

GITHUB

staging area bizim görebileceğimiz bir şey değil. burada somut olarak algılayabileceğimiz tek sistem working space. staging area ve commit store soyut alanlar. staging area geçici olarak versiyonların depolandığı bir yer. çalışma alanımızda (working space) bir dosya üretiyorum. orada oluşturduğum yapıyı öncelikle staging area ya gönderiyorum geçici bir şekilde orada tutuluyor. karar verip de bunun bir versiyonunu oluşturacağım dersem o zaman bu versiyon staging areadan commit store a gönderilir. commit store lokalimizde versiyonların tutulduğu yerdir.

working space --> staging area

staging area --> commit store

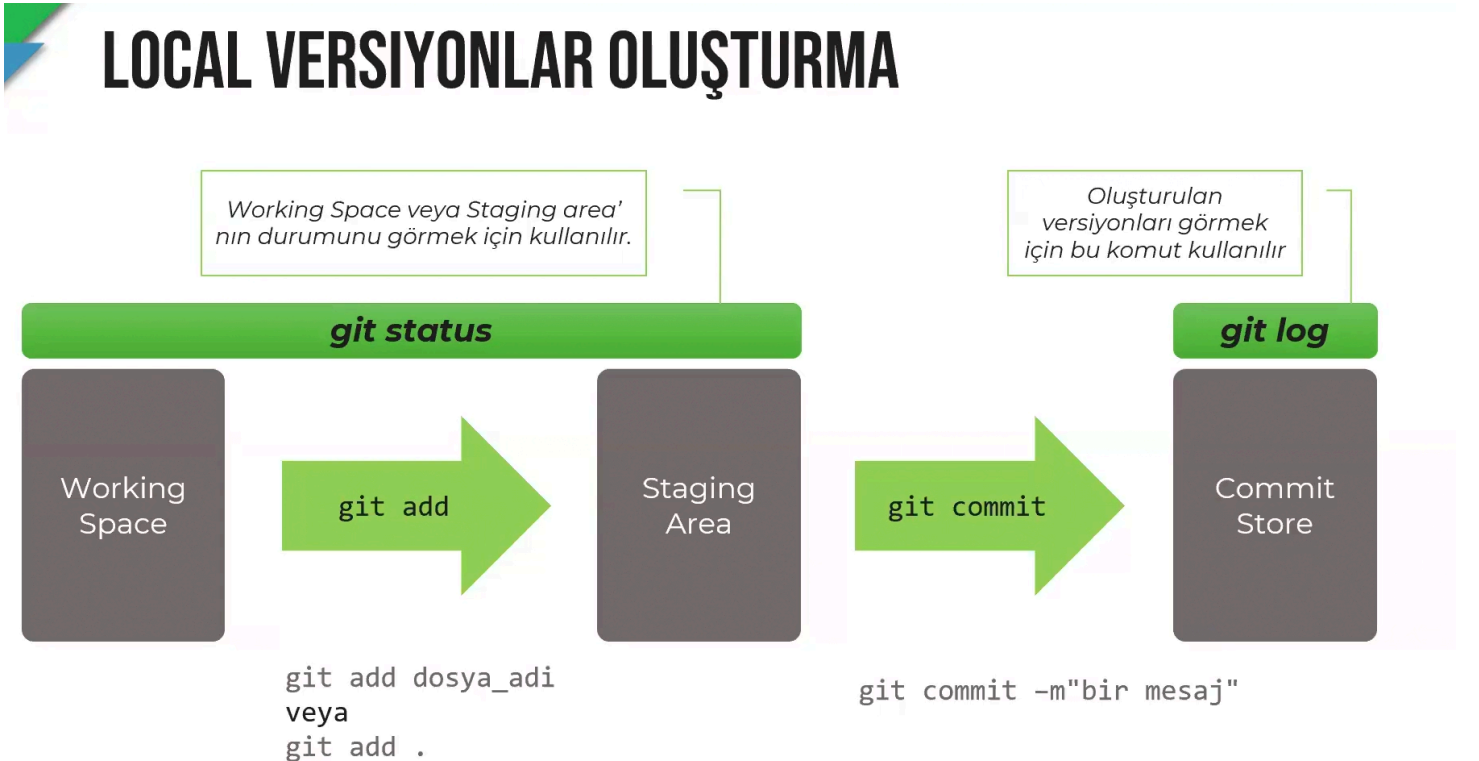
git init - bunu bir dosya için bir kez kullanırız, başlangıçta kullanırız. yeni bir git repositorysi oluşturmak için kullanılır. her yeni dosya oluşturulduğunda git init yazmaya gerek yok

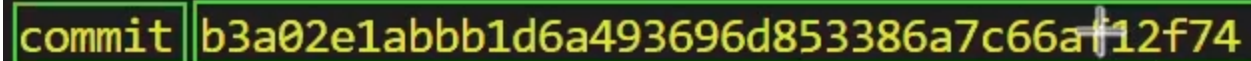
git status --> working space veya staging area'nin durumunu görmek için kullanılır

git add . , **git add dosya_adi** --> dosya ekler

git commit -m"mesaj" --> versiyonları ekleriz

git log --> oluşturulan versiyonları bu şekilde görürüz



A terminal window with a dark background and green text. The text shows a git commit command followed by a long hexadecimal hash: `commit b3a02e1abbb1d6a493696d853386a7c66a112f74`.

```
commit b3a02e1abbb1d6a493696d853386a7c66a112f74
```

hash code unique'tir

git log --oneline --> ilk 7 karakteri verir

git show 0000000 --> versiyonda hangi degisikliklerin yapildigini gosterir

git restore dosya_adi , **git restore .** --> working space'te olan hali hazirda staging area'ya gondermedigimiz datalari silebiliriz. en son kaydedilen versiyona doner

git restore --staged dosya_adi , **git restore --staged .** --> staging area'daki degisiklikleri working space'e geri gonderir

git reset --hard --> working space'teki degisiklikleri iptal eder, staging area'yi bosaltir

git checkout 0000000 .** , **git checkout .** --> hash codu'u yazilan versiyona gider

U --> untracked, bu dosyayi takip etmiyorsun (dosya adi yaninda cikar)
bir onceki versiyonda olusturulmus olmayan

M --> modifiye edildi
hali hazirda olan ve uzerinde degisiklik yapilmis olan

VERSİYON OLUŞTURMAK İÇİN KODLAR

Ana komutlar

```
git init  
git add .  
git commit -m "versiyon metni"
```

Repo oluşturur. Her projede en başta bir kere kullanılır.

Dosyaları staging area ya gönderir

Versiyon oluşturur

Yardımcı komutlar

```
git status  
git log  
git show [hash_kodu]
```

Genel durum ile ilgili bilgi verir

Versiyonların listesini verir

Versiyondaki değişiklikleri gösterir

head en son alınan commit'e head denir

DEĞİŞİKLİKLERİ IPTAL ETMEK

Working space

```
git restore [dosya]
```

Tek bir dosyayı iptal eder

```
git restore .
```

Tüm dosyaları iptal eder

```
git reset --hard
```

Working space deki değişiklikleri iptal eder, staging area yı boşaltır.

Stage Area

```
git restore --staged [dosya]
```

Tek bir dosyayı iptal eder

```
git restore --staged .
```

Tüm dosyaları iptal eder

Commit Store

```
git checkout [hash] [dosya]
```

Dosya,hash ile belirtilen versiyona döner

```
git checkout [hash] .
```

Hash değeri verilen versiyona döner



BRANCH KOMUTLARI

`git branch [isim]`

Yeni branch oluşturur

`git checkout [isim]`

Branch aktif hale gelir

`git branch -m [isim]`

Branch ismini değiştirir.

`git branch`

Mevcut branch leri listeler

`git merge [isim]`

İki branch i birleştirir

`git branch -d [isim]`

Branch i siler

git branch ' te asteriks olan branch kullaniliyor demektir. ilk repository olusturuldugu zaman bize bir tane default branch verir. onun adi da master'dir. bu yeni sistemde main diye geciyor. yeni repo oluturdugumuz zaman giyhub uzzerinden master branch'i main'e donusturmemizi ister.

git branch isim --> bu sekilde yeni isim olusturulur

git checkout isim --> branch degistirilir

git checkout master(main) --> master branch'ine doner

git merge master(main) --> masterdaki ya da userdaki calismalari birlestirir

git clone --> git ve github'i baglamanin iki yontmi var biri git init digeri git clone. ya sistemi lokalde acip github'a baglariz ya da repo'yu github'ta acip lokale baglariz.

bir yerde hali hazirda olusturulmus repo'yu almak icin bu komutu kullaniriz

kendi lokalimizde git init kullanarak calistiracagimiz kodlar asagida sirasi iledir

```
echo "# b304_5_6_practice" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/JaneBlue-Algorithm/b304_5_6_practice.git
git push -u origin main
```

git clone --> *link*

repo'yu lokale klonlar

```
git add .  
git commit -m 'v2'  
  
git push
```

```
git branch  
  
git branch ozenc  
git branch  
  
git checkout ozenc  
  
git branch
```

pushlamak sadece ayni branch'ten ayni branch'e yapilir:

git push --set-upstream origin ozenc

sonrasinda;

Compare & pull request

Create pull request





This branch has no conflicts with the base branch
Merging can be performed automatically.

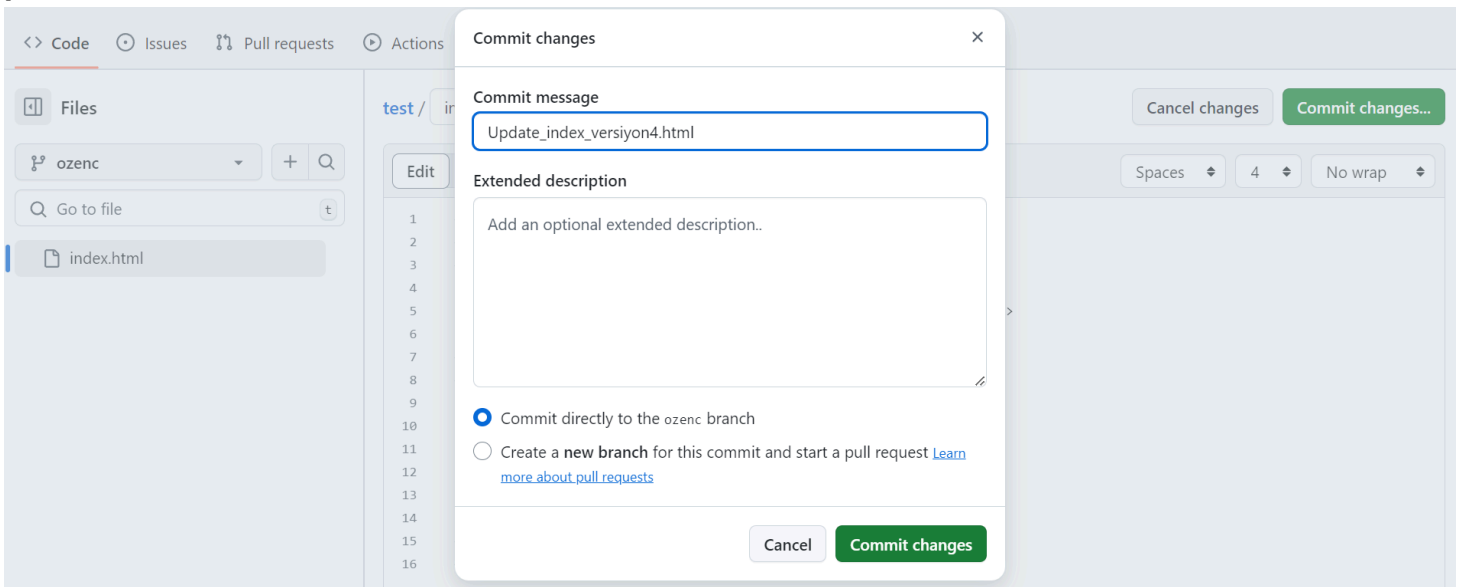
Merge pull request



You can also [open this in GitHub Desktop](#)

Confirm merge

pull'lamak



ozenc

2 Branches **0 Tags**

This branch is **1 commit ahead of, 1 commit behind** **main**.

```
git branch
```

```
git pull
```

TERMINAL KISAYOLLARI

clear --> terminali temizler

cls --> terminal sayfasini temizler

pwd --> su anki konumu gosterir

cd dosya_adi --> klasoru acar

cd .. --> bir onceki klasore gider