— Git-4 In Class

Part-1

1. Create repo on your Github account named it-fundamentals.
2. Create a folder on your Computer named my-github.

   mkdir my-github

3. Clone your it-fundamentals remote repo in my-github. folder.

   cd my-github
   git clone http://github........ {clone yaptığımızda git init yapmamıza gerek kalmıyor. repoyu klonelayacak oluşturuyor)

4. Create another folder on your computer named clarusway-github

   mkdir clarusway-github

5. Clone the clarusway-it-fundamentals-8-21 repo in clarusway-github folder.

   cd clarusway-github
   git clone http://github.com/clarusway/clarusway-it-fundamentals-8-21

6. Copy all the files and folders in the clarusway-github folder to my-github folder.

   (Bu işlemi windowstan pratik olarak yaptık)
   .git klasörünü kopyalamadık!! my-github'da .git klasörü zaten var.

7. Add the changes to your local repo

   cd my-github
   git add .
   git commit -m "copied all the files"

8. Then send them to the remote repo.

   git push ('Dikkat ederseniz, commit yapmadan push yapılmıyor!!)
   Yoksa az önceki kopyalama işlemini commit yapmadığımızdan herşey zaten güncel der.!

Part-2

9. Download the changes from clarusway-it-fundamentals-8-21 (see them in the working directory)

   git pull

10. Copy the lab1.txt file from clarusway github folder to my-github/it-fundamentals/git/lab folder.

↓ 6. md de tüm dosyaları kopyalamıştık lab1.txt'yi sonradan mı oluşturduk?

→ Hoca clarusway altına lab diye klasör ve içine lab1.txt diye dosya oluşturdu. (Biz kopyaladıktan sonra da kaynak klasörde değişiklik olabilir simülesi

(Git 4/1)

— Git 4 - In Class —

Clarusway yetkilisi dedi ki ben lab1.txt diye dosya ekledim. Onu alın (github'da)

git bash'e dönüyoruz ve clarusway klasöründeyken yeni dosyayı alabilmek için pull yapacağız

```
git pull
cd git
cd lab  → lab1.txt'yi gördük
```

Şimdi windows'tan clarusway/1st-funda / git / lab / lab1.txt'yi my-github / it-fundamentals / git / lab 'ın içine kopyaladık.

11- In the my-github/it-fundamentals directories:
Check the status of the project. Add lab1.txt file to index area and see the status again.

```
git status
git add lab1.txt   (başka bir değişiklik yapmadığımız için git add . de olur)
git status
```

12- Save the changes to the local repo.

```
git commit -m "for lab1.txt"
```

13. See commit history.

```
git log --oneline
```

14. Upload the changes to your remote repository.

```
git push.  (Dikkat! Önce commit ettik!)
```

Part 3

15- Go to my-github/it-fundamentals/git/lab directory:
Create a new branch named front-end

```
git branch front-end
```

See branches

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

- Git-4 - In Class -

- Switch to front-end branch

```
git checkout front-end
```

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

```
ls
git status
```

- make some changes in the lab1.txt file, and check th status.

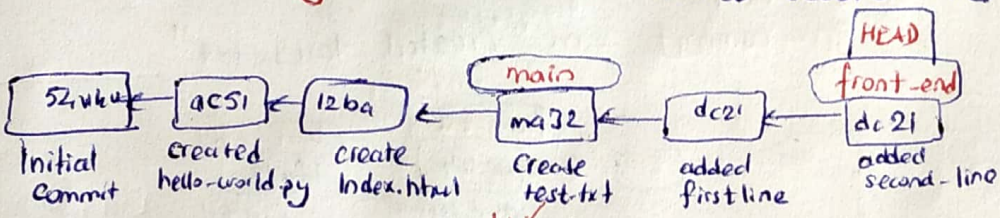
```
vim lab1.txt (insert changes, save and quit)
git status
```

- Store the changes to the repo and check the status

```
git commit -am "added first line to lab1.txt"
git status
```

- Add another line to lab1.txt and store it to the local repo.

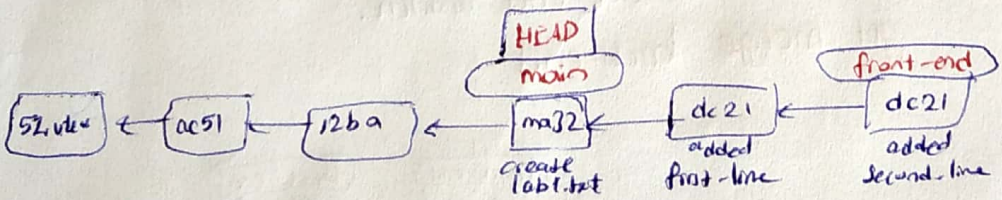
```
vim lab1.txt
git commit -am "added second line to lab1.txt"
```



↳ burada front-end branch'i olusturduk.

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

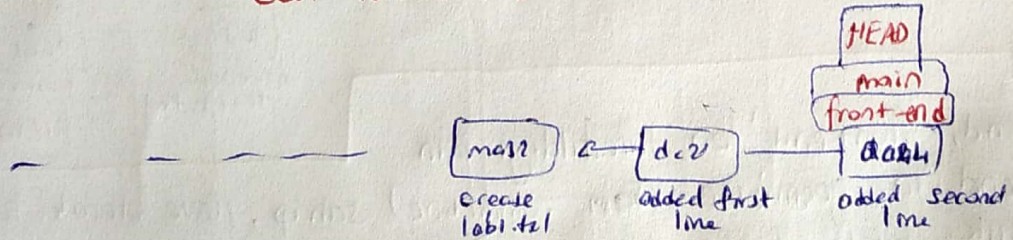
```
git checkout main
cat lab1.txt
```



- merge front-end branch to main branch.

```
git merge front-end
cat lab1.txt
```

→ artık main br.'da lab1.txt'yi görebiliyoruz

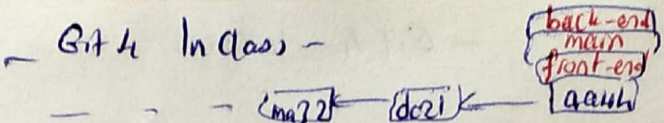


- Create a new branch named back-end and switch to it.

```
git checkout -b back-end
```

(Git 4 / 3)

Git 4 In Class -

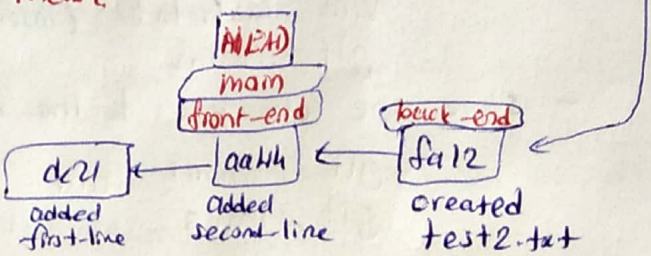


- Create a new file named test2.txt and store the changes to repo

```
touch test2.txt
git add .
git commit -m "created test2.txt"
```

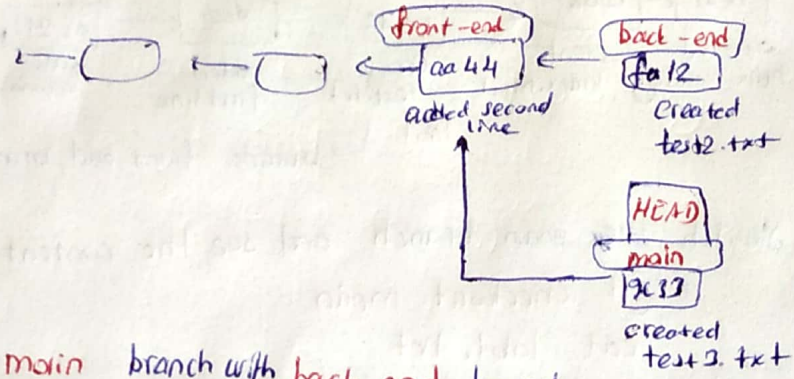
- Switch the main branch again

```
git checkout main
```



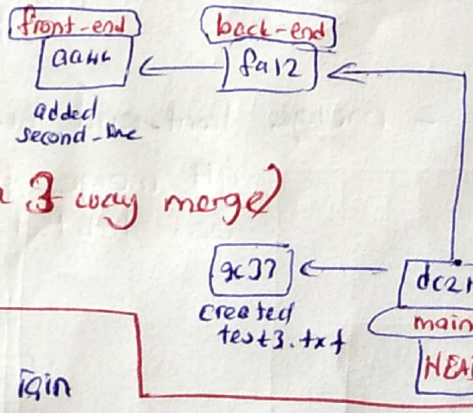
- Create a new file test3.txt and send the changes to local repo

```
touch test3.txt
git add .
git commit -m "created test3.txt"
```



- Merge main branch with back-end branch.

```
git merge back-end
```



(You have performed a 3 way merge)

merge commit oluşturdu

***

back-end, front-end'ten türetildiği için front-end'in geçmişine (son commitine) sahip. İlave olarak test2.txt'yi oluşturduk. Geri dönüp main'e geçtik ve main'de test3.txt'yi oluşturduk. Sonra back-end'i main'e merge ettik. Dolayısıyla şimdi en güncel olurum main branch'tadır. (test2.txt ve test3.txt dahil) Yani frontend ve backend'e artık ihtiyacımız yok!

(Git 4 14)

-Git-4 In Class-

Part 4

- Send the changes to the remote repository.

```
git push
```

(Son durum main branch'ta olduğu için buradan remote'a push ettik. artık diğer branchlarla işimiz bitti.)

- Go and check the remote repository

Part 5

- Go to the terminal and delete the branches named frontend and back-end.

```
git branch -d front-end
git branch -D Back-end
```

- list the branches

```
git branch -a
```

Git in Class -

Forking a Project

Part 1. How to propose changes to a project. The workflow is something like this:

1. Fork the repository
2. Create a fix (commit changes)
3. Submit a pull request to the original project.

Part 2 To get started with Git, we have to do 2 things, firs of all configure Git with our name and email This is to identify who has done what on GitHub.

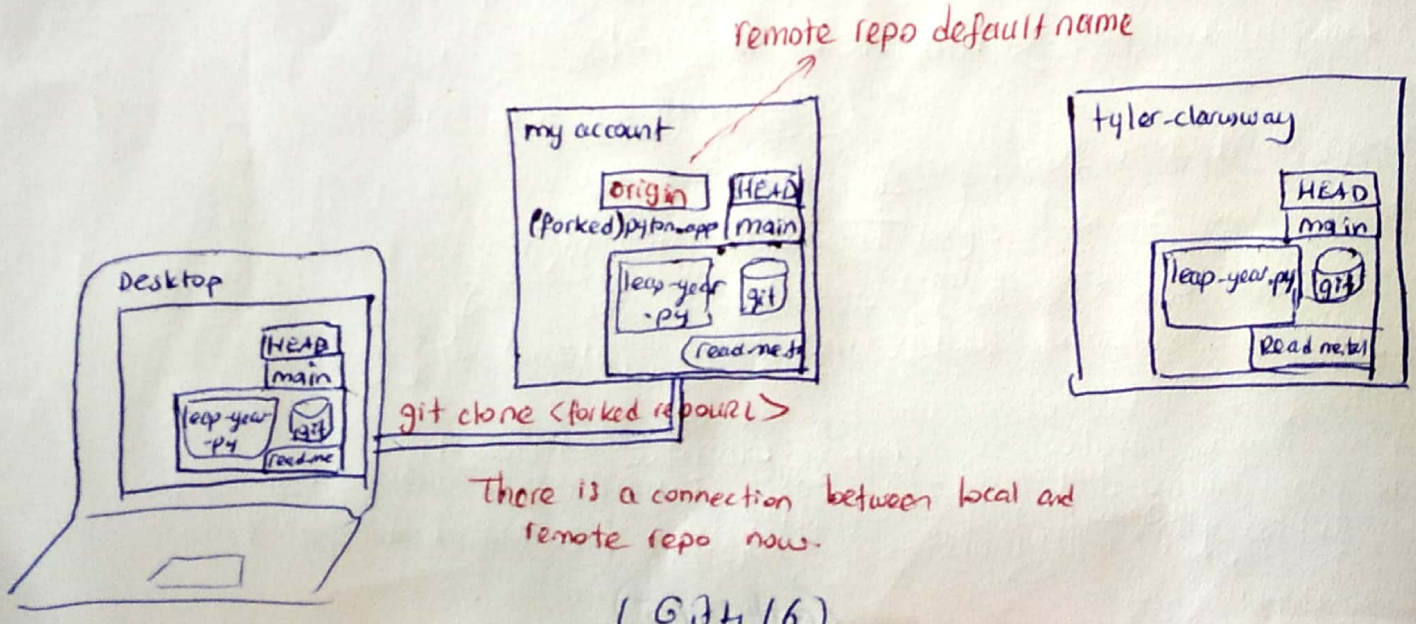
```
git config --global user.name " _____ "
git config --global user.email " _____ "
```

Secondly, we need to create a Repository.

a. git init command
b. clone a existing repository from Git Hub

Part 3 Cloning command is awesome, not only does it get the repository, it also sets up the remote tracking for all of the branches.

```
git clone <forked-repo URL>
```



There is a connection between local and remote repo now.

— Git 4 In Class —

Part 4 (Make your changes)

Branching is a way for you to work on different features in the code at the same without destroying the master / main branch

The basic idea is to create a feature branch, do some work, then integrate it into the master / main branch.
You can see a branch as a clean slate where you can do whatever you want without ruining someone elses work

git branch → The only branch for now is the main branch

git branch <new branch name> → create a br.
git checkout -b <new branch name> → create and switch to br

Go ahead and open the leap-year.py in the text editor and change it

```
git status
git add leap-year.py
git commit -m "added new line"
```

Part 5

Get latest changes from original repository.

Now there may have been some changes that are of interest in the original repo. that we've forked from. Let's take a look and get them.

(Fork ettiğimiz asıl repo!)

Specify a new remote upstream repository that will be synced with the fork

git remote add upstream <owner_repoe-URL>

(Fork ettiğimiz owner repo'yu upstream repo olarak ~~tanımladık~~ ~~afpsladık~~.)

(buraya istediğin ismi verebilirsin ama yaygın olarak "upstream" kullanılıyor)

Verify the new upstream repository you've specified for your fork.
git remote -v

git pull upstream main. → upstream olarak tanımladığımız repo'nun main branch'ından pull ettik

-Git 4 In Class-

Part 6 (Push!)

Run the following command to push what you've done to the new-feature branch of your fork! (but it won't work)

git push origin

{fatal: the current branch new-feature has no upstream branch}

To push the current branch and set the remote as upstream, use; git push --set-upstream origin new-feature

or shortcut; git push -u origin new-feature

(origin repo'da new-feature adı branch oluşturup oraya push ettik.)

fork ettiğimiz bizim github hesabındaki repo.

(main branch'a merge edip main (local) den main (origin)'e push edebilirdik. Ama çalıştığımız şirkette herkesin branch'tan göndermemiz isteniyor. ~~...~~ çalıştığı branchar bellidir)

Part 7 (Pull Request)

This is were you tell the owner of the original repository that you forked from that you would like them to merge your changes into their repository. You could say that you send them a request to pull the your work.

This is an operation performed on github.

- Go to the fork repository (in your own account)
- Click th big, friendly, green button that says "compare & pull request"
- Note: you can also reach "pull request" button if click "branches" button.

Now, find the text ~~say~~ that says "compare across forks" as part of the sentence right underneath the large "Open a pull request" write something nice about it so that the owner knows what you (at least tried to) accomplish. Then click the "create pull request"

- Wait for the ~~original~~ answer of the owner of the original repo.
- Owner of the original repo: Click "Files changed"
  Click the "Review changes"
- owner can have 3 options "Comment", "Approve" or "Request changes"
- click "submit review" (Git 4/8) Then click "merge pull request" Finally "Confirm merge"

! Not

Pull requestte destination repo yani katkıda bulunmak istediğimiz asıl projenin sahibi bir hata veya eksiklik görürse "Approve" etmeyip bunun düzeltilmesini isteyebilir. ("comment" ile)

Bu durumda biz gene çalıştığımız branchta düzeltmeyi yapıp yine kendi remote reposmuza push yaptığımızda yeniden pull request'e gerek kalmadan otomatik olarak orjinal pull request'e bunu ekliyor. Ve proje sahibi bunu pull request forumunda görüp merge edebiliyor.

Tüm bunlar pull request history'de olup bitiyor (yazışmalar vs.)

— Git 4 - In Class —

(Github'da dolaşırken 'clarusway' accountunda bir proje gördük)

- Clarusway adlı github'ı olan birinden fork ile repo-sunu kendi github' hesabımıza aktardık.
- Bu repoyu Bit Bash 'ten (klasör oluşturduktan sonra) clone yaparak working dir.'e ve local repo'ya aldık
- Sonra üzerinde çalışmak için new-feature adlı bir branch oluşturduk (Genelde farklı branch'ta çalışılır)
  Python.py dosyasında değişiklik yaptık, kaydedip çıktık.
- Fork yaptığımız orjinal repo ile connection sağlayıp ordaki son durumu almak istiyorum - Çünkü ben çalışırken değişiklik yapılmış olabilir. Bunu şöyle yapıyoruz;

  git remote add upstream <clarusway github repo URL'si>

  ↓ clarusway repository'sini upstream olarak bizimkiyle karışmasın diye tanımladık! upstream dedik origin demedik

- Şimdi remote repository'leri görelim ve az önceki işlemi doğrulayalım (verify)

  git remote -v

  (hem bizim remote repoyu hem de clarusway'in remote reposunu gördük)

~~Artık değiştirdiği kendi remote reposuna push edebiliriz~~

  git pull upstream main

  clarusway'in remote repo'sundan çektik.
  yeni değişiklik olmadığını gördük.
  artık yapığımız değişikliği kendi remote reposuna push edebiliriz.

  git push origin main → hata verdi çünkü remote repoda üzerinde çalıştığımız new-feature branchi yok!

  git push --set-upstream origin new-feature

  komutu ile push ediyoruz. (Origin'de new-feature branchi oluşuyor)

(6.4.19)