Git / GitHub Workshop

Clarusway

Subject: Git Operations

Learning Goals

- Practice using the Git commands.

Introduction

We've covered a lot of Git concepts, but now it's time to put the concepts in to practice. We'll start with Git commands.

Code Along

Part 1 - Create a local repository

1. Open the terminal (Git Bash for Windows user)

- Go to Desktop and create a directory named "my-github" if you do not have already. And, go to "my-github" directory.

```
mkdir my-github
cd my-github
```

- Create another folder named "git-workshop" and go to "git-workshop" directory.

```
 mkdir git-workshop
 cd git-workshop
```

2. Git configuration

- Configure git with our name and email. This is to identify who has done what on git and github.

```
git config --global user.name <your_user_name>
git config --global user.email <your_email>
```

- Check the setting

```
git  config --list
```

3. Create a local repository

- We can do that by running the "init" command.

```
git init
```

- Check the if ".git" folder is created.

```
ls -a
```

Part 2 - Create a remote repository

4. Create a remote repository on GitHub

- Go to your GitHub account and create a repository named "git-workshop".
  - Write a description for your repo
  - select Public
  - add a README.MD file

5. Go to terminal

- Check the connected remote repositories. The 'git remote -v' lists all currently configured remote repositories, which at this point is none.

```
git remote -v
```

- connect to remote repository

```
git remote add origin <remote repo URL>
```

- Verify the new connection

```
git remote -v
```

6. Create a file named "file1.txt"

- check the status of the project folder

```
git status
```

- store the change in the local repo

```
git add file1.txt"
git commit -m "xxxx"
```

7. upload the changes to the remote repo

```
  git push -u origin master
```

- check the files on the github repo. (select master branch in GitHub)

Part 3 - Working with branches

8. Create a new remote repo named "git-workshop-2" in GitHub.

9. Clone the remote repo

- go the terminal
- clone the "git-workshop-2"

```
git clone <remote repo URL>
```

- check the files in the "git-workshop-2" and see the README.MD and .git file.

```
ls -a
```

10. Create a file named **test.txt**
11. Create a new branch named new-feature-1.

```
git branch new-feature-1
```

- See branches

```
git branch (show local branchs)
git branch -r (show remote branchs)
git branch -a (show all local and remote branchs)
```

- Switch to new-feature-1

```
git checkout new-feature-1
```

- List the files and check the status of the working directory

```
ls
git status
```

- Make some changes in the test.txt file, and check the status

```
vim test.txt
git status
```

- Store the changes to the repo and check the status

```
git commit -am "added first line"
git  status
```

- Add another line to test.txt and store it to the local repo.

```
vim test.txt
git commit -am "added second line"
```

- Switch the main branch and see the content of the test.txt

```
git checkout main
cat text.txt
```

- Merge new-feature-1 branch to main branch.

```
git merge
```

```
cat test.tst
```

12. Create a new branch named new-feature-2 and switch to it.

```
 git checkout -b new-feature-2
```

- Create a new file named test2.txt, add a line in it and store the changes to repo.

```
vim test2.txt
git add .
git commit -m “created text2.txt”
```

- Switch the main branch again.

```
git checkout main
```

- Create a new file test3.txt and send the changes to local repo.

```
touch test3.txt
git add .
git commit -m “created text3.txt”
```

- Open the file named test2.txt, add a line in it and store the changes to repo.

```
vim test2.txt
git add .
git commit -m “created text2.txt”
```

- merge main branch with new-feature-2

```
git merge new-feature-2
```

13. RESOLVE THE CONFLICT

- edit the file.
- then commit it.

14. Push the local changes to the remote repository

```
git push
```

15. Go and check the remote repository, you will see the new files

16. Go to the terminal and delete the branches named new-feature-1 and new-feature-2

```
git branch -d new-feature-1
git branch -D new-feature-2
```

- List the all branches

```
git branch -a
```

☺ Thanks for Attending ✍

Clarusway