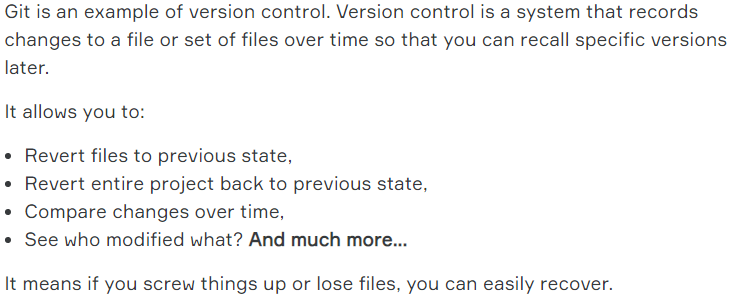
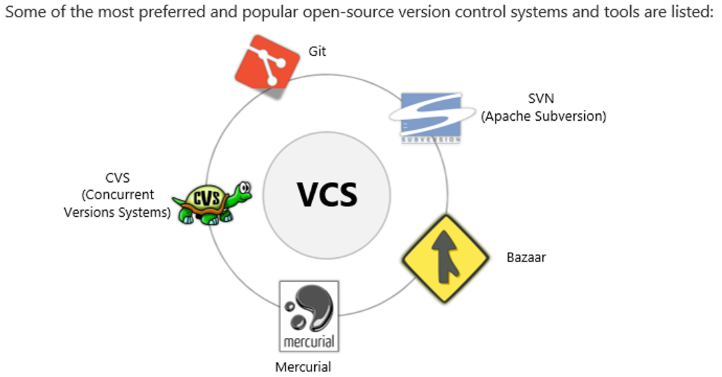
**GIT**

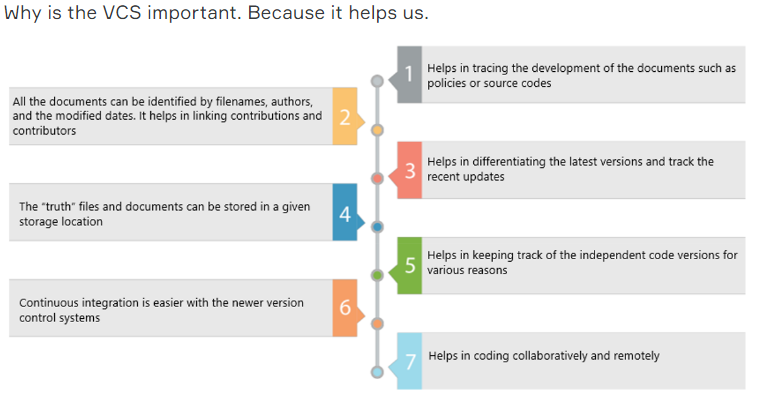
1. GIT:

Once GITHUB’taki reponun URL’sini kopyalayip klonlariz. Sonra bilgisayarda bir konuma gideriz linux ile ve asagidaki komutlari yazariz.

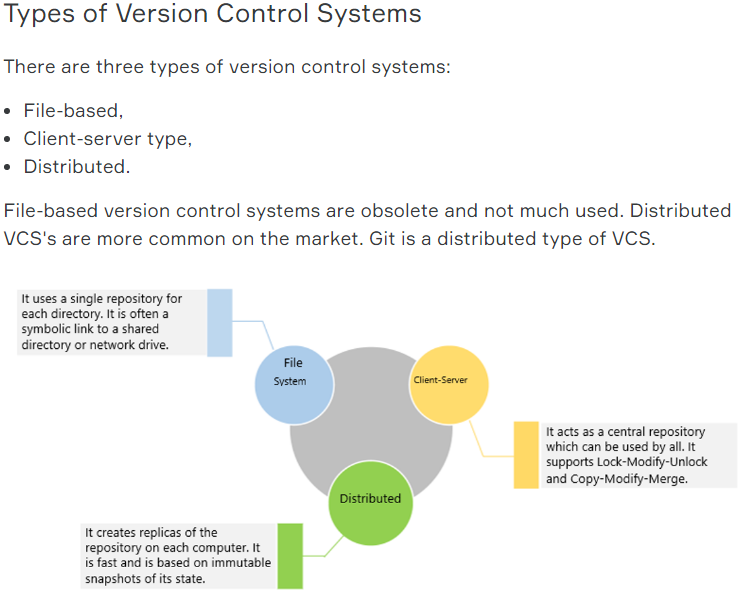
|  |  |
| --- | --- |
| **git clone URL** |  |
| **git add .** |  |
| **git status** |  |
| **git commit -m «yorum»** |  |
| **git push / pull** |  |

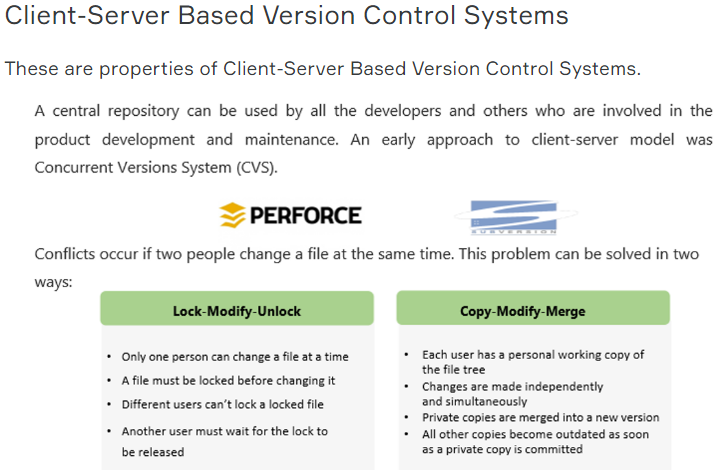


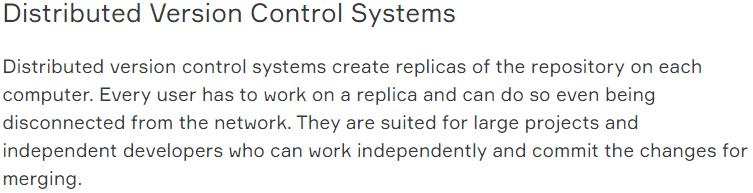




1. **VERSION CONTROL SYSTEMS:**







1. **DOWNLOAD ve INSTALL GIT:**

Windows:

<https://www.git-scm.com/download/win>

or

[https://www.gitforwindows.org](https://www.gitforwindows.org/)

MAC:

<https://git-scm.com/download/mac>

Linux:

For Ubuntu:

$ sudo apt-get install git

For Fedora (works also on **AWS Linux 2** Free Tier AMI):

$ sudo yum install git

1. **KULLANIM:**

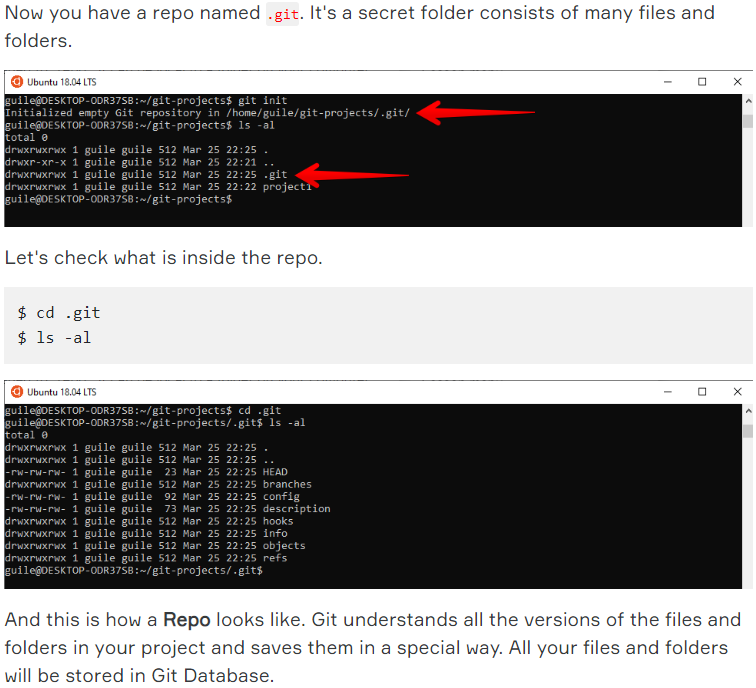
|  |  |
| --- | --- |
| **git --version** | Versiyonu gosterir. |
| **git config --global user.name «Umit Ceylan»** | Kullanıcı adı tanımlama. Basta tanimlanmali. |
| **git confıg --global user.email «byildiz2019@gmail.com»** | Emaıl tanımlama. Basta tanimlanmali. |
| **git config –global core.editor «text editor»** | Text editor tanimlama. Zorunlu degil. |
| git config --global color.ui auto | Renk ayarına ızın verır. |
| git config --list | Settingsleri kontrol eder. |
| q | Exit. |

* 1. **Folder Olusturma:**

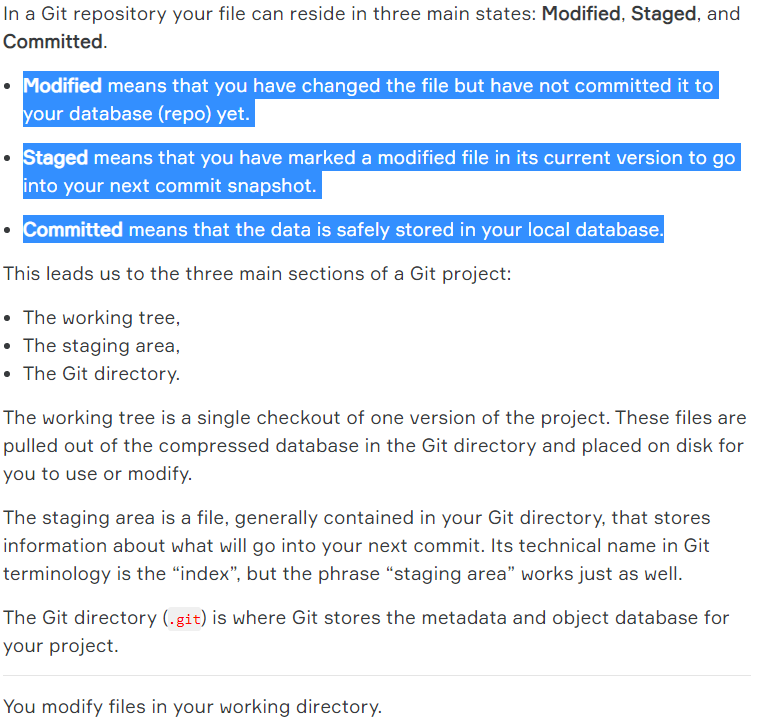
|  |  |
| --- | --- |
| mkdir git-projects | mkdir: dosya olusturma |
| cd git-projects | Dosyaya gir |
| **mkdir lab1.1** | Icinde ayri bir dosya olustur. |
| **cd lab1.1** |  |
| ls -al | git-projects folderinde hangi gizliler dahil dosya var. |

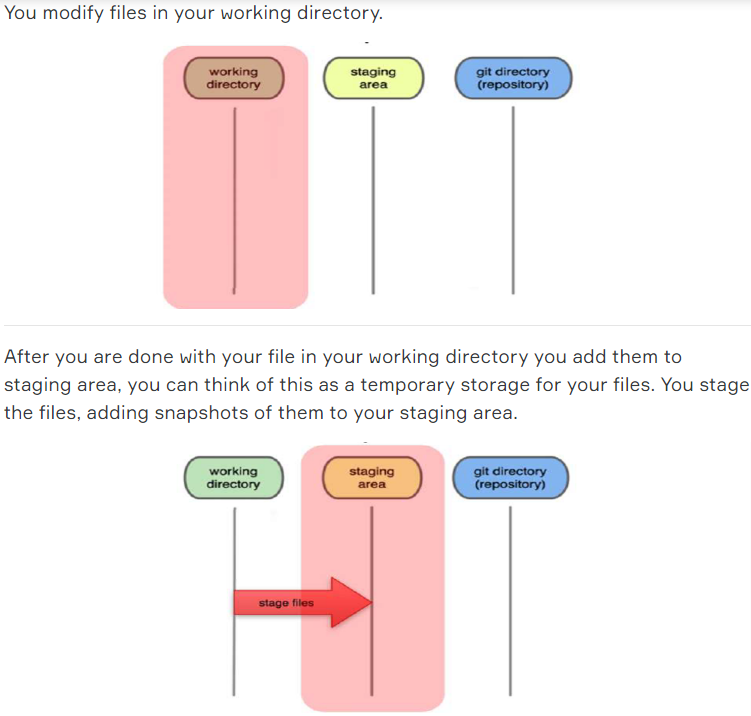
* 1. **Repo Olusturma:**

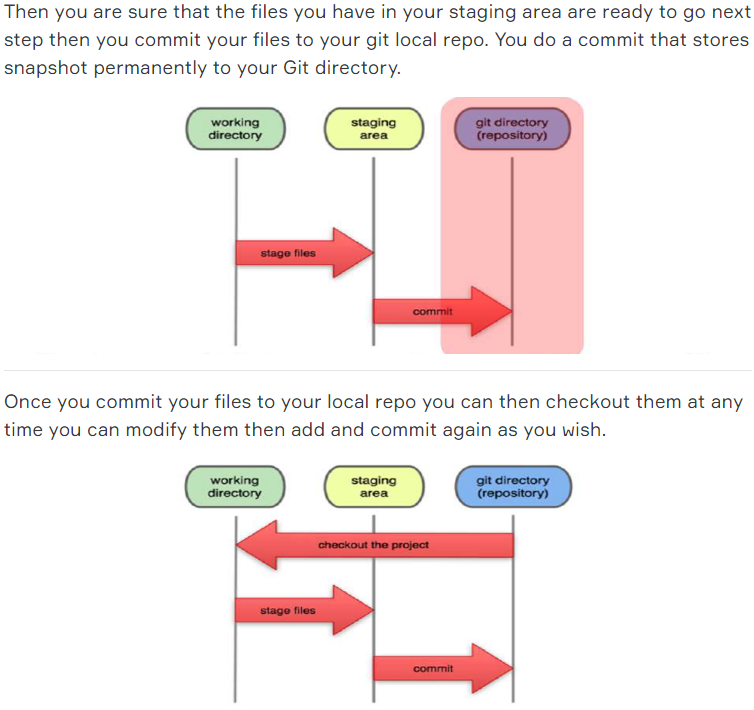
|  |  |
| --- | --- |
| **git init** | Repo type olusturma. .git gizli repo olustu. Soz konusu repo GITHUB’daki repoya verilen ayni isimdeki bir dosyanin altina olusturulur. |
| ls -al | Updated files gorme. |
| cd .git | Girilir. |
| ls -al | Acilir. |



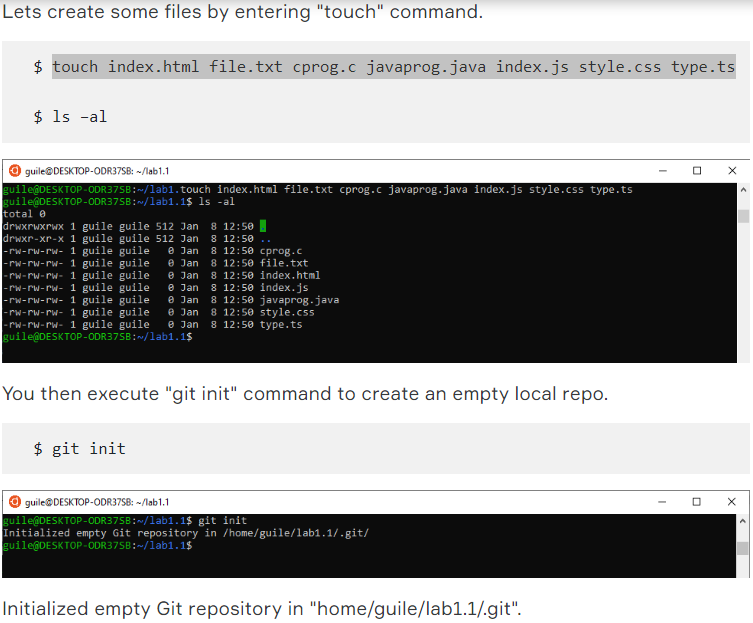
* 1. **GIT Datayi Nasil Yonetir:**

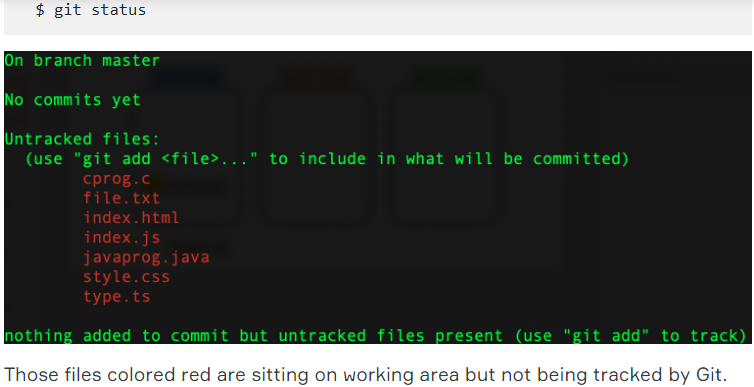


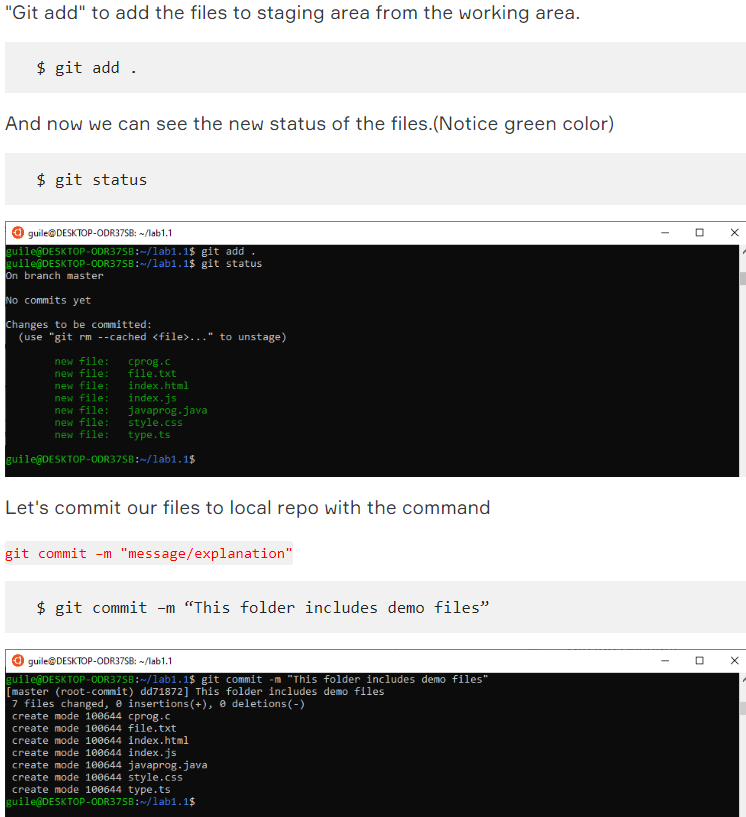




|  |  |
| --- | --- |
| **touch** index.html file.txt cprog.c javaprog.java index.js style.css type.ts | touch: Fileler olusturma. |
| ls -al |  |
| **git init** | Repo olusturduk. |
| **git status** | Repodaki working, staging vb alanlarindaki dosyalari gosterir. Kirmizilar working area’da yesiller staging areadadir. |
| **git add .** | Working areadaki fileleri staging areaya ekler. |
| **git commit -m ‘Kisisel dosyalar’** | Yorum eklemeye yarar. |
| **git commit -am «Kisisel dosyalar»** | Yorum yazip ayni anda add yapar. |
| git commit –ammend «Degisiklik» | Yorumu degistirmeye yarar. |
| git commit | Historynin bir snapshotini gosterir. |
| git diff | Repodaki herhangi Iki referans arasindaki farki gosterir. |
|  |  |







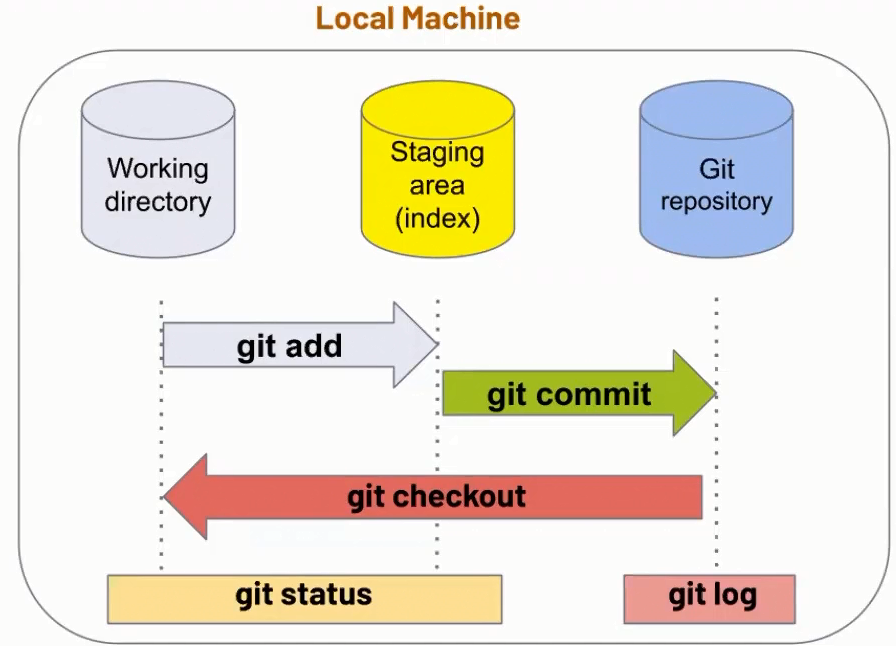
* 1. **GIT-GITHUB BAGLANTISI KURULUMU EN BASTAN:**

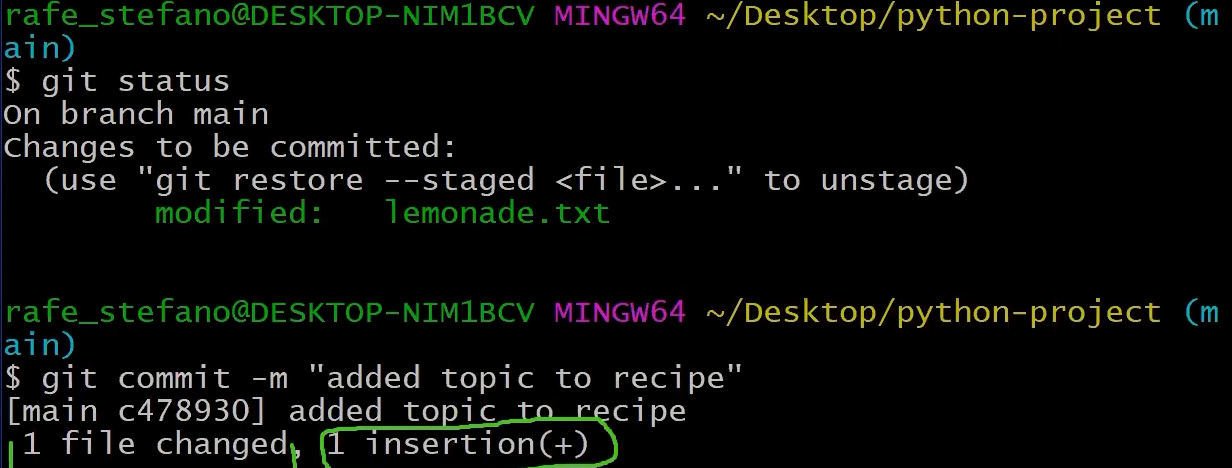
|  |  |
| --- | --- |
| **mkdir lab1.1** | Bir dosya olusturulur. (veya asagidaki islem yapilir.) |
|  | Bilgisayarda GITHUB repo ile ayni isimde bir dosya olusturur. > RC > GitBash here > GitBash acilir. |
| **cd lab1.1** | Dosyanin icine girilir. |
| **git init** | Local repoyu olusturduk. Olusturulan .git klasoru tum versiyonlari lokalde saklar. |
| ls -al | Gizli dosyalar dahil goruruz. |
|  | GITHUB > (bir) Repo > Code > Copy |
| **git clone URL** | Remote Github hesabindaki repoyu klonladik. Calistigimiz sirket reposuna ulasiriz. Kodlara ulasilir. |
| **git remote add origin URL** | **Local repoyu remote repoya baglantiyi kurariz. (Klonlama yapmadiysak.) Bu usul localde olusturulan dosyalari gonderirken tercih edilir.** |
| **git push -u origin master** | **git remote ile baglanti kurulursa ilk gonderim bu komutla olur.** |
| **git push filename** | **Ilk gonderimden sonra (git remote komutu kullanilirsa) artik sadece git push kullanilir.** |
| ls | Indirilen dosyalari goruruz. |
| ls -la | Gizliler dahil yetkilendirmeleriyle dosyalari goruruz. |
| **git add filename** | Kendi olusturdugumuz ilgili dosyayi Working’ten Stage seviyesine kaydeder, tracklar. **(Ayrica dosyada sonradan yapilan degisimler kirmizi working areada gorulur. Bunlar da git add ile stage seviyesine gonderilir.)** (Local seviyesi islem) |
| **git add .** | Tum dosyalari Working’ten Stage seviyesine kaydeder.  (Local seviyesi islem) |
| **git commit -m «message»** | Commit the files on the stage. Yorum ekler. |
| **git commit -am «mess»** | Stage’ten Local Repository’e dosyayi kaydeder. Ayrica yorum icerir.  (Local seviyesi islem) |
| **git commit --ammend** | Modify the most recent commit. |
| **git log** | Repository’deki herbir commiti gormek icin kullanilir. Kim, ne zaman ve commitleri (yorumlari, degisimin amacini) gosterir.  (Local seviyesi islem) |
| **git log --oneline** | Repository’deki herbir commiti gormek icin kullanilir. HASH’i 7 karakterli gosterir ve commitleri gosterir. |
| **git diff c478..36be** | HASH numarasi bazinda iki dosyanin farkini gosterir, HASH 4 karakter yazilsa yeter. |
| **git rm –cached filename** | Staging Area’dan Working areaya dosya remove edilir. Ilgili istenmeyen dosya atilir. |
| **git checkout** | Repodan dosyayi working area’ya atar. |
| **git clone URL** | GITHUB, GIT’e klonlanir. Tum repo iner. (initilize + pull) |
| **git push** | GIT > GITHUB dosya gonderilir. Kendi commitli dosyani gonderirsin. **git clone ile dosyalari indirsek de push ile guncellemeleri aliriz.** |
| **git pull** | GUTHUB > GIT dosya cekilir. Daha once clone yapilmissa bu komut degisimleri indirir. Son versiyon indirilir. |
| **git add filename** | Klonlanarak alinan dosyalar ve kendi dosyalarimiz da yapilan degisimleri ayrica yeni dosyalari Stage seviyesine cikarir. |
| **git commit -m «gelistirdim»** | Repo seviyesine kaydettim. |
| **git push** | Tum GIT’teki degisiklikler GITHUB’a kaydedilir.  GITHUB’ta ilgili repo acilirsa commit mesaji ile gonderilen dosya gorulur. |

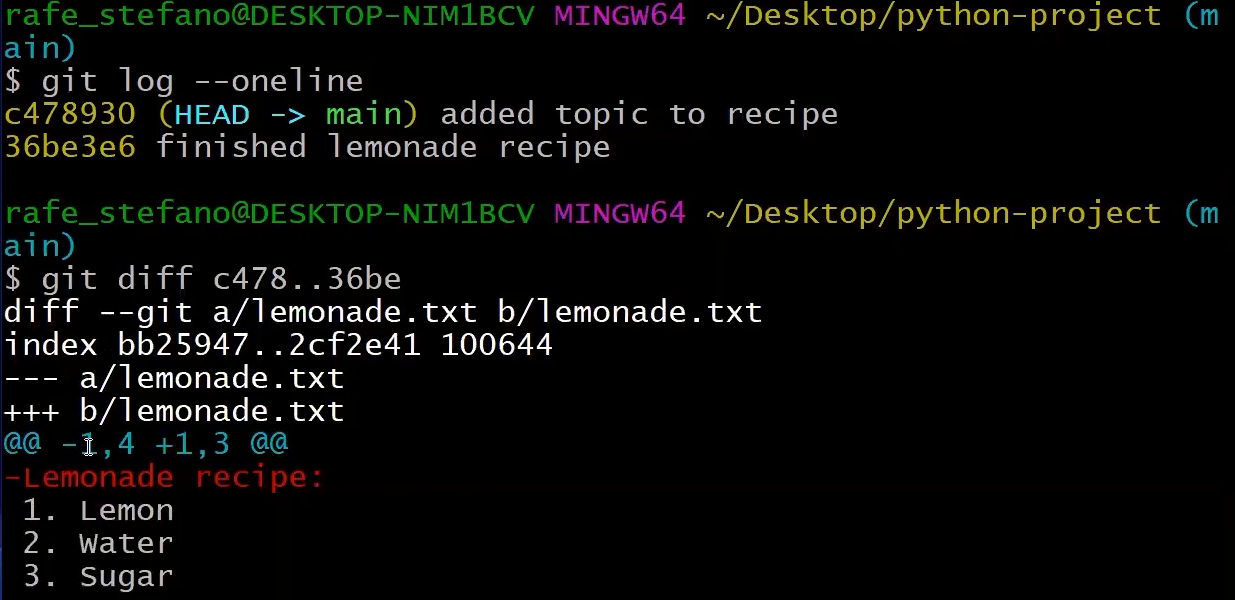
Dosya stage area ve repoda iken git degisimleri tracklar ve kaydeder. Working areada takip yapmaz.

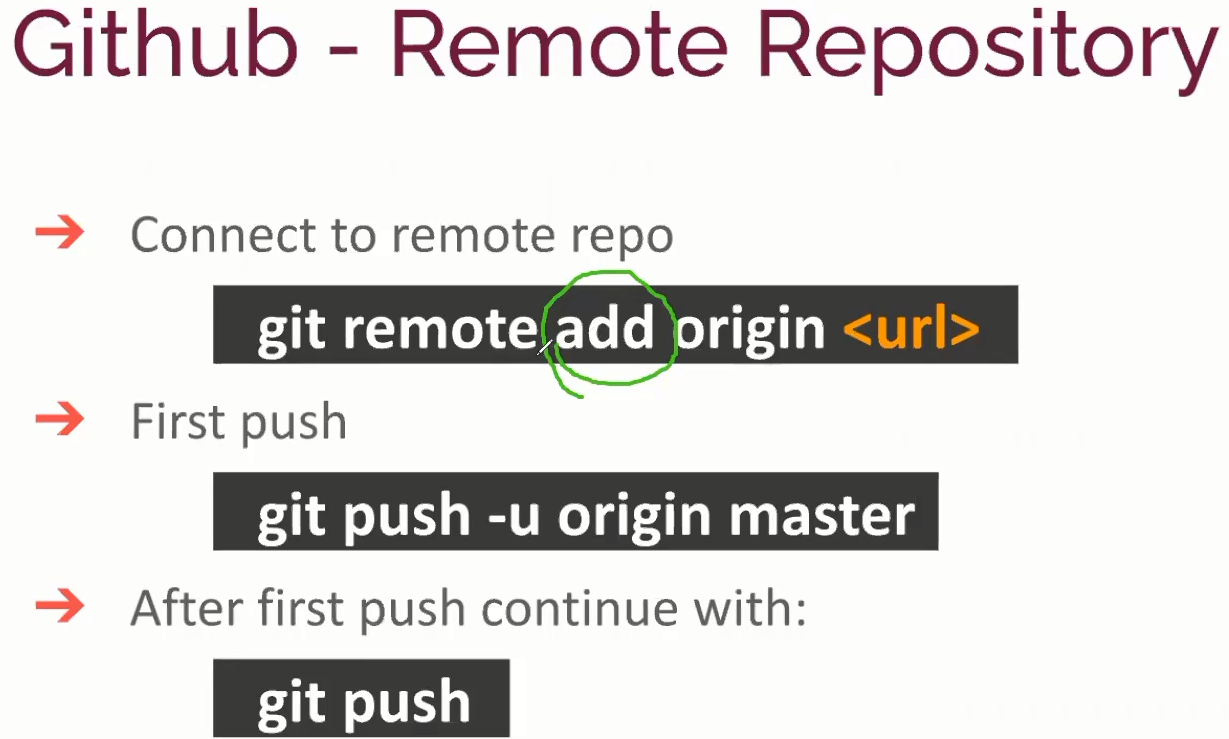
Iki turlu dosya olusturulabilir. Birincisi dogrudan localde yeni bir dosya olusturup devam etmek, ikincisi GITHUB’tan klonlayarak dosya indirmek. Her ikisinde de calismayi tamamlayinca GITHUB’a push edilir.

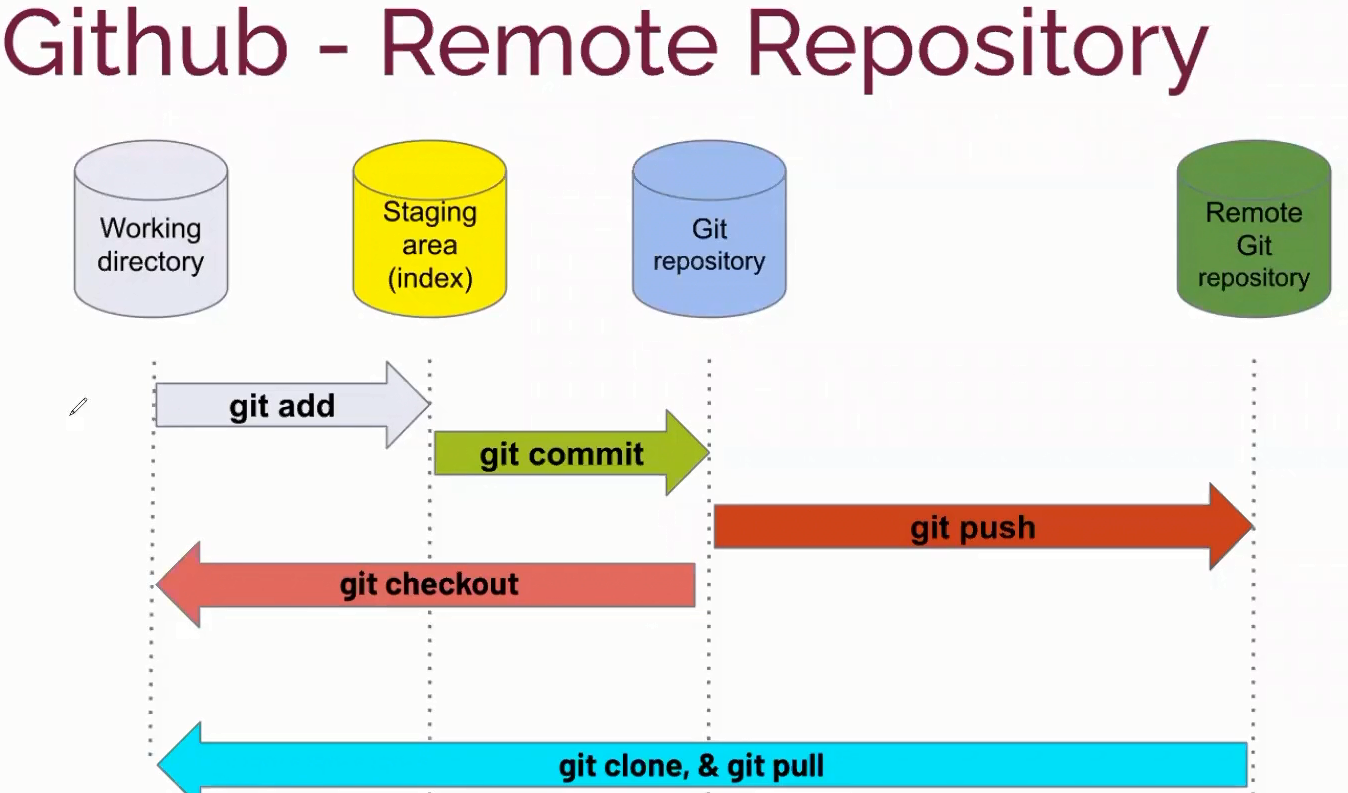
Get status ile «origin» gozukenler remote repository (GITHUB)’dan geldigini gosterir.





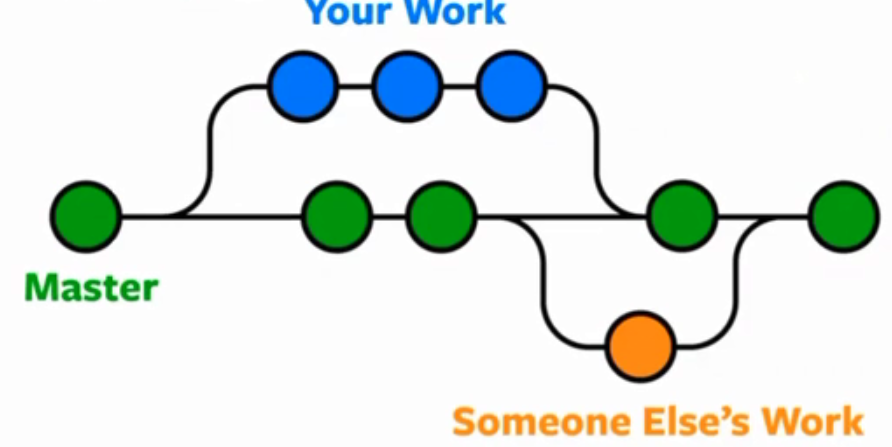






**MAIN/MASTER ve BRANCHES**





Herkes main uzerinde branch olarak calisir. Daha sonra bitince main ile branch/duzeltmelerini **merge** eder.

|  |  |
| --- | --- |
| **git branch** | See local branches. Yildiz calisilandir. |
| **git branch -r** | See remote branches. |
| git branch -a | See all branches. |
| **git branch** branch name | Create a new branch. |
| **git checkout** branch name | Switch to a branch. |
| git checkout -b branch name | Create and switch to a branch |
| **git branch -d** branch name | Delete a local branch. Mevcut branch silinmez. |
|  |  |

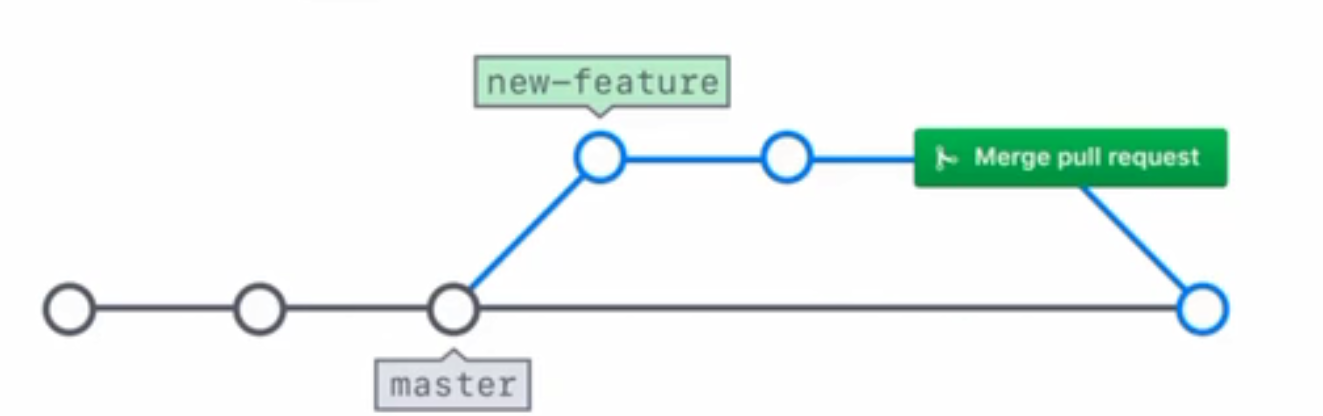
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

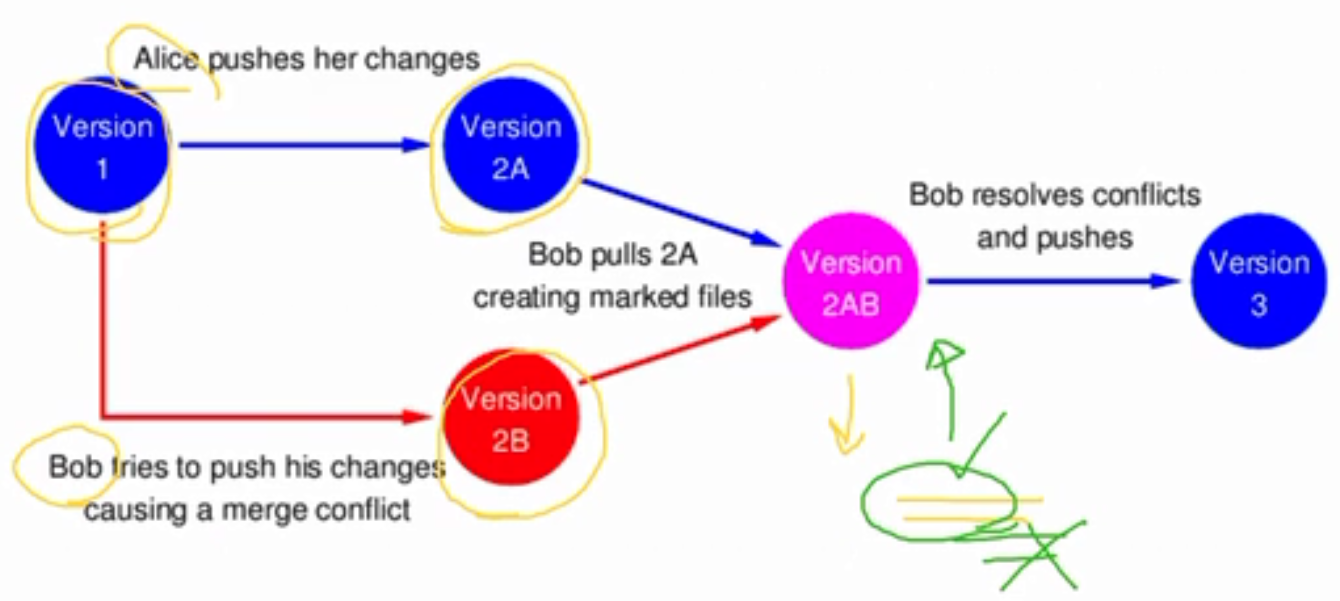
Localde mevcut branchi \* ile gosterirken remote’da /HEAD ile gosterir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu





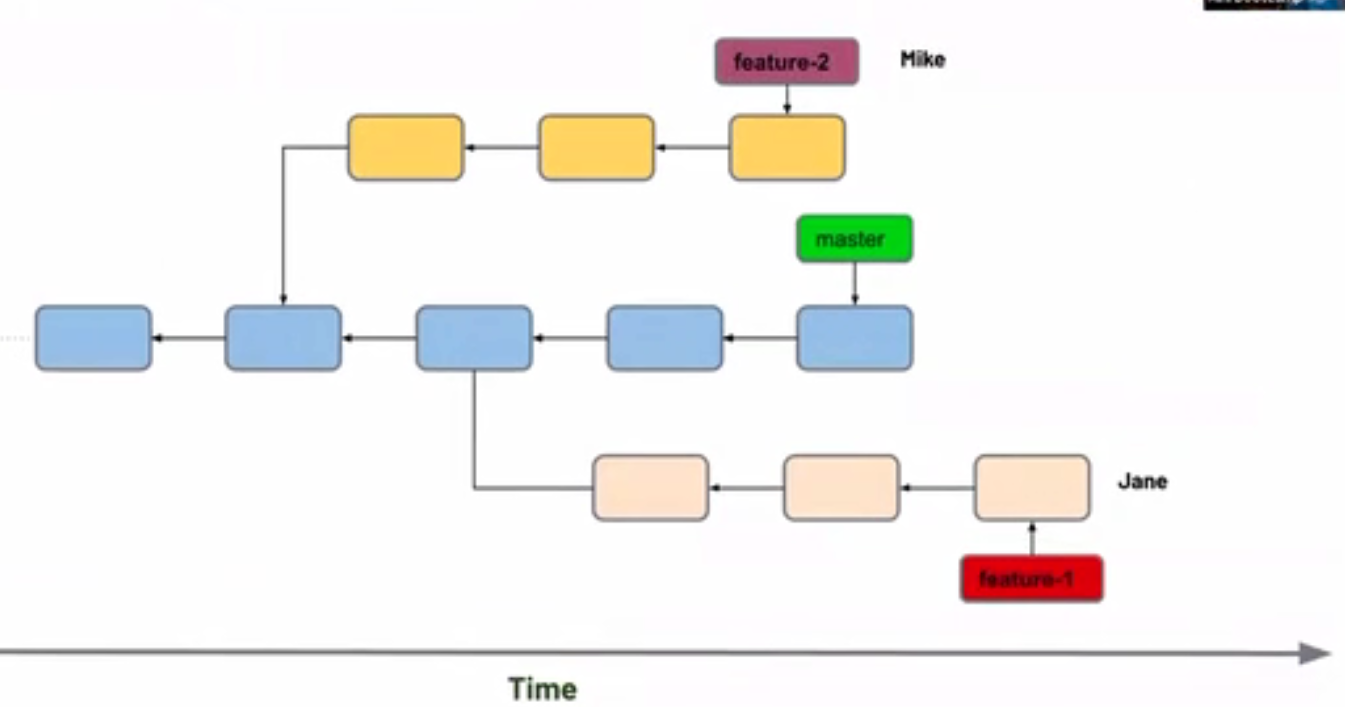


Merge islemi esnasinda farkli alanlardaki degisimler birlesir. Ancak ayni alanda iki farkli branchta da degisiklik varsa buradaki degisikligi siz karar vermelisiniz.

---------------------------------------------------------------------------------------------------------------

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

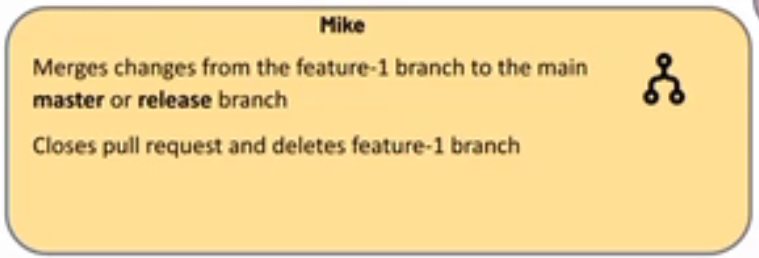
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



New feature implemented.

**FORK:**

Clonlamada remote (kendi) > local (kendi)

Forklamada remote (yabanci )> remote (kendi)

|  |  |
| --- | --- |
|  |  |

