Exercises

Branching & Merging

1- Create a new branch called feature/login. Switch to the new branch.

2- Show all the branches.

3- Update file1.txt in the current branch (feature/login) and make a new commit.

4- Show the commits across all branches.

5- Switch back to the master branch. Show the commits in the feature branch that don't exist on master.

6- View the differences between master and feature/login.

7- Merge feature/login into master.

8- View the merged and unmerged branches.

9- Delete the feature branch.

10- Create a new branch called feature/logout. On this branch, write blue to file1.txt and make a commit.

Switch back to master, write green to file1.txt and make another commit.

Try to merge your feature branch into master. You'll see a conflict. Resolve the conflict by accepting both changes. When you're done merging, delete the new branch.

11- Create a new branch called bugfix/login. On this branch, write orange to file1.txt and make a commit.

Switch back to master, write green to file2.txt and make another commit.

View a graph of your branches. You'll see divergence.

Rebase the new branch on top of master.

View the graph of branches again. Note that the divergence is gone.

Do a fast-forward merge to bring the changes in the bugfix branch into master.

Solutions

1- Create a new branch called feature/login. Switch to the new branch.

```
git switch -C feature/login
```

2- Show all the branches.

```
git branch
```

3- Update file1.txt in the current branch (feature/login) and make a new commit.

```
echo sky >> file1.txt
git add .
git commit -m "Write sky to file1"
```

4- Show the commits across all branches.

```
git log --oneline --all
```

5- Switch back to the master branch. Show the commits in the feature branch that don’t exist on master.

```
git switch master
git log master..feature/login
```

6- View the differences between master and feature/login.

```
git diff master..feature/login
```

7- Merge feature/login into master.

```
git merge feature/login
```

8- View the merged and unmerged branches.

```
git branch --merged
git branch --no-merged
```

9- Delete the feature branch.

```
git branch -d feature/login
```

10- Create a new branch called feature/logout. On this branch, write blue to file1.txt and make a commit.

Switch back to master, write green to file1.txt and make another commit.

Try to merge your feature branch into master. You'll see a conflict. Resolve the conflict by accepting both changes. When you're done merging, delete the new branch.

```
git switch -C feature/logout
echo blue >> file1.txt
git commit -am "Write blue to file1"

git switch master
echo green >> file1.txt
git commit -am "Write green to file1"

git merge feature/logout
git mergetool
git add file1.txt
git commit

git branch -d feature/logout
```

11- Create a new branch called bugfix/login. On this branch, write orange to file1.txt and make a commit.

Switch back to master, write green to file2.txt and make another commit.

View a graph of your branches. You'll see divergence.

Rebase the new branch on top of master.

View the graph of branches again. Note that the divergence is gone.

Do a fast-forward merge to bring the changes in the bugfix branch into master.

```
git switch -C bugfix/login
echo orange >> file1.txt
git commit -am "Write orange to file1"

git switch master
echo green >> file2.txt
git commit -am "Write green to file2"

git log --oneline --all --graph

git switch bugfix/login
git rebase master

git log --oneline --all —graph

git switch master
git merge bugfix/login
```