Exercises

Collaboration

1- Create a GitHub repository called Tokyo.

2- Clone this repository in two different folders: John and Amy. We're trying to simulate two users collaborating on this repository.

3- Go to John's folder and make a commit. To save yourself from the hassle of re-entering your credentials every time you do a push, store your credentials in memory. Now, do a push.

4- Go to Amy's folder, and fetch the new commit. View the history and see how the remote master branch has moved forward. Merge origin/master into master.

5- From Amy's folder, create a new branch called feature/login. Make a commit on this branch. Once again, you need to configure Git to store your credentials in memory because you're in a different repository. The configuration we made earlier only applies to John's repository. Now, do a push.

6- View the local and remote branches.

7- Go to John's folder and do a fetch. View the history. Note the new branch. Create a local branch and map it to origin/feature/login. View the local and remote branches again to make sure your branches are set up properly.

8- Go back to Amy's folder, make another commit on the feature branch and do a push.

9- Back to John's folder, pull Amy's changes into the feature branch. View the history. The feature branch should be two commits ahead of master.

10- Merge the feature branch into master. View the history. Note that master is ahead of origin/master by two commits. So now you need to do a push to bring origin/master forward. View the history again to make sure master and origin/master are at the same place.

11- You're done with the feature branch. So it's time to remove it. Now, view the local and remote branches. The remote tracking branch is gone, but the local branch is still there and should be removed.

12- Go to Amy's folder. Do a fetch. View the history. Note that origin/master has moved forward and is ahead of master by two commits. Switch to master and merge origin/master into it. View the history to ensure master and origin/master are at the same location.

13- It's time to remove the local and remote-tracking branches.

Well done! You rocked! The stuff you've done requires a deep understanding of Git. Now, go grab a coffee or your favourite drink. :)

Solutions

1- Create a GitHub repository called Tokyo.

2- Clone this repository in two different folders: John and Amy. We're trying to simulate two users collaborating on this repository.

git clone <url> John
git clone <url> Amy

3- Go to John's folder and make a commit. To save yourself from the hassle of re-entering your credentials every time you do a push, store your credentials in memory. Now, do a push.

cd John
echo hello > file1.txt
git add .
git commit -m "Add file1"
git config credential.helper cache
git push

4- Go to Amy's folder, and fetch the new commit. View the history and see how the remote master branch has moved forward. Merge origin/master into master.

cd ..
cd Amy
git fetch
git log --oneline --all --graph
git merge origin/master

5- From Amy's folder, create a new branch called feature/login. Make a commit on this branch. Once again, you need to configure Git to store your credentials in

memory because you're in a different repository. The configuration we made earlier only applies to John's repository. Now, do a push.

git switch -C feature/login
echo hello > file2.txt
git add .
git commit -m "Add file2"
git config credential.helper cache
git push -u origin feature/login

6- View the local and remote branches.

git branch -vv
git branch -r

7- Go to John's folder and do a fetch. View the history. Note the new branch. Create a local branch and map it to origin/feature/login. View the local and remote branches again to make sure your branches are set up properly.

cd ..
cd John
git fetch
git log --oneline --all --graph
git switch -C feature/login origin/feature/login
git branch -vv

8- Go back to Amy's folder, make another commit on the feature branch and do a push.

cd ..
cd Amy
echo world >> file2.txt
git commit -am "Update file2"
git push

9- Back to John’s folder, pull Amy’s changes into the feature branch. View the history. The feature branch should be two commits ahead of master.

cd ..
cd John
git switch feature/login
git pull
git log --oneline --all --graph

10- Merge the feature branch into master. View the history. Note that master is ahead of origin/master by two commits. So now you need to do a push to bring origin/master forward. View the history again to make sure master and origin/master are at the same place.

git switch master
git merge feature/login
git log --oneline --all --graph
git push
git log --oneline --all --graph

11- You’re done with the feature branch. So it’s time to remove it. Now, view the local and remote branches. The remote tracking branch is gone, but the local branch is still there and should be removed.

git push -d origin feature/login
git branch -vv
git branch -d feature/login

12- Go to Amy’s folder. Do a fetch. View the history. Note that origin/master has moved forward and is ahead of master by two commits. Switch to master and merge origin/master into it. View the history to ensure master and origin/master are at the same location.

cd ..
cd Amy
git fetch
git log --oneline --all --graph
git switch master
git merge origin/master
git log --oneline --all —graph

13- It’s time to remove the local and remote-tracking branches.

git remote prune origin
git branch -d feature/login

Well done! You rocked! The stuff you’ve done requires a deep understanding of Git. Now, go grab a coffee or your favourite drink. :)