Git / GitHub Workshop-1

Clarusway

Subject: Git Operations

Learning Goals

- Practice using the Git commands.

Introduction

- We've covered some basic 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.

```
cd deskop

mkdir my-githup

cd my-githup
```

- Create another folder named "git-workshop" in the "my-github" folder and go to "git-workshop" directory.

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

2. Git configuration (if you have already done, skip this part)

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

```
git config --global user.name "Jack"
git config --global user.email "muslu34613461@gmail.com"
```

- 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 -al
```

4. Check your default branch name and If your branch name is "master", change it to "main" then check the branch again.

```
git branch -m master / git branch main
```

5. Create a file named "file1.txt"

```
touch file1.txt
```

- check the status of the project folder

```
git status
```

6. Stage **file1.txt** and check the status of the project folder again.

```
git add file1.txt
```

7. Store it to the local repository.

```
git commit -m "file1.txt created"
```

8. Using Vim editor, create a file named **test2.txt**

```
vi test2.txt
```

9. Stage **file2.txt** and check the status of the project folder.

```
git add file2.txt
```

10. Unstage **file2.txt**

```
git restore  --staged file2.txt
```

11. check the status of the directory

```
git status
```

12. Store the changes to the local repeository

```
git commit -m "This is second commit"
```

13. List the commits

```
git log
```

20. switch to the first commit

```
git checkout " ilk beş karakter"
```

21. switch to the last commit.

```
git checkout ....
```

Part 2 - Working with branches

22. Create a file named **test.txt**

```
touch test.txt
```

23. Create a new branch named "new-feature-1".

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

- See branches

```
git branch
```

- Switch to new-feature-1

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

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

```
ls -al
git status
```

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

```
echo "bu bir denemedir" > test.txt
git status
```

- Store the changes to the repo and check the status

```
git add test.txt
git status
```

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

```
echo "bu ikinci denemedir" >> test.txt
git status / git add ... /git commit
```

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

```
git checkout
cat test.txt
```

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

```
git merge new-feature-1
```

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

```
git branch new-feature-2
git checkout new-feature-2
```

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

```
echo "bu ikinci dosyanın ilk denemesidir" > test2.txt
git add test.txt
```

- Switch the main branch again.

```
git checkout master
```

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

```
touch test3.txt
git add
```

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

```
vi test2.txt (add line eklendi)
git add test2.txtgit commit
```

- merge main branch with new-feature-2

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

25. RESOLVE THE CONFLICT

- edit the file.
- then commit it.

26. 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
 master var sadece
```

☺ **Thanks for Attending** ✍

Clarusway