

# GIT

## Git Nedir?

Git free ve opensource bir versiyon kontrol sistemidir.  
kod daki değişiklikleri takip eden dağıtık bir sistemdir.

## Git Ne işe yarar?

- Revert files to previous state,
- Revert entire project back to previous state,
- Compare changes over time,
- See who modified what? **And much more...**

## Neden?

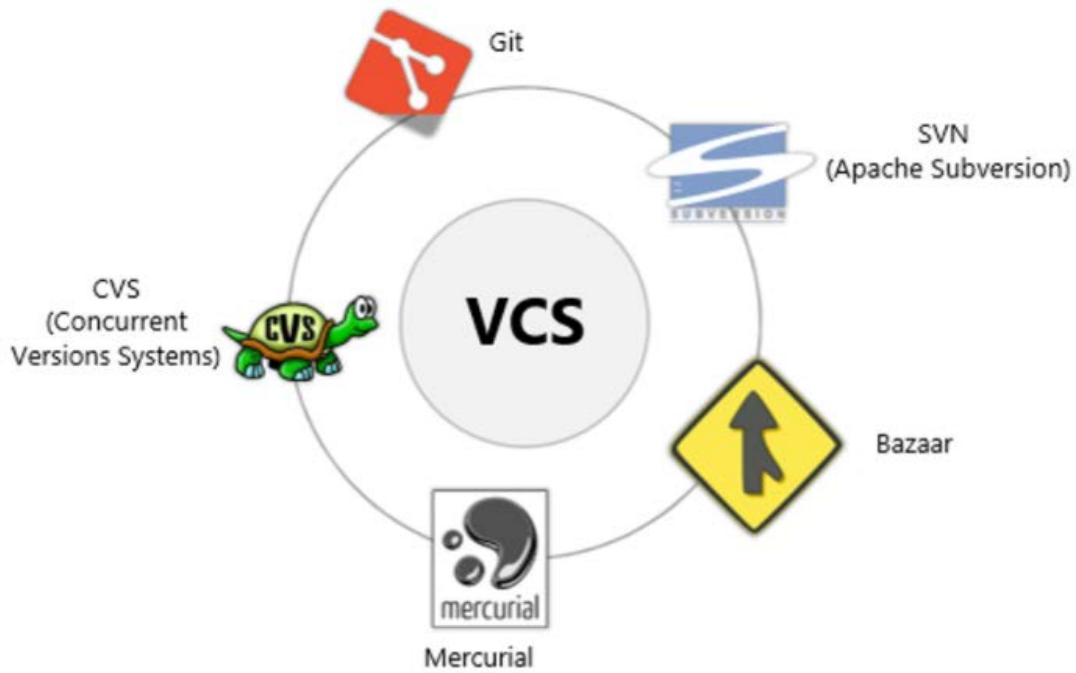
- Everything is local (full history tree available offline),
- Everything is fast,
- Snapshots, not diffs,
- It is distributed not centralized,
- Great for those who hate: CVS/SVN (earlier version control systems).

## VCS ne işe yarar?

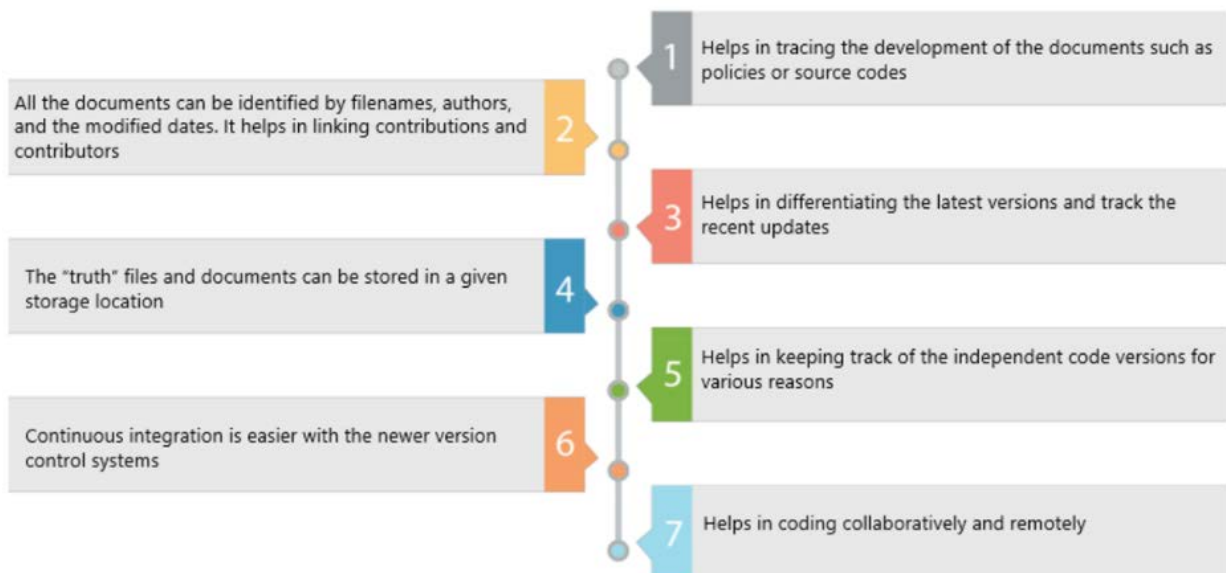
- What had changed
- When it changed
- Why it changed
- Who changed it

Dosyaların saklanması gerekir. VCS kullanırsanız kaybetmezsiniz, tüm versiyonlarına erişebilirsiniz.

Populer versiyon kontrol sistemleri:

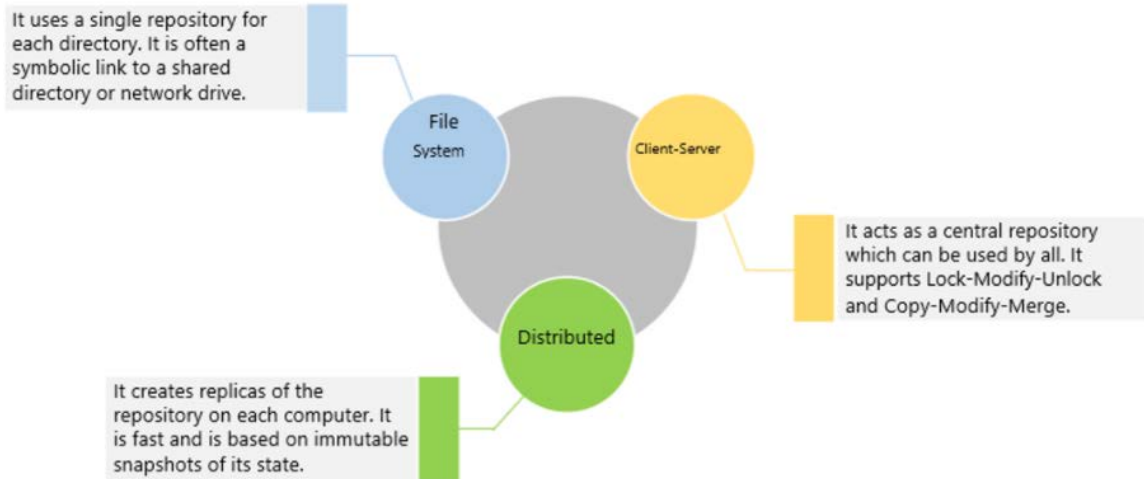


## VCS in Önemi:



## VCS Çeşitleri:

- File-based, eski
- Client-server type, git gibi
- Distributed.



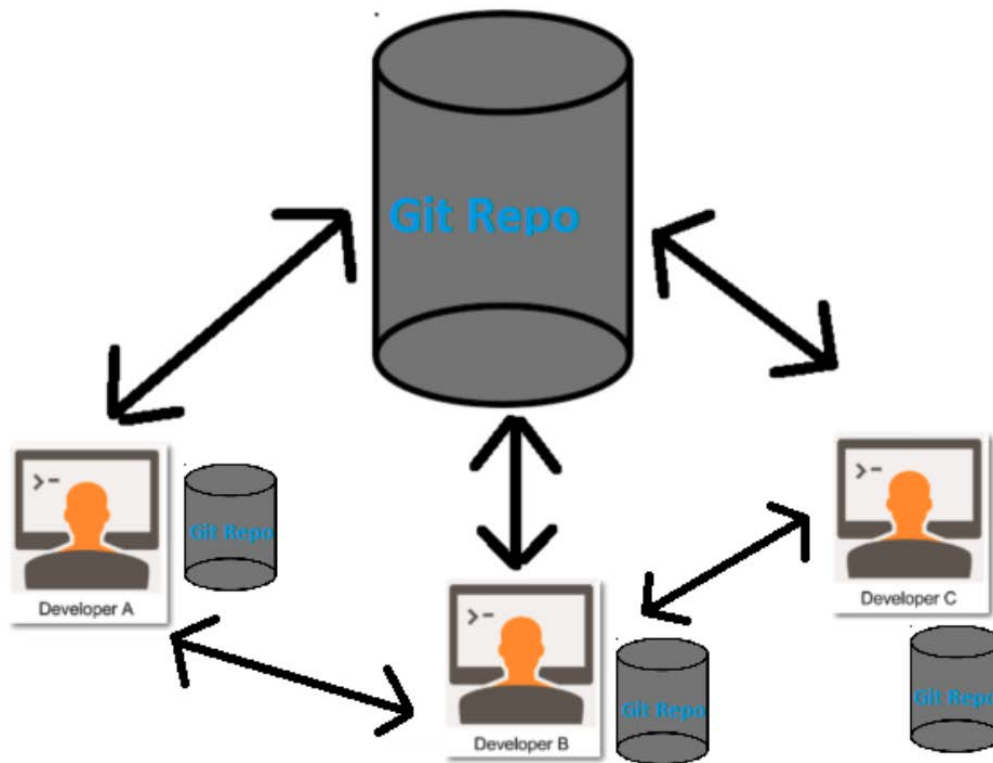
Client-Server tip iki çeşit:



Conflicts occur if two people change a file at the same time. This problem can be solved in two ways:

Lock-Modify-Unlock	Copy-Modify-Merge
<ul style="list-style-type: none"> <li>• Only one person can change a file at a time</li> <li>• A file must be locked before changing it</li> <li>• Different users can't lock a locked file</li> <li>• Another user must wait for the lock to be released</li> </ul>	<ul style="list-style-type: none"> <li>• Each user has a personal working copy of the file tree</li> <li>• Changes are made independently and simultaneously</li> <li>• Private copies are merged into a new version</li> <li>• All other copies become outdated as soon as a private copy is committed</li> </ul>

Distributed sistemlerde her kullanıcı için bir kopya oluşturulur. Ağ bağlantısı kopsa bile devam eder.



Here are some features of Distributed VCS.



Repository:

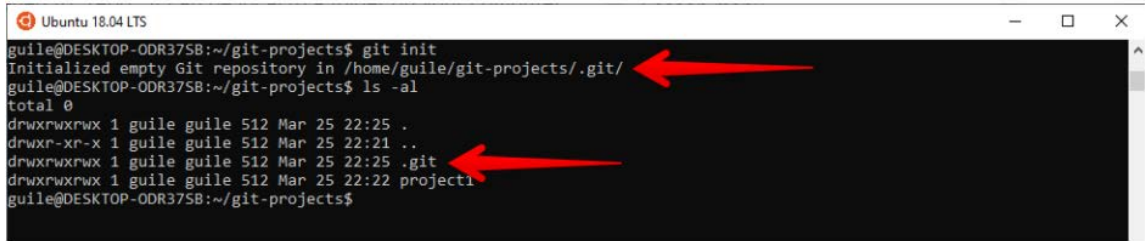
Projeler için dizin veya depolama alanı

Localde veya bulutta olabilir.

Git başlatma komutu:

```
$ git init
```

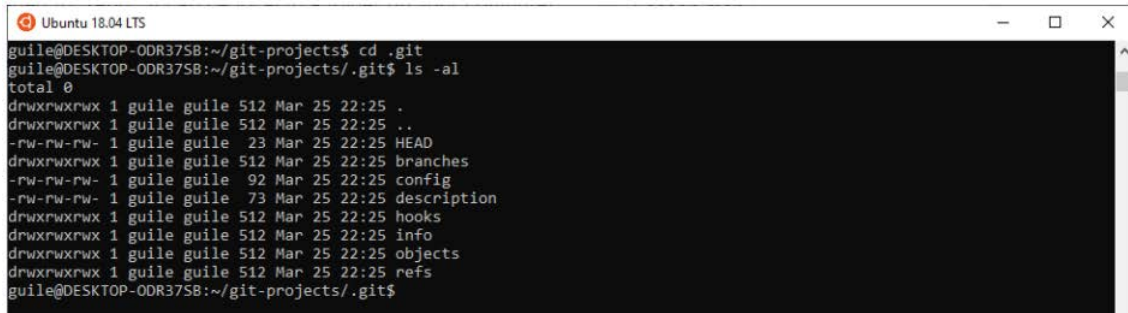
```
$ ls -al
```



```
Ubuntu 18.04 LTS
guile@DESKTOP-ODR37SB:~/git-projects$ git init
Initialized empty Git repository in /home/guile/git-projects/.git/
guile@DESKTOP-ODR37SB:~/git-projects$ ls -al
total 0
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 .
drwxr-xr-x 1 guile guile 512 Mar 25 22:21 ..
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 .git
drwxrwxrwx 1 guile guile 512 Mar 25 22:22 project1
guile@DESKTOP-ODR37SB:~/git-projects$
```

```
$ cd .git
```

```
$ ls -al
```



```
Ubuntu 18.04 LTS
guile@DESKTOP-ODR37SB:~/git-projects$ cd .git
guile@DESKTOP-ODR37SB:~/git-projects/.git$ ls -al
total 0
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 .
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 ..
-rw-rw-rw- 1 guile guile 23 Mar 25 22:25 HEAD
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 branches
-rw-rw-rw- 1 guile guile 92 Mar 25 22:25 config
-rw-rw-rw- 1 guile guile 73 Mar 25 22:25 description
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 hooks
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 info
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 objects
drwxrwxrwx 1 guile guile 512 Mar 25 22:25 refs
guile@DESKTOP-ODR37SB:~/git-projects/.git$
```

Git Takip Süreci:

# Workflow



git repoda dosya 3 aşamada bulunabilir. Modified, Staged, and Committed.

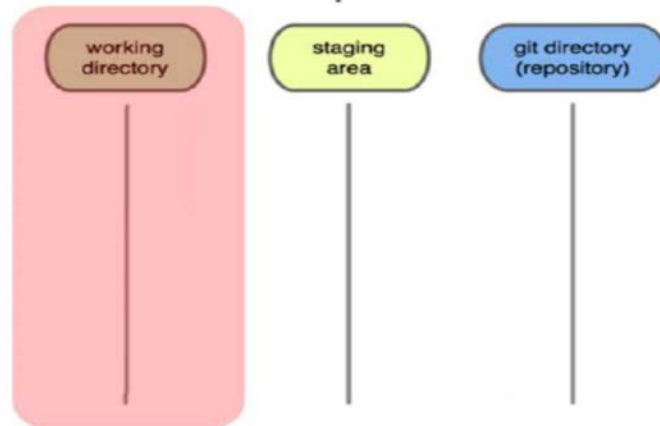
- **Modified** means that you have changed the file but have not committed it to your database (repo) yet.
- **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
- **Committed** means that the data is safely stored in your local database.

git projesinin 3 ana bölümü

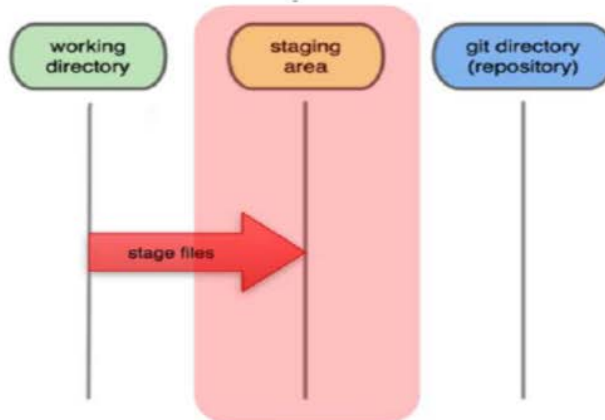
- The working tree,
- The staging area,
- The Git directory.

---

You modify files in your working directory.

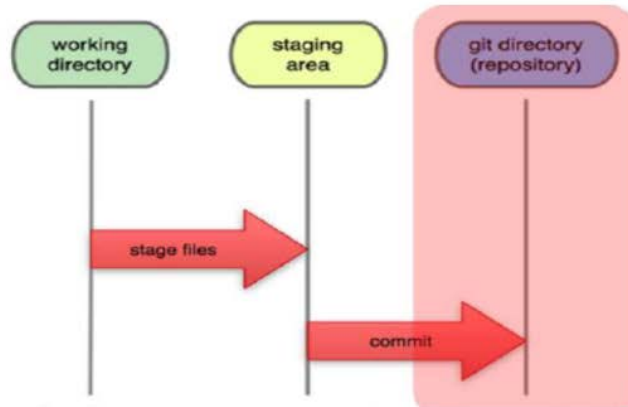


After you are done with your file in your working directory you add them to staging area, you can think of this as a temporary storage for your files. You stage the files, adding snapshots of them to your staging area.

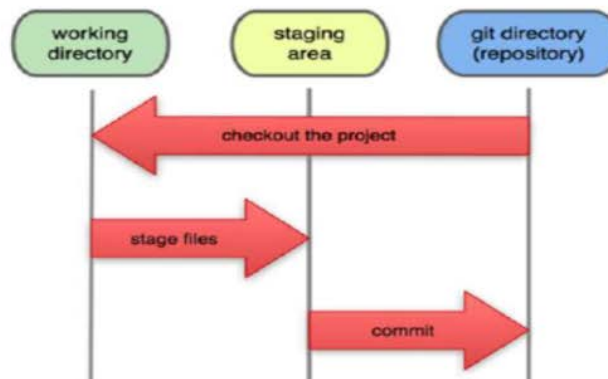


Alışveriş yapıyorsun sepete doldurdun sepet staging area. daha almadın değiştirebilirsin.

Then you are sure that the files you have in your staging area are ready to go next step then you commit your files to your git local repo. You do a commit that stores snapshot permanently to your Git directory.



Once you commit your files to your local repo you can then checkout them at any time you can modify them then add and commit again as you wish.



Örnek ;

1-) Takip etmek istediğin klasörü localde oluşturmaya başla.

```
$ touch index.html file.txt cprog.c javaprogram.java index.js style.css type.ts  
$ ls -al
```



```
guile@DESKTOP-ODR375B: ~/lab1.1
guile@DESKTOP-ODR375B:~/lab1.1$ touch index.html file.txt cprog.c javaprog.java index.js style.css type.ts
guile@DESKTOP-ODR375B:~/lab1.1$ ls -al
total 0
drwxrwxrwx 1 guile guile 512 Jan  8 12:50 .
drwxr-xr-x 1 guile guile 512 Jan  8 12:50 ..
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 cprog.c
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 file.txt
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 index.html
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 index.js
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 javaprog.java
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 style.css
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 type.ts
guile@DESKTOP-ODR375B:~/lab1.1$
```

2-) Oluşan klasörde takip sistemi yani git başlatma.

\$ git init (o klasörde vcs başlatır)

```
guile@DESKTOP-ODR375B: ~/lab1.1
guile@DESKTOP-ODR375B:~/lab1.1$ git init
Initialized empty Git repository in /home/guile/lab1.1/.git/
guile@DESKTOP-ODR375B:~/lab1.1$
```

3-) Check git status.

\$ git status

```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        cprog.c
        file.txt
        index.html
        index.js
        javaprog.java
        style.css
        type.ts

nothing added to commit but untracked files present (use "git add" to track)
```

kırmızılar çalışma alanında duruyor. git takip etmiyor

4-) "Git add" to add the files to staging area from the working area.

\$ git add .

And now we can see the new status of the files.(Notice green color)

git add dosya\_adı commite hazır.

\$ git status

```
guile@DESKTOP-ODR375B: ~/lab1.1
guile@DESKTOP-ODR375B:~/lab1.1$ git add .
guile@DESKTOP-ODR375B:~/lab1.1$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   cprog.c
        new file:   file.txt
        new file:   index.html
        new file:   index.js
        new file:   javaprogram.java
        new file:   style.css
        new file:   type.ts

guile@DESKTOP-ODR375B:~/lab1.1$
```

## Stage files options

- stage one file  
**git add filename**
- stage all files (new, modified)  
**git add .**
- stage all changes  
**git add -A**
- stage modified and deleted files only  
**git add -u**

git restore —staged dosya\_Adı →stage i geri alma

5-) Let's commit our files to local repo with the command

```
git commit -m "message/explanation"
```

\$ git commit -m "This folder includes demo files"

```
guile@DESKTOP-ODR375B: ~/lab1.1
guile@DESKTOP-ODR375B:~/lab1.1$ git commit -m "This folder includes demo files"
[master (root-commit) dd71872] This folder includes demo files
7 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 cprog.c
create mode 100644 file.txt
create mode 100644 index.html
create mode 100644 index.js
create mode 100644 javaprogram.java
create mode 100644 style.css
create mode 100644 type.ts
guile@DESKTOP-ODR375B:~/lab1.1$
```

# Commit

- Commit the files on the stage  
**git commit -m "message"**
- Add and commit all tracked files  
**git commit -am "message"**
- amend commit message  
**git commit --amend**

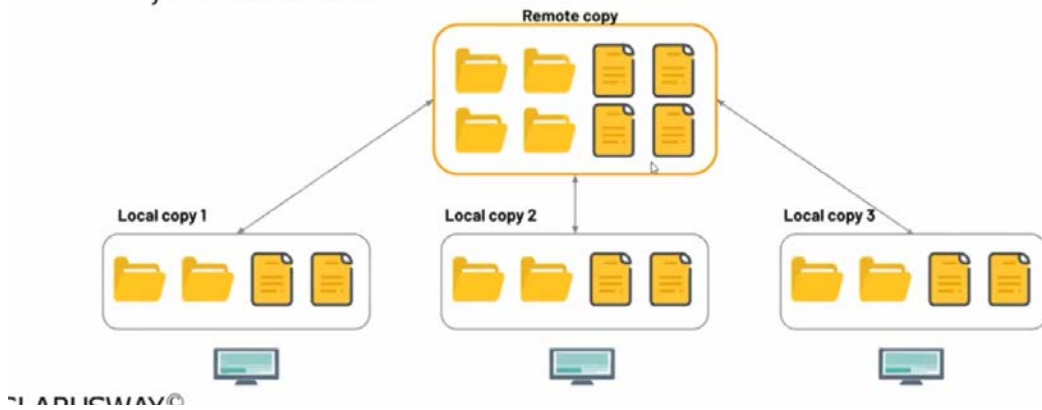
git commit -am "message" direk değişiklikleri stage e atmadan commit etme.

git commit --amend commit mesajı değiştirme.

Backuplama mantığı ve aynı dosya üzerinde birden fazla kişi çalışırsa doğabilecek sorunlar;

## Backup

- In any case if your remote server crashes, a backup is available in your local servers.



1 ve 3 değişiklik yaptı remote a gönderdiğinde proje yöneticisi karar veriyor.

Konfigürasyon ayarları;

## Git Repository

- Let's check if you have git in your computer

```
git --version
```

- git needs your identity to mark/label changes / editor

```
git config --global user.name "Your Name"
```

```
git config --global user.email "Your Email"
```

```
git config --global core.editor "vim"
```

```
git config --list
```

ARUSWAY®  
TO REINVENT YOURSELF

Working directory de bir değişiklik yapıldığında ;

git status yazınca kırmızı gözükecek

git add new\_file2 yapınca stage area ya alır ve ststus yapınca yeşil olur.

birden fazla dosyayı aynı anda stagein areaya koymak için git add .

git add -u;

değiştirilen ve silinen dosyaları staging areaya atmak için.

git log ;

yapılan tüm commitleri görebilen komut.

```
Samet USTAOGU@DESKTOP-SL2E65K MINGW64 ~/Desktop/git_lesson/work1 (master)
$ git log
commit c02e9050400cfbaa740b8d64288267abf2ad47d1 (HEAD -> master)
Author: sametusta <ustaoglusamet05@gmail.com>
Date:   Fri Jul 9 08:54:24 2021 +0300

    new file 1 üzerine ilk satır eklendi

commit ae0952e572b756db371917230071782ce38b1269
Author: sametusta <ustaoglusamet05@gmail.com>
Date:   Fri Jul 9 08:47:10 2021 +0300

    first commit
```

sarı hash kodları ile commitleri takip edebilirsiniz.

eski komuta gelmek için git checkout hash kodunun ilk 5 hanesini gir

git checkout ae0952e5

git checkout ile hem ileri hem de geri gelebilirsiniz.

HEAD hangi commit de olduğumuz ifade eder.

## GITHUB

GitHub is a Git repository hosting (Source Code Hosting) service

### What is the difference between Git and GitHub?

Git is a version control system that lets you manage and keep track of your source code history locally. GitHub is a cloud-based hosting service that

lets you manage Git repositories.

GitHub is the most popular one as you see

Name	Users	Projects
GitLab	100,000	546,000
Bitbucket	5,000,000	Private
SourceForge	3,700,000	500,000
launchpad	3,965,288	40,881
GitHub	24,000,000	69,000,000

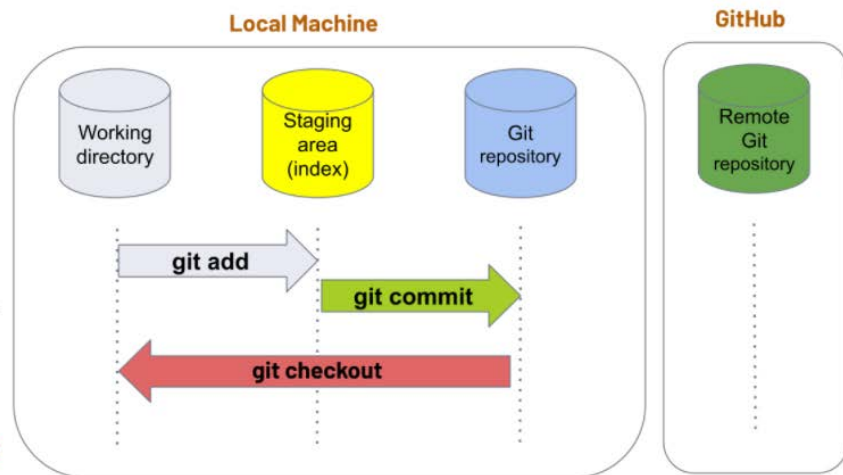
git config

git config list ile config bilgilerini görebilirsiniz.

```
Samet USTAAGLU@DESKTOP-SL2E65K MINGW64 ~/Desktop/git-lesson-3 (master)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=SametUSTAAGLU
user.email=ustaoglusamet05@gmail.com
core.editor=vim
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

## ► Recap-Basic Commands

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

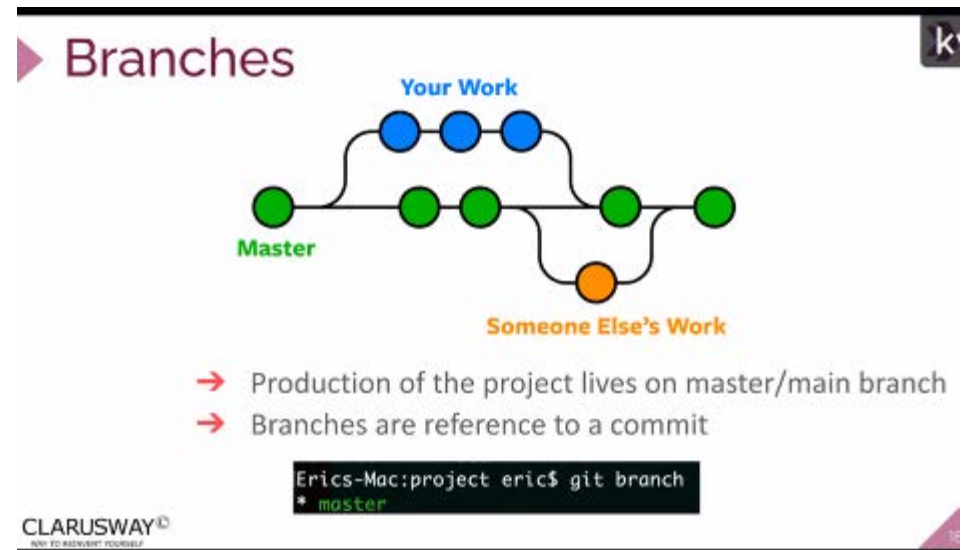
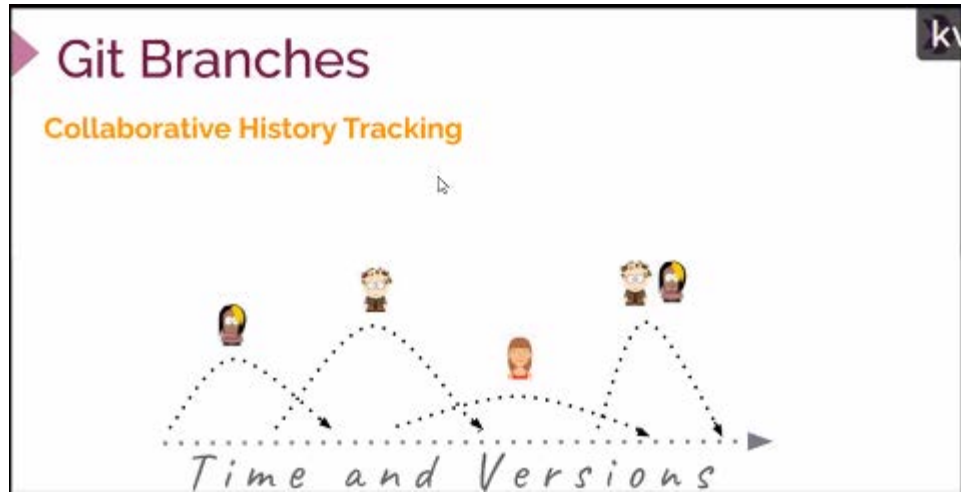


CLARUSWAY®  
WAY TO REINVENT YOURSELF

BRANCH

Aynı projede birden fazla grup çalışması için herkes ayrı çalışıyor en sonda birleştirecek.

hangi branch de iseniz head orayı gösterir.



Branch komutları:



## ► Creating/switching branches

→ create a new branch

```
git branch Branch name
```

→ switch to a branch

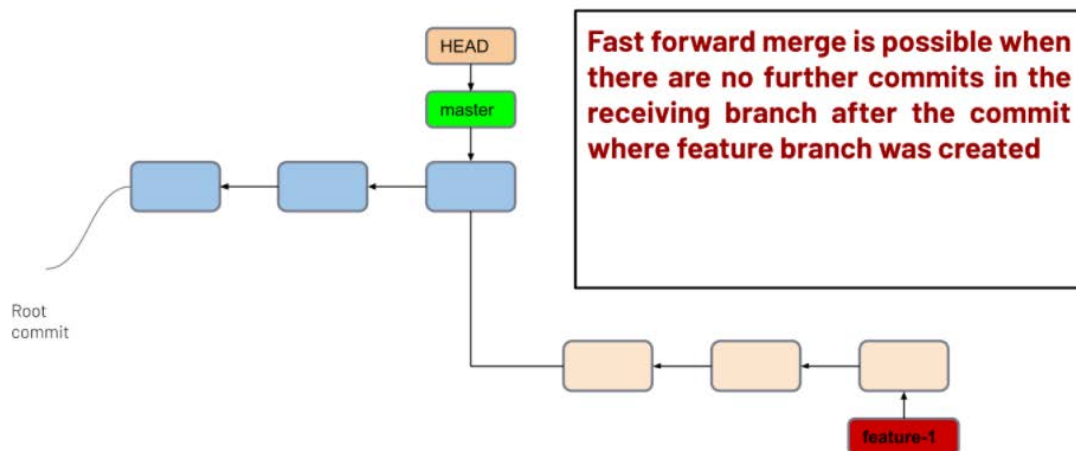
```
git checkout Branch name
```

→ create a new branch and switch to that branch

```
git checkout -b Branch name
```

git checkout -b newbranch3 yaratır ve geçer.

## ► Fast forward merge



nerde iseniz ordaan birleştirmek istediğiniz yeri seçiyorsun

aynı dosya üzerinde farklı değişiklikler varsa conflict verir. bunu proje yöneticisi çözer.

bir grup hiç değişiklik yapmadan diğer kol yaparsa fast forward merge oluyor.

```
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 log --pretty=oneline
```

```
git log --oneline
```

# Connecting your local with remote

→ connect to remote repo

```
git remote add origin Repo address
```

origin = alias for your repo address

→ first push

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

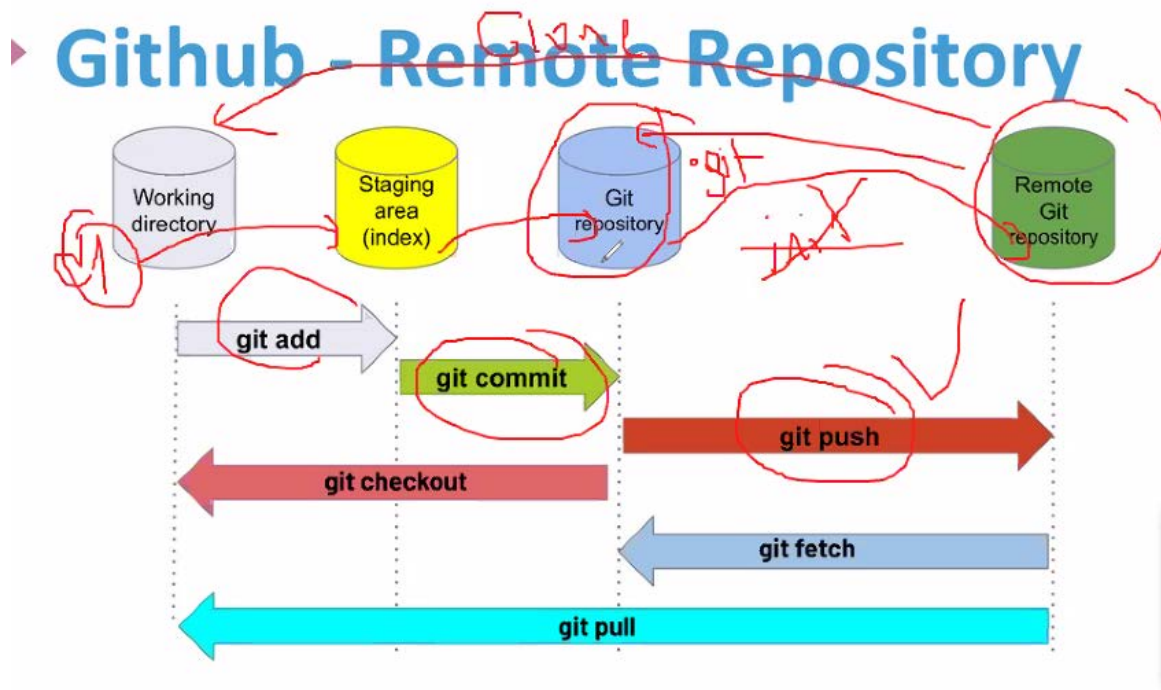
→ remove remote origin

```
git remote rm origin
```

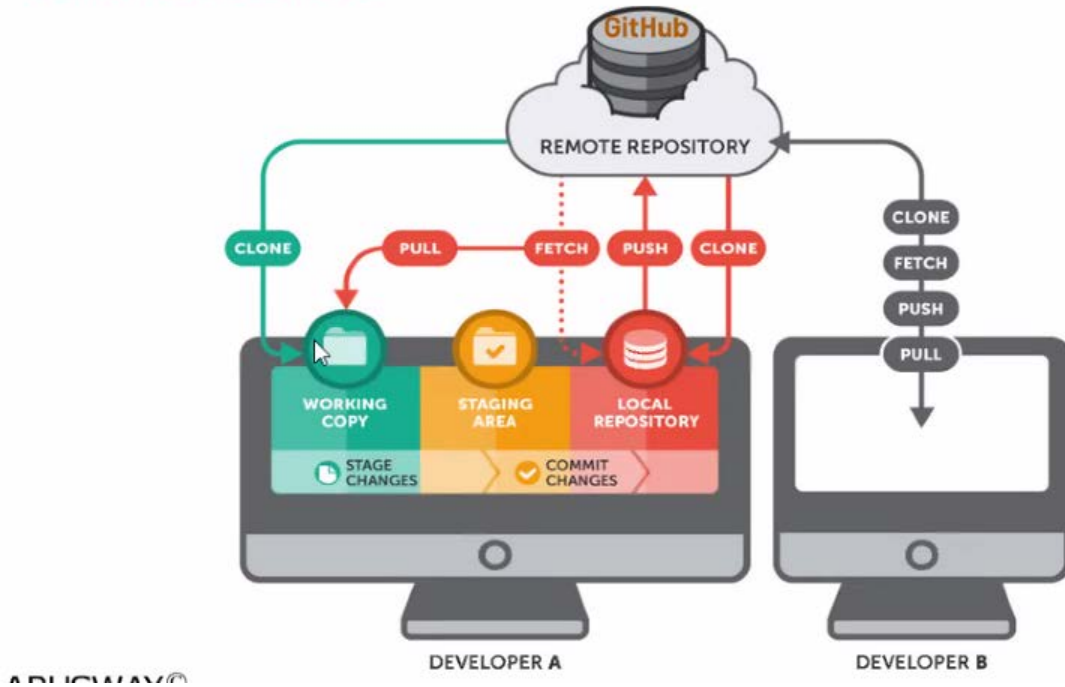


ARUSWAY©

önce clone ile çek daha sonra pull ,le devam



# Git Basics



## UZAKTAKİ REPOYU(PRIVATE) KENDİ GİTHUB İMIZA EKLEME

1-) Öncelikle bilgisayarınızda uygun bir yere klasör oluşturun. O klasöre uzaktaki repoyu clone edeceğiz.

2-)klasörün içinde iken git bash çalıştırıp

git clone <uzak repo linki> komutunu çalıştırın.

dosyalar klasörünüze inmiş olacak.

Bu adımda yüklemek istediğimiz repoyu bilgisayara indirdik.

3-) Kendi Github hesabınızda yeni bir repo create edin. oluşan reponun url sini kopyalayın.

4-) git clone <uzak repo linki> komutunu çalıştırın.

dosyalar klasörünüze inmiş olacak.

Bu adımda boş yeni yarattığımız repoyu bilgisayara indirdik.

5-) 2 numaralı adımda oluşan dolu klasörün içindeki dosyaları .git hariç kopyalayıp 4. adımda oluşturduğumuz kendi klasörümüze kopyalayın.  
Dikkat!! .git kopyalanmamalı.

6-) yapıştırdığımız yani kendi klasörümüzde dosyaları göndermeden önce add ve commit adımlarını yapmamız gerekiyor.

git add .

git commit -m "mesaj"

7-) dosya push edilmeye hazır.

git push

8-) sayfayı yenilediğinizde reponun dolduğunu görebilirsiniz.

```
github girdik
aşağıdaki adrese girdik
https://github.com/clarusway/clarusway-it-fundamentals-9-21
yeşil üzerinde code yazan butonu tıklıyoruz
kopyalama butonu tıklıyoruz
masaüstünde bir klasör cw-repo adında oluşturun
bunun içine git bash ile girin
git clone kopyalanan_url
tekrardan masaüstünüze girip my-repo adında klasör oluşturun
bu klasöre de git bash ile girin
kendi reponuzun url ini yukarıdakine benzer şekilde kopyalayın
git clone kendi_url
```

```
cw-itf.. klasörü içerisindeki klasör ve dosyaları kopyalayıp yeni clone ettiğiniz  
kendi repo klasörünüze yapıştırın(.git klasörü hariç)
```

```
kendi repo klasörünüzde git bash ile girin
```

```
git add .
```

```
git commit -m "initial commit"
```

```
git push
```

C

## **Extra**