

Git 3. Ders

9 Kasım 2022 Çarşamba 19:08

Git ve Github farklı şeyler ancak birbirinden ayrı düşünmek de imkansızdır.

GitHub Reposu iki şey için önemli:

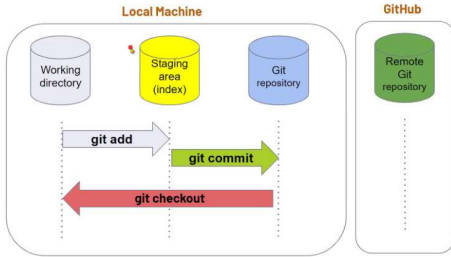
Birincisi projelerimizin %99 u source code ları remote repolarda olacak. Bu nedenle öğrenmemiz şart.

İkincisi iş başvurularında cv, linkedin hesapları (bu coğrafyada linkedin den işe giriliyor.) GitHub hesabımızı ve varsa kendimize ait bir web site ya da blog onun linkini isteyecekler.

Bu nedenle repository oluşturmak önemli.

Recap-Basic Commands

```
git init
git status
git add .
git rm --cached
git commit -m "abc"
git log
git checkout commitID
```

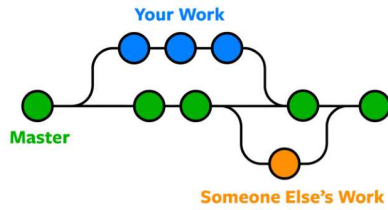


Lokalde bir klsörede git init yaparak lokalde repo oluşturuyorduk.

Diğer yöntemse GitHub da repo açmak ve lokale klonlamaktır.

Recap-Branches

```
git branch branch_name
git branch
git branch -r
git branch -a
git checkout branch_name
git checkout -b branch_name
git branch -d branch_name
git branch -D branch_name
git merge branch_name
```



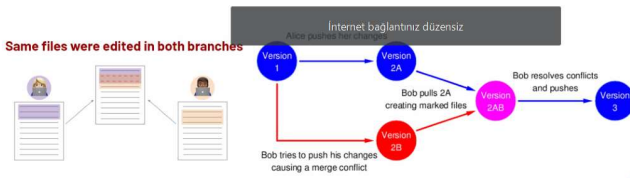
"git branch -r" remote taki branchları listeler

"git checkout -b" yeni bir branch oluşturur ve o brancha gider.

"git merge" branchları merge eder.

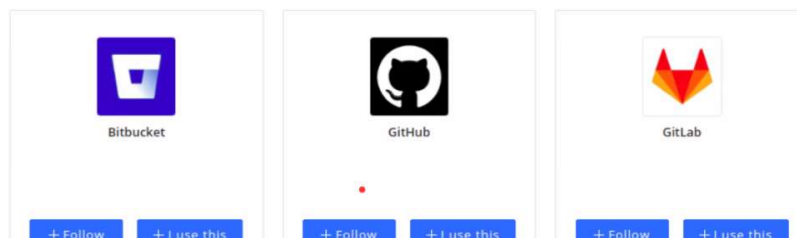
Merge Conflicts

Merge conflicts happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge.



GitHub arkada Git teknolojisini kullanıyor.

Github - Remote Repository



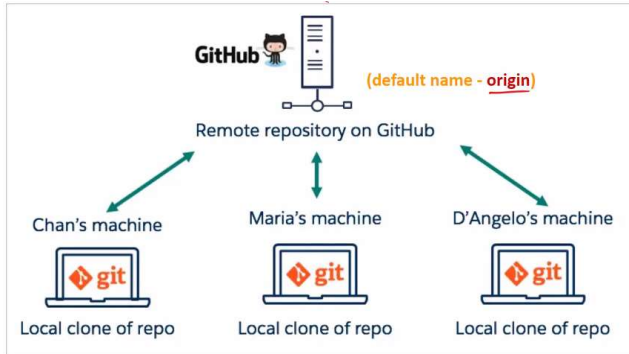
Stacks	Followers	Votes
25.8K	19.2K	2.8K

Stacks	Followers	Votes
132.1K	99.8K	10.1K

Stacks	Followers	Votes
30.5K	23.4K	2.3K

Bu versiyon kontrol sistemlerinden birisini biliyorsanız diğerini öğrenmek çok kısa sürüyor.

Artık bunlar arka planda devops fonksiyonlarını yapan toollar geliştirdiler.



"origin" lokaldeki reponun remote taki bağlı olduğu reponun default adıdır. GitHub repositüsü public ya da private olabilir. Private olursa kimse giremez. Public olursa başkaları girip klonlayabilir ancak bir şey gönderemezler.

Github - Remote Repository

- Act of copying a repository from remote server to your local machine is called **cloning**
- Cloning allows team to work together
- Downloading commits from others : **fetch, merge**
- Downloading commits from others : **pull (fetch + merge)**
- Uploading your commits (local changes) to remote : **push**

Remote taki repoyu lokal bilgisayara kopyalamaya klonlamak denir.

Forklama ise remote taki bir repoyu kendi remote repomuza kopyalamaktır.

"git fetch" remote repodan kullandığımız bir klasörde sonradan yapılan değişiklikler için kullanılıyor. Fetch remote repodaki değişiklikleri lokal repoya alıyor, sonra merge ederek working directoryde görebiliyoruz. Ancak sonradan merge etmemiz gerekiyor.

Fakat "git pull" komutu fetch ve merge komutlarını birlikte yapıyor.

Connecting your local with remote

- connect to remote repo

git remote add origin Repo address

git remote -v

origin = alias for your repo address

- first push

git push -u origin master

- remove remote origin

git remote rm origin

Lokaldeki repo ile remote taki repoyu bağlantılandırmanın yolu "git remote add origin repo address" komutu ile yapılır.

"git push -u origin master" komutu lokalde yapılan değişikliği remote göndermek için, "git remote rm origin" komutu da bağlantıyı silmek için kullanılır.

```

cyler@DESKTOP-MV83M01:~/Desktop$ cd Desktop/
cyler@DESKTOP-MV83M01:~/Desktop$ ls
clarusway-github/  desktop.ini  git-lesson/  my-works/  to-do-list.txt
./  classroom/  devops-prep/  junk/  mygithub/
cyler@DESKTOP-MV83M01:~/Desktop$

```

Desktop altında veya home directory de main ya da master yazıyorsa orada ls -a yapıp .git dosyasını bulup silmemiz gerekir.

Branch ı master dan main e default olarak değiştirmek için kullanacağımız kod:

"git config --global init.defaultBranch main"



Activity graph ne kadar çok commit ya da pull request gönderirseniz burası yeşilleniyor. Bu da sizin git le uğraştığınızı gösteriyor gibi bir algı var. Hoca çok önemli olduğunu düşünmüyor.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

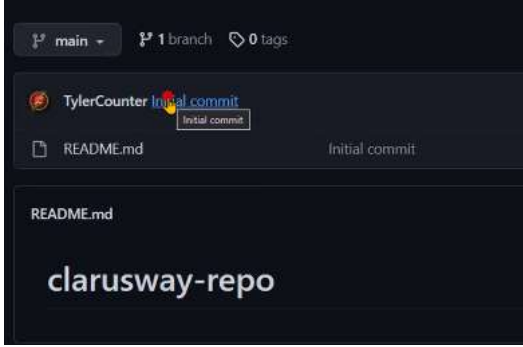
Proje ile ilgili uzun bir açıklama yapabileceğimiz kısım.

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)
.gitignore template: Terraform ▼

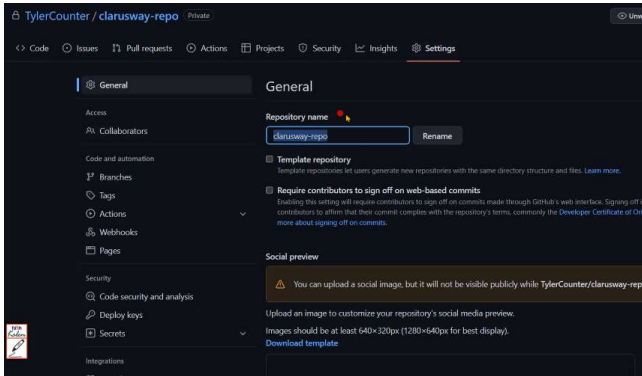
Neleri, hangi uzantılı dosyaları GitHub a göndermeyeceğimizi buradan seçiyoruz.

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)
License: None ▼

Hazır lisanslar var buradan seçebiliyoruz.



Biz commit yapmadığımız halde readme.md file oluştururken bu commiti otomatik yapıyor. Commitin üzerine (commit id si) bastığımızda detaylarını görebiliyoruz.



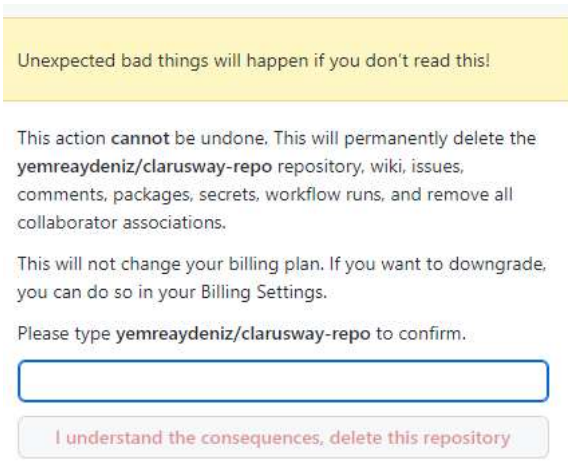
Bu repoyu silmek için Settings sekmesine gelip aşağı iniyoruz.

Danger Zone

Change repository visibility This repository is currently private.	Change visibility
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	Transfer
Archive this repository Mark this repository as archived and read-only.	Archive this repository
Delete this repository Once you delete a repository, there is no going back. Please be certain.	Delete this repository

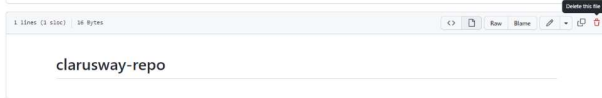
En aşağıda danger zone var. Delete repository var. Ona basınca;

Are you absolutely sure? [X](#)



Repositorynin ismini yazıp aşağıdaki butona bastığınızda silinir.

Sadece içindeki dosyayı silmek için dosyayı basıp sağ taraftaki çöp kutusuna basıyoruz:

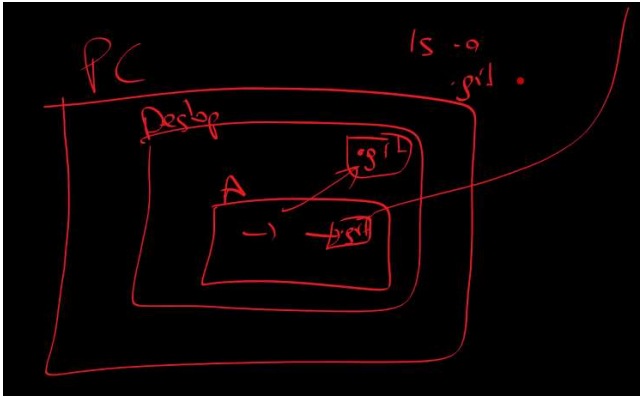


Reponun id si olmaz url si olur. Commit lerin id si olur.

Bu repoyu masaüstüne klonlamak istiyorum.

Biz klonlamayı token la yapmazsak sonradan push yapmak isterken sorun yaşarız.

Devops olarak çalışırken windows yok. Cloud bir bilgisayarda gecenin içinde pull ya da push işlemi yapılması gerekiyor. Kalkıp kullanıcı adı ya da token mi gireceğiz hayır. Daha önceden token la klonlama yaptığımız için yapacağı işlemi kendisi otomatik yapacak.



Desktopta .git klasörü olduğunda pull-push yaparken kesin hata alırsınız.

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$ ls -la
./ ./ .git/ README.md

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$ git remote -v
origin https://ghp_PeUGk19ksSqGtpLPiPqZ8r3X1qvzPJ1Fr7GW@github.com:clarusway/clarusway-repo.git
origin https://ghp_PeUGk19ksSqGtpLPiPqZ8r3X1qvzPJ1Fr7GW@github.com:clarusway/clarusway-repo.git

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$
```

Localdeki repo nun remote taki hangi repo ile bağlantılı olduğunu "git remote -v" komutuyla görüyoruz. Orada yazan origin de remote taki bu reponun default addıdır.

Bu ne sağlar? Push yaparken repo nun url sini yazmamıza gerek kalmayacak. "git push main" ya da "git push origin main" yazmamız yeterli olacak. Yani origin GitHub ın linki (origin in karşısında yazan) yerine kullanılan takma isimdir.

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(Use "git push" to publish your local commits)

nothing to commit, working tree clean

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$ git log --oneline
e25d8b4 (HEAD -> main) yeni dosyalar eklendi
9f0e3f8 (origin/main, origin/HEAD) Initial commit

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/clarusway-repo (main)
$
```

Local repoda clarusway-aws-devops-13 ten çektiğimiz dosyaları clarusway-repo ya ekleyip commit edince, bize "your branch is ahead of 'origin/main' by 1 commits" diyor. Yani origin den (remote dan) bir

commit öndesin.

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/c\larusway-repo (main)
$ git push
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (31/31), 18.66 MiB | 4.79 MiB/s, done.
Total 31 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/yemreaydeniz/c\larusway-repo.git
9f0e3f8..e25d8b4  main -> main

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/c\larusway-repo (main)
$
```

"git push" ile lokaldeki bu değişiklikleri remote a gönderiyoruz.

Git	yeni dosyalar eklendi	7 minutes ago
Python	yeni dosyalar eklendi	7 minutes ago
aws	yeni dosyalar eklendi	7 minutes ago
teamwork-agendas	yeni dosyalar eklendi	7 minutes ago
README.md	Initial commit	1 hour ago

GitBush a gidip sayfayı yenilediğimizde değişikliği görüyoruz.

Şimdi ise önce pc de "git init" ile bir repo oluşturacaz. Daha sonra GitHub da boş bir repo oluşturacaz. İçinde hiç commit olmaması gerekiyor. Readme de dahil. Çünkü commitlerin aynı histroyden gitmesi lazım. Çünkü bir commit diğerini referans gösteriyordu.

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop
$ cd git_lesson

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson
$ mkdir üçüncü-ders

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson
$ cd üçüncü-ders/
```

Desktop a gelip git_lesson un içine girdik ve orada üçüncü-ders isimli bir klasör oluşturduk.

Daha sonra GitHub a gidip yeni bir repository oluşturuyoruz ve readme eklemiyoruz:

yemreaydeniz / test-repo

Great repository names are short and memorable. Need inspiration? How about sturdy-octo-parakeet?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None

☒ You are creating a public repository in your personal account.

Create repository

Bu kez public repo oluşturduk. Ve boş olduğu için şöyle boş bir repo görüntüsü aldık:

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/yemreaydeniz/test-repo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

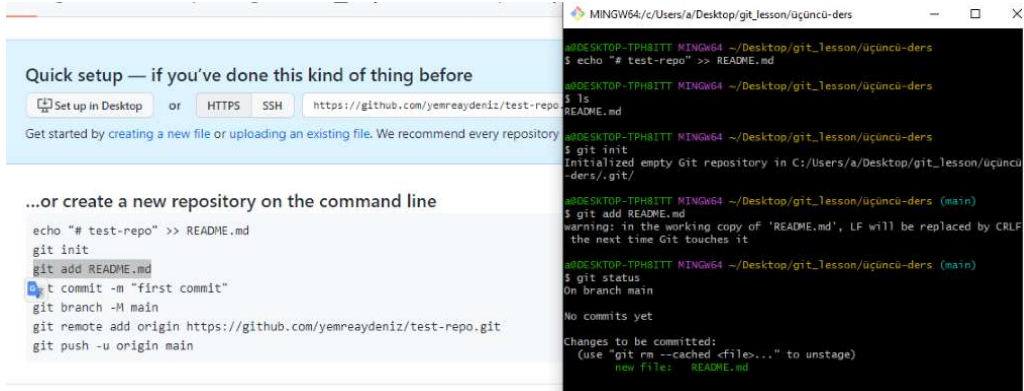
```
echo "# test-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/yemreaydeniz/test-repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/yemreaydeniz/test-repo.git
git branch -M main
git push -u origin main
```


...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

Şimdi üçüncü-ders klasöründe lokal repo oluşturup readme file ekleyerek remote repoya göndereceğiz:



Sol taraftaki kodları sırasıyla Git Bash te yazıyoruz. git branch -M main kodunu yazmamıza gerek yok zaten main branch tayız. Bu kod master ı main e çevirmek için kullanılır.

...or create a new repository on the command line

```
echo "# test-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/yemreydeniz/test-repo.git
git push -u origin main
```

Buradaki komutu çalıştırırken token kullanacağız:

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson/üçüncü-ders (main)
$ git remote add origin https://ghp_PeUGk19ksSqGtpLPiPqZ8r3XlqvzPJ1Fr7Gw@github.com/yemreydeniz/test-repo.git

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson/üçüncü-ders (main)
$ git remote -v
origin https://ghp_PeUGk19ksSqGtpLPiPqZ8r3XlqvzPJ1Fr7Gw@github.com/yemreydeniz/test-repo.git (fetch)
origin https://ghp_PeUGk19ksSqGtpLPiPqZ8r3XlqvzPJ1Fr7Gw@github.com/yemreydeniz/test-repo.git (push)
```

Git remote -v ile remote repoya bağlandığını görüyoruz. Burada git push komutunu kullandığımızda push yapmamıza izin vermez. Çünkü remote repoda hiç commitimiz yok:

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson/üçüncü-ders (main)
$ git push
fatal: The current branch main has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin main

To have this happen automatically for branches without a tracking upstream, see 'push.autoSetUpRemote' in 'git help config'.
```

0 nedenle solda işaretli kodu kullanıyoruz:

```
git branch -M main
git remote add origin https://github.com/yemreydeniz/test-repo.git
git push -u origin main
```

```
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/git_lesson/üçüncü-ders (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/yemreydeniz/test-repo.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
```

...or push an existing repository from the command line

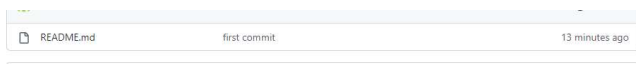
\$ git push --set-upstream origin main

Bu kod da "git push -u origin main" kodunun alternatifidir.

Son versiyonlarda git push origin main kodu da çalışıyor:

```
enes@DESKTOP-JNAPU0U MINGW64 ~/Desktop/example/ikinci_ders (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 516 bytes | 516.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/eneseren25/test-repo.git
 * [new branch] main -> main
```

Şimdi Github da sayfayı yenilediğimizde test-repo da readme file ı görebiliyoruz:



README.md

**test-repo**

Bir sonraki ders pull-request ve fork işleyeceğiz.

Token değiştirip yeni token aldıktan sonra daha önce oluşturduğumuz repolara yeni token ı tanıtmamız gerekiyor. Bunu aşağıdaki komutla yapıyoruz:

git remote set-url origin https://**token**@url