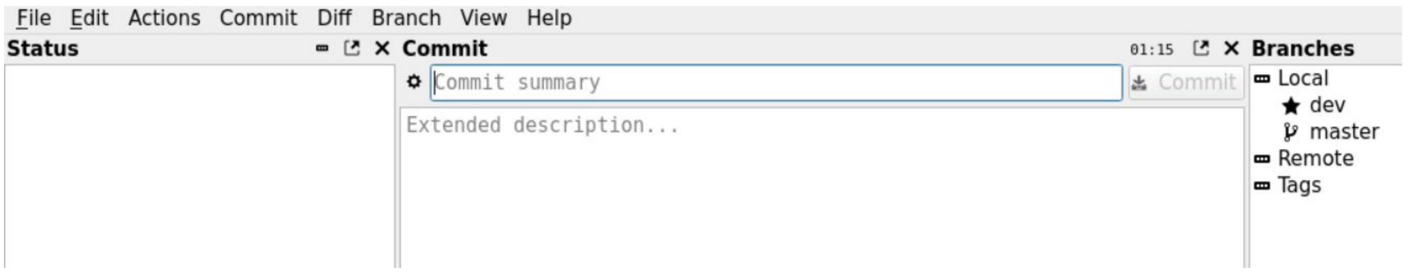
Step 3: Right-click on the dev.md to select "Stage Selected" and add a commit summary.



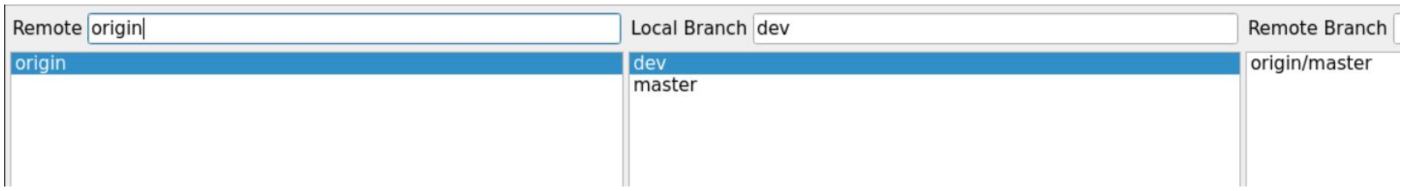
Staged
dev.md

dev file update

Step 4: Click on the Commit button

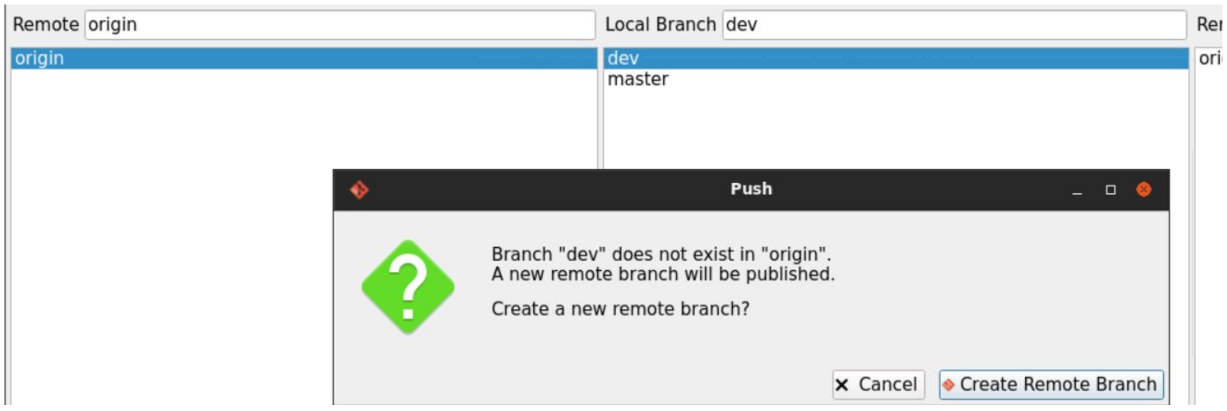


Push the file to the remote repository. Click on the Action button and select Push



Remote: origin
Local Branch: dev
origin
dev
master
origin/master

Step 5: Select the dev branch and click on the Push button located at the bottom right

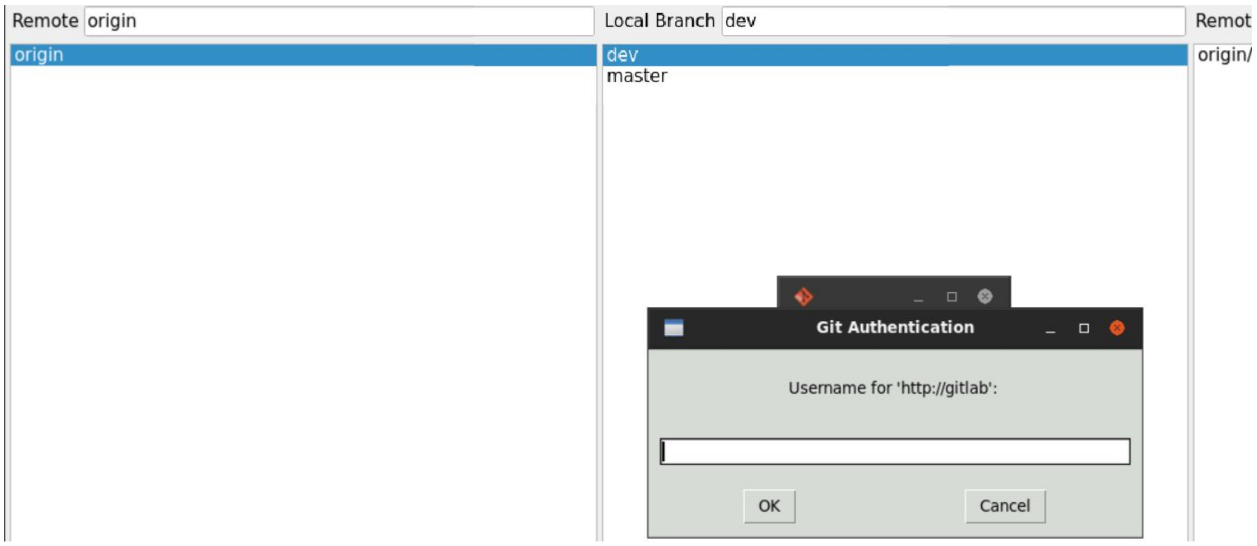


Remote: origin
Local Branch: dev

Branch "dev" does not exist in "origin".
A new remote branch will be published.

Create a new remote branch?

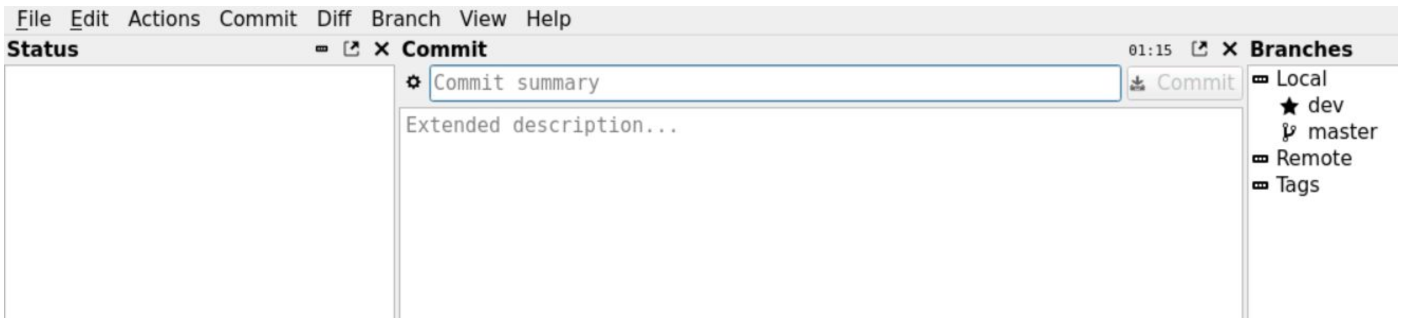
Click on “Create Remote Branch” button.



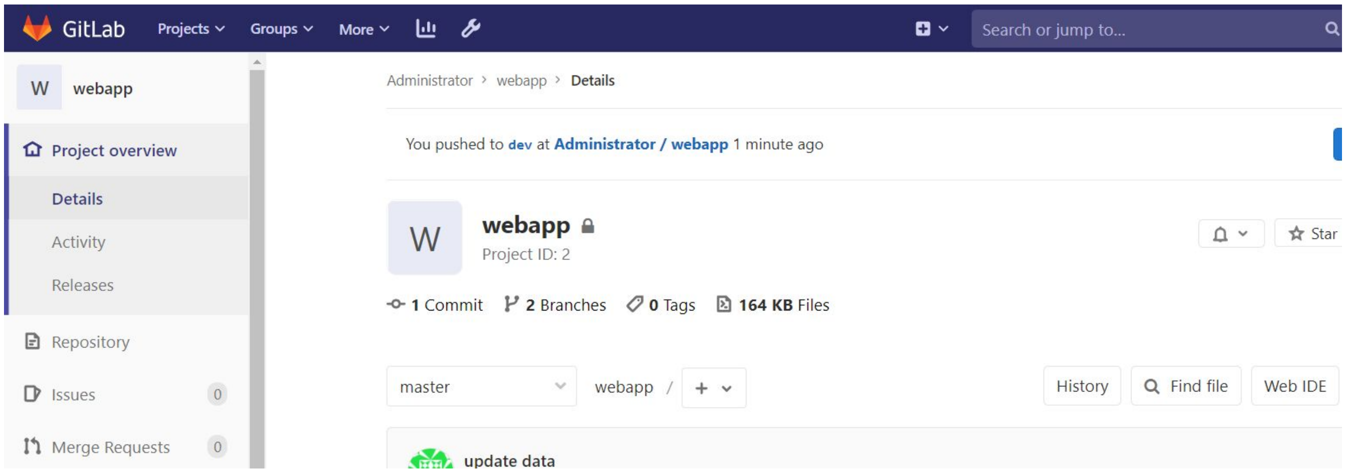
Git Authentication

Username for 'http://gitlab':

Enter the user credentials.



The file has been uploaded on the remote repository. Open the Gitlab Website to check the changes



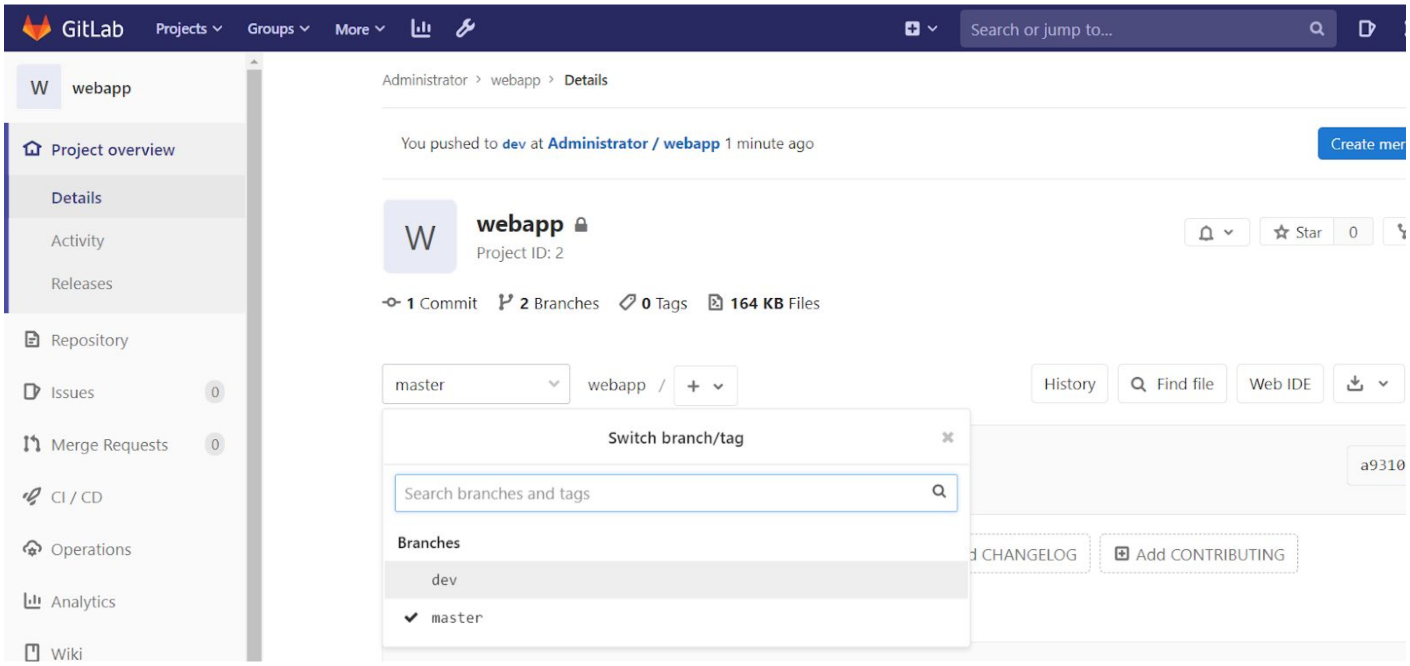
You pushed to dev at Administrator / webapp 1 minute ago

webapp
Project ID: 2

1 Commit 2 Branches 0 Tags 164 KB Files

update data

Click on the Change branch drop down to choose the branch



You pushed to dev at Administrator / webapp 1 minute ago

webapp
Project ID: 2

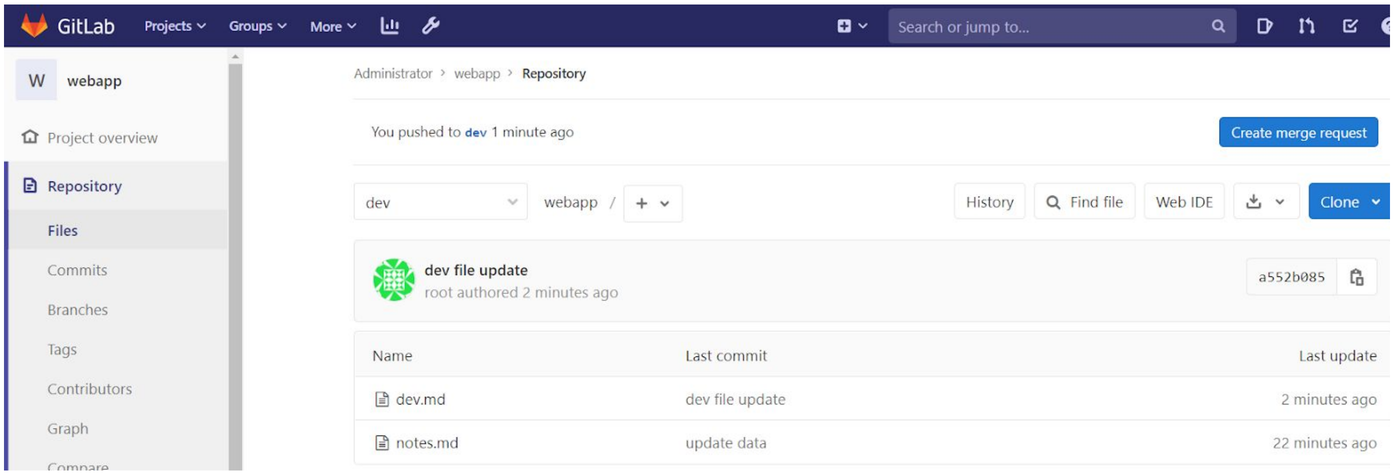
1 Commit 2 Branches 0 Tags 164 KB Files

Switch branch/tag

Search branches and tags

Branches
dev
✓ master

Select the Dev branch.



You pushed to dev 1 minute ago

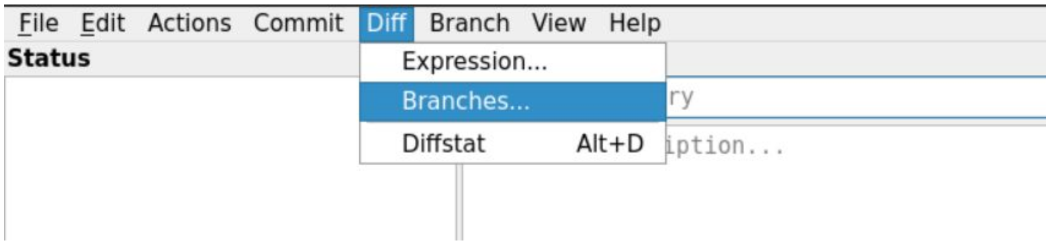
dev file update
root authored 2 minutes ago
a552b085

Name	Last commit	Last update
dev.md	dev file update	2 minutes ago
notes.md	update data	22 minutes ago

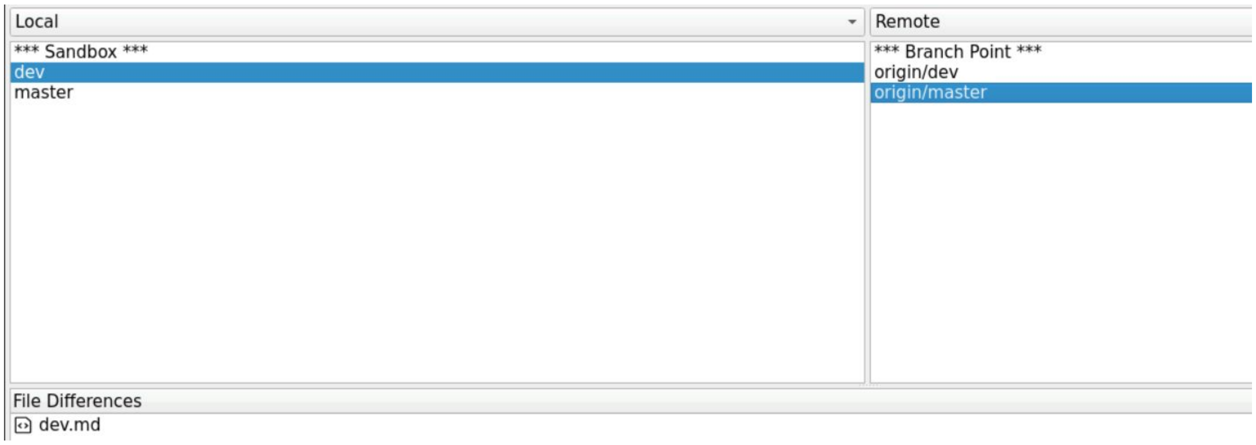
The dev.md file is created successfully in the dev branch.

7. Check the differences in two branches

Step 1: Navigate back to the kali instance and click on the Diff drop-down



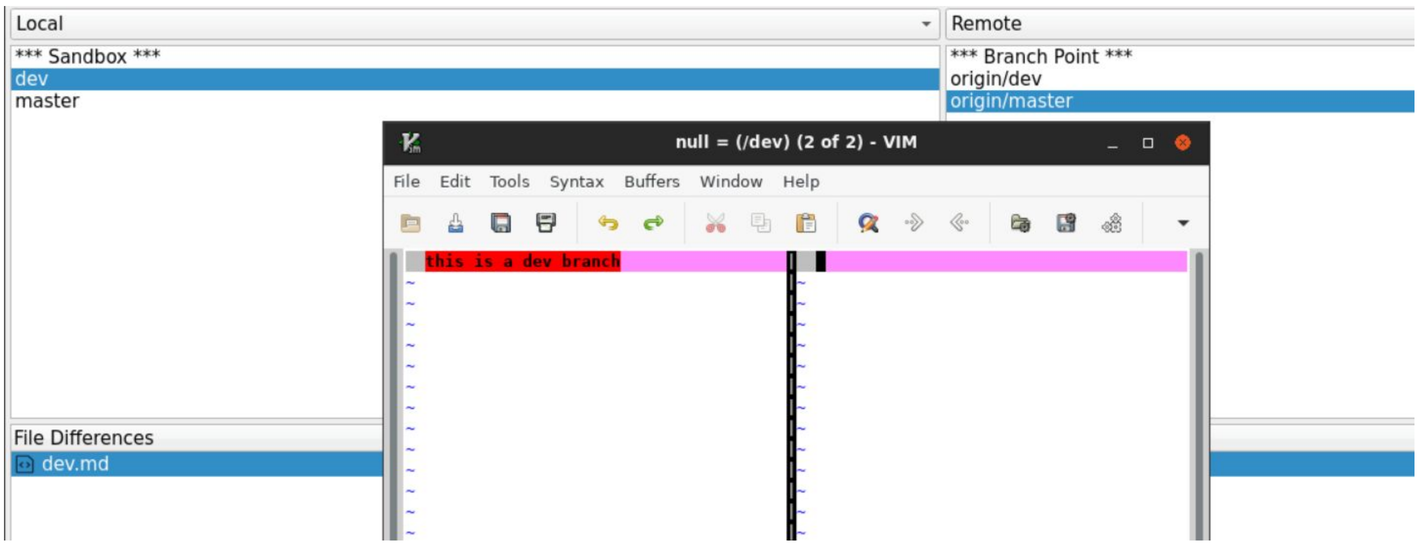
Step 2: Choose the Remote option in both sides and select dev and master



Local	Remote
*** Sandbox ***	*** Branch Point ***
dev	origin/dev
master	origin/master

File Differences
dev.md

Step 3: Select the dev.md and click on the Compare button.

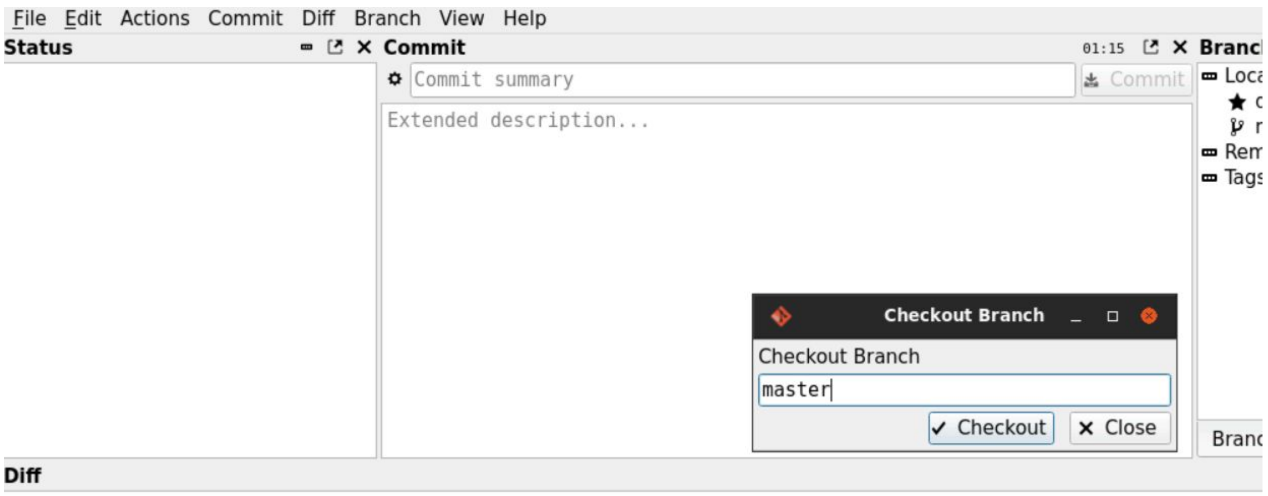


```
this is a dev branch
```

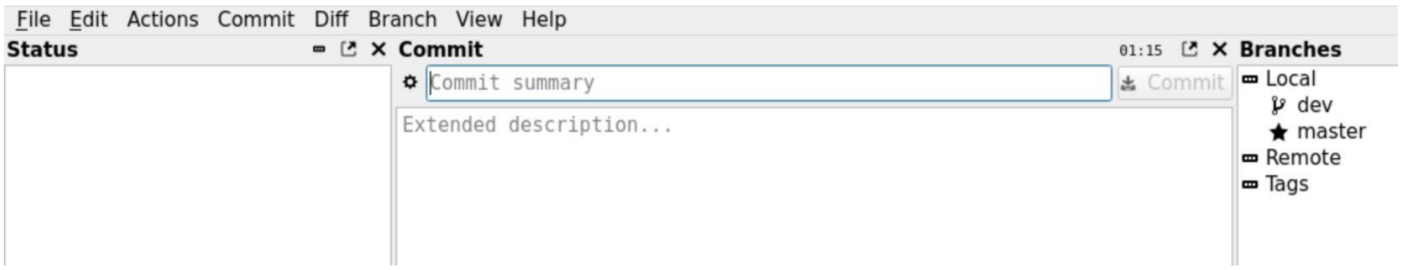
The results show that the file dev.md exists only in the dev branch. Close the vim and Diff viewer window.

8. Merge the branches

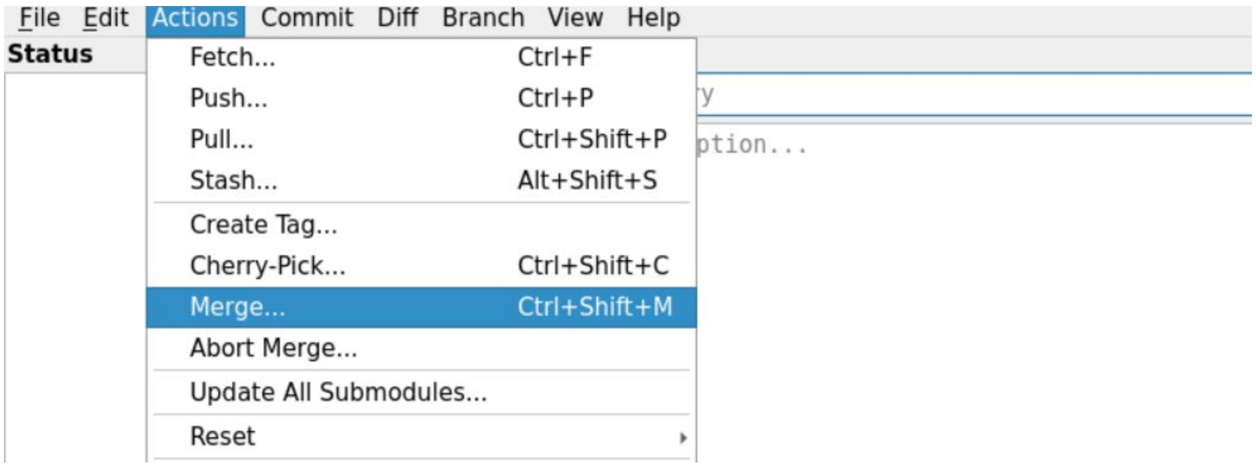
Step 1: Switch to the master branch by clicking on Branch drop-down and selecting the Checkout option



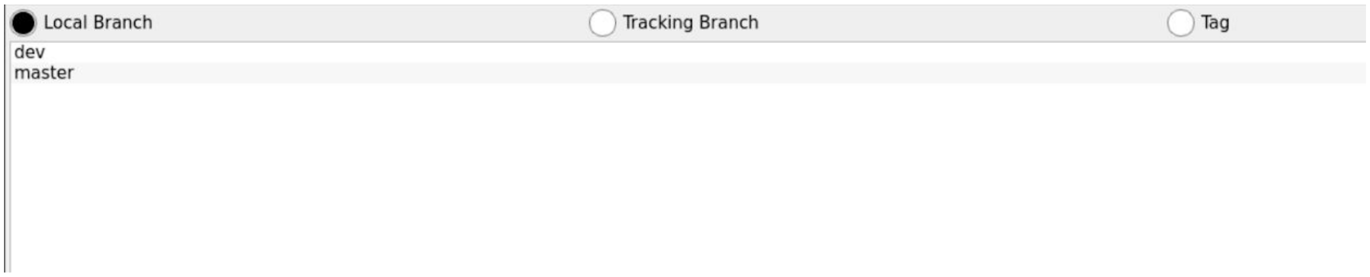
Select the master branch and click on Checkout



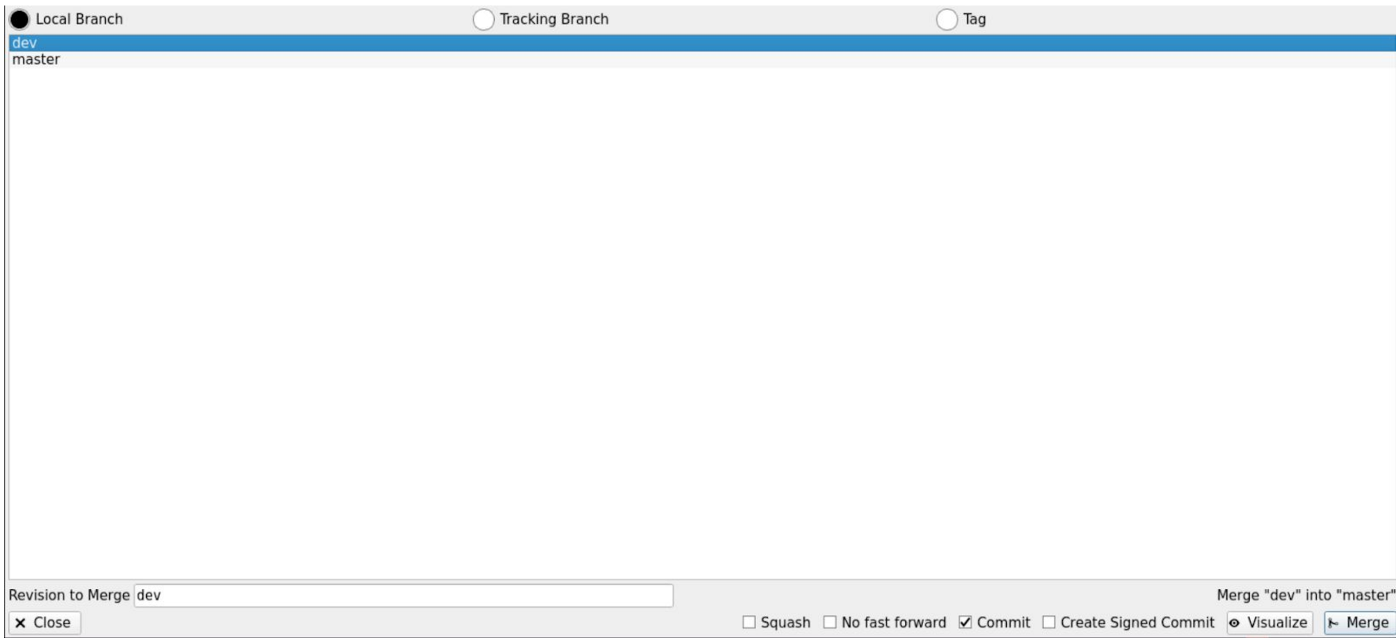
Step 2: Click on the Action drop-down



Choose the "Merge" option



Step 3: Select the dev branch



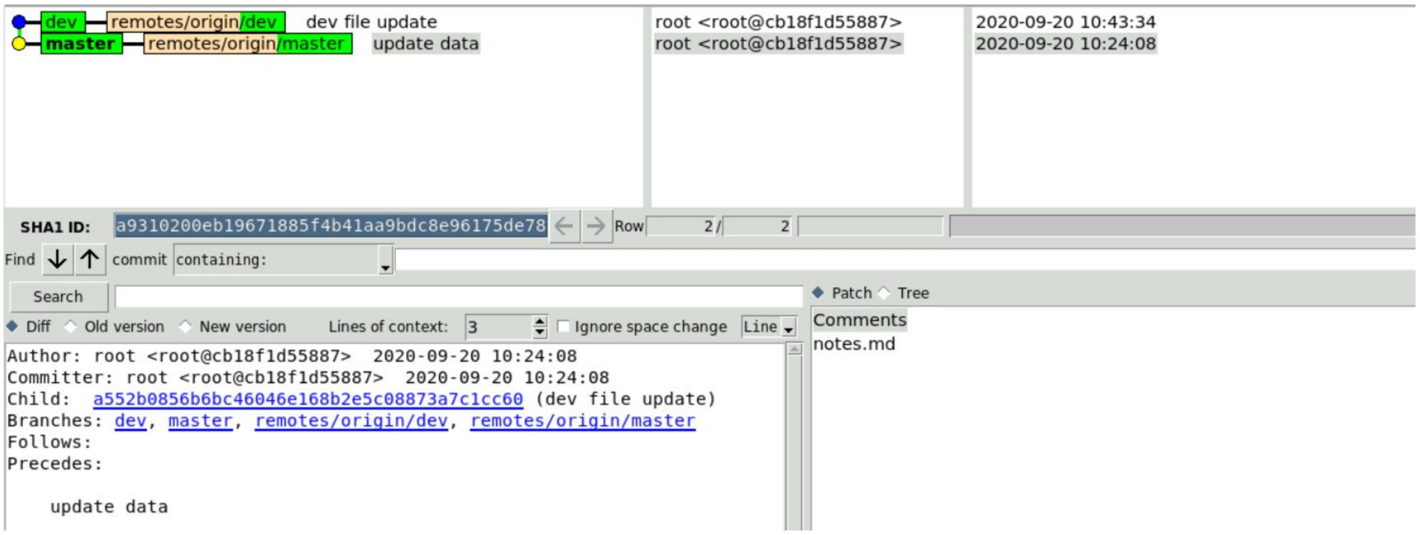
dev
master

Revision to Merge dev

Merge "dev" into "master"

☐ Squash ☐ No fast forward ☑ Commit ☐ Create Signed Commit

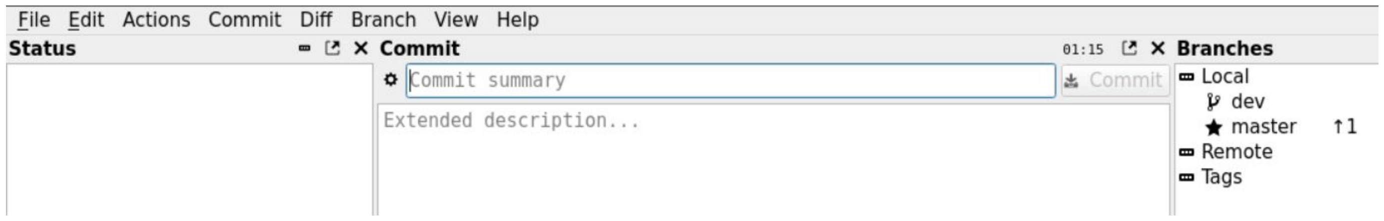
Step 4: Select the Visualise button to check the changes between both of the branches.



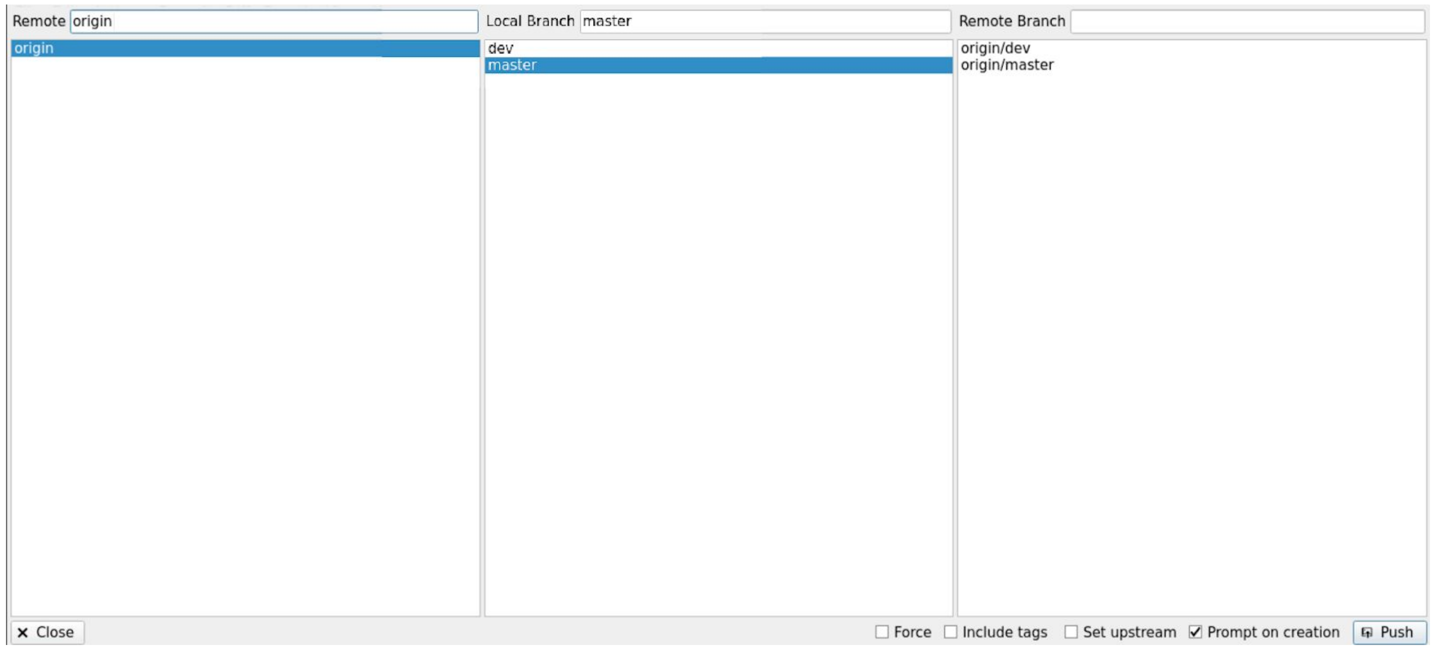
```
dev  remotes/origin/dev  dev file update        root <root@cb18f1d55887>  2020-09-20 10:43:34
master  remotes/origin/master  update data       root <root@cb18f1d55887>  2020-09-20 10:24:08

SHA1 ID: a9310200eb19671885f4b41aa9bdc8e96175de78   Row 2 / 2

Lines of context: 3

Author: root <root@cb18f1d55887>  2020-09-20 10:24:08
Committer: root <root@cb18f1d55887>  2020-09-20 10:24:08
Child:  a552b0856b6bc46046e168b2e5c08873a7c1cc60 (dev file update)
Branches: dev, master, remotes/origin/dev, remotes/origin/master
Follows:
Precedes:

    update data
```

Comments
notes.md

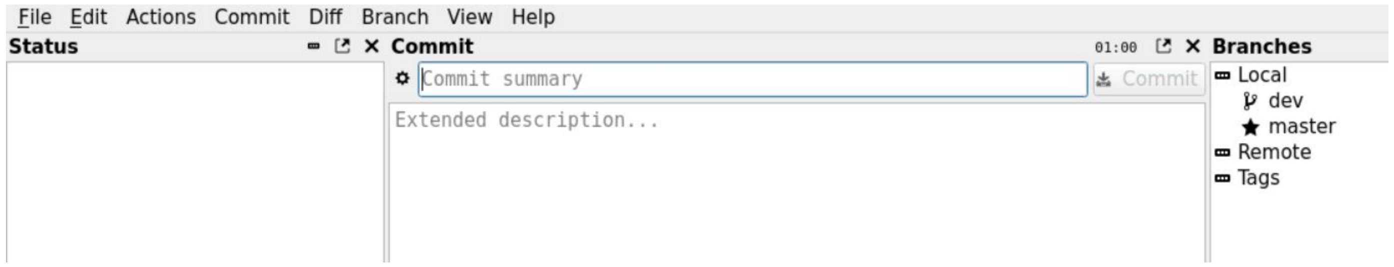
Close the visual tab and click on the Merge button.



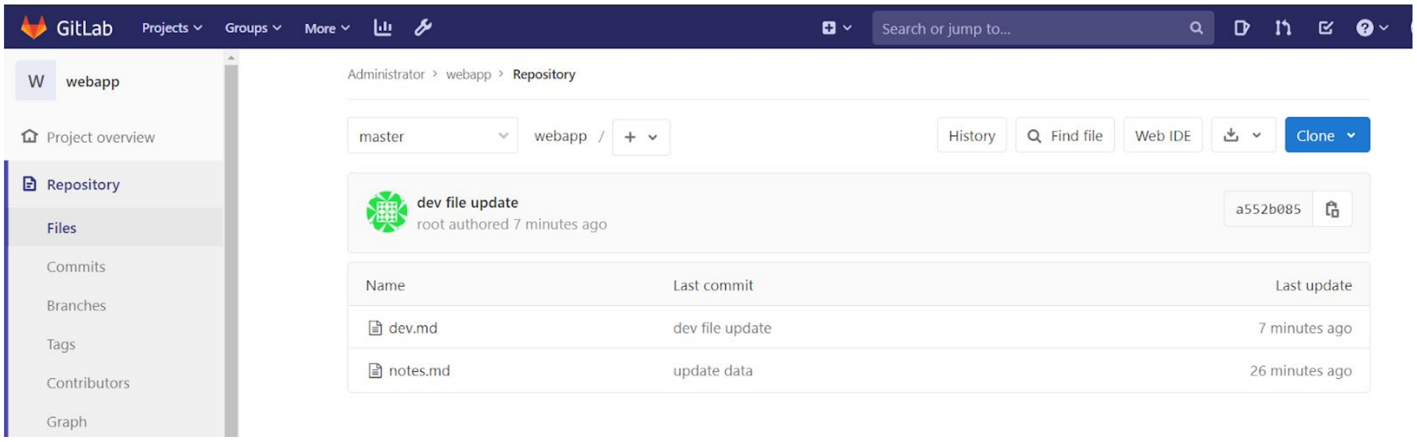
Step 5: Push the changes by selecting the push option under Action dropdown



Click on the Push button



The Branches have merged successfully, Switch to the master branch at Gitlab website to check the changes.



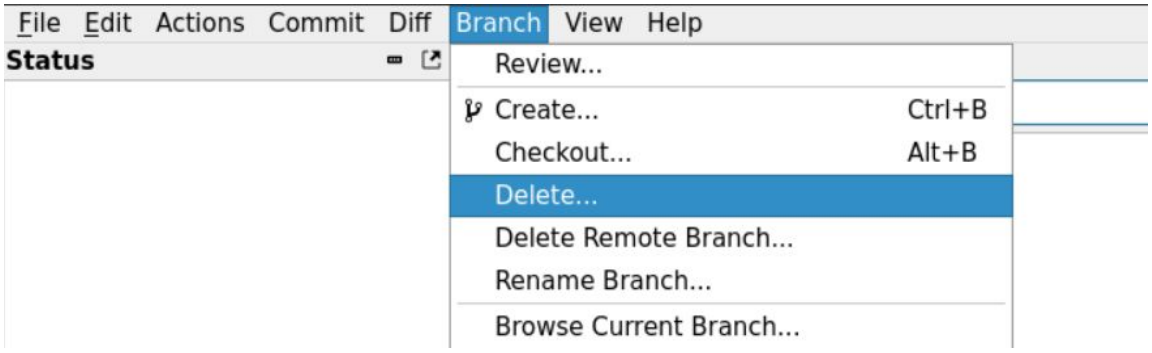
dev file update
root authored 7 minutes ago
a552b085

Name	Last commit	Last update
dev.md	dev file update	7 minutes ago
notes.md	update data	26 minutes ago

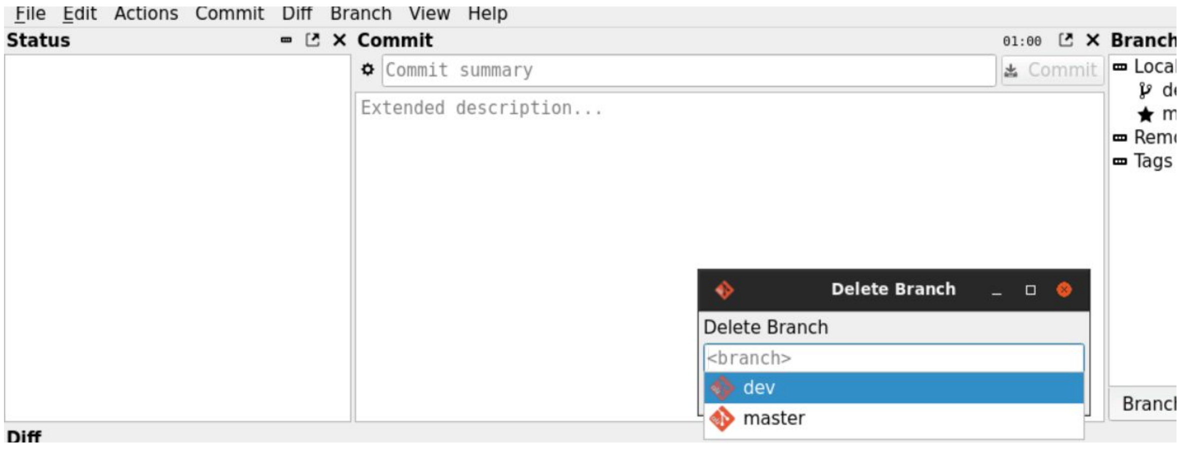
The dev.md is present in the master branch of the repository.

9. Delete the Dev branch

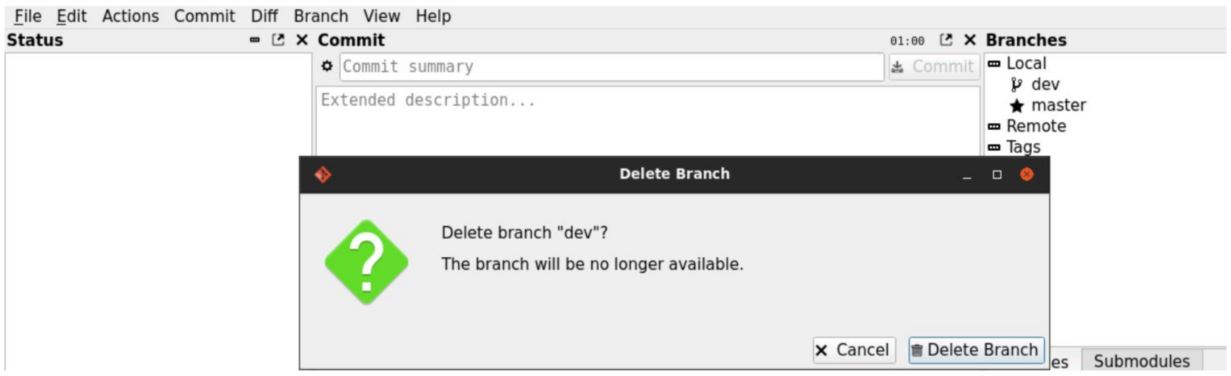
Step 1: Navigate back to the kali instance and click on the Branch drop-down and select the delete option



Step 2: Select the Dev branch



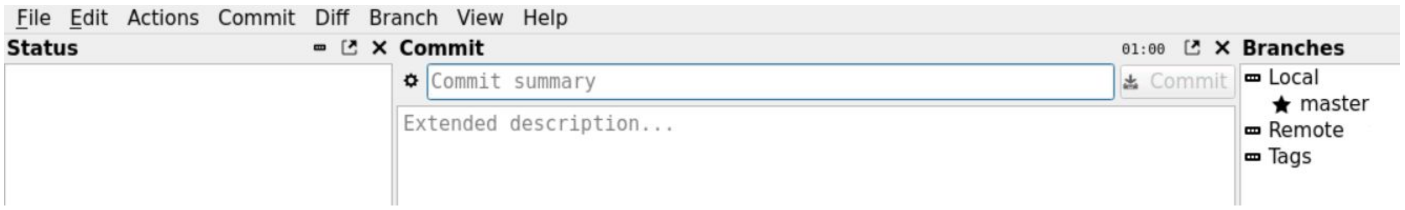
Step 3: Click on the Delete button



Delete branch "dev"?

The branch will be no longer available.

Step 4: Click on the Delete Branch button.



The dev branch has been deleted successfully.

Learning

- Interacting with Git VCS using Git Cola Git client.

References:

1. Git Manual (https://git-scm.com/docs)