Lab: Working Remotely

In this lab, you'll learn how you can share the changes in your repository with other people and combine their changes into your repository. This environment has been configured with a Git repository with a single commit. A remote repository is configured but has not been linked to the local repository.

With Git being a distributed version control system it means the local repository contains all the logs, files and changes made since the repository was initialised. To ensure that everyone is working on the most recent version changes need to be shared. When sharing these changes with other repositories only the differences will be synced making the process extremely fast.

Let's get started.

Pre-reqs:

- Google Chrome (Recommended)

Lab Environment

There is no requirement for any setup.

Run the following command in **terminal**: `cd ~/Desktop/gitlab-terraform/lab8/ && mv git .git`

Step 1 - Git Remote

Remote repositories allow you to share changes from or to your repository. Remote locations are generally a build server, a team members machine or a centralised store such as gitlab.com.

The friendly name allows you to refer to the location in other commands. Your local repository can reference multiple different remote repositories depending on your lab.

Task

This environment has a remote repository location of `/root/Desktop/gitlab-terraform/remote-project/1`. Using git remote, add this remote location with the name origin.

Solution

```
git remote add origin /root/Desktop/gitlab-terraform/remote-project/1
```

Step 2 - Git Push

When you're ready to share your commits you need to push them to a remote repository via git push. A typical Git workflow would be to perform multiple small commits as you complete a task and push to a remote at relevant points, such as when the task is complete, to ensure synchronisation of the code within the team.

The git push command is followed by two parameters. The first parameter is the friendly name of the remote repository we defined in the first step. The second parameter is the name of the branch. By default all git repositories have a master branch where the code is worked on.

Task

Push the commits in the master branch to the origin remote.

Solution

```
echo "first file" >> file.txt
git add file.txt && git commit -m "added file.txt"
git push origin master
```

Step 3 - Git Pull

Where git push allows you to push your changes to a remote repository, git pull works in the reverse fashion. git pull allows you to sync changes from a remote repository into your local version.

There should be terminal opened already. You can also open another New terminal by Clicking `File` > `New` > `Terminal` from the top menu.

Let's make some change from another folder and push them to remote. Run following remote in **terminal 2**:

```
cd ~/Desktop/gitlab-terraform/duplicate/ && mv git .git
git remote add origin /root/Desktop/gitlab-terraform/remote-project/1
git pull origin master

echo "new file" >> newfile.txt
git add newfile.txt && git commit -m "added newfile.txt #1234"
git push origin master
```

The changes from the remote repository are automatically merge into the branch you're currently working on.

Task

Pull the changes from the remote into your master branch. In the next step we'll explore what changes have been made.

Solution

Run the following command in **terminal 1**: `git pull origin master`

Step 4 - Git Log

As described in the previous lab you can use the git log command to see the history of the repository. The git show command will allow you to view the changes made in each commit.

In this example, the output from git log shows a new commit with the message "added newfile.txt #1234". The output of git show highlights the new lines added to the file in green.

Protip

Use the command `git log --grep="#1234"` in **terminal 1** to find all the commits containing **#1234**

Step 5 - Git Fetch

The command git pull is a combination of two different commands, git fetch and git merge. Fetch downloads the changes from the remote repository into a separate branch named **remotes//**. The branch can be accessed using git checkout.

Using git fetch is a great way to review the changes without affecting your current branch. The naming format of branches is flexible enough that you can have multiple remotes and branches with the same name and easily switch between them.

Task

Additional changes have been made in the origin repository. Use git fetch to download the changes and then checkout the branch to view them.

Protip

You can view a list of all the remote branches using the command `git branch -r`

Step 5 - Git Fetch

```
git fetch
```

```
git checkout remotes/origin/master
```