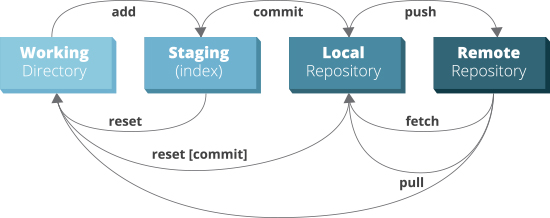
GIT Bash ile GIT Temel Komutları

GIT Temel Komutları



**Başlıca bilmemiz gereken bazı terimler;**

* **untracked (izlenmeyen):** GIT tarafından henüz takip edilmeyen, yani yeni oluşturulmuş dosyaları ifade eder.
* **unstaged (hazırlanmamış):** Güncellenmiş ancak *commit*’lenmek için hazırlanmamış dosyaları ifade eder.
* **staged (hazırlanmış):** *Commit*’lenmeye hazır olan dosyaları ifade eder.
* **deleted (silinmiş):** Projeden silinmiş ama GIT üzerinden kaldırılmamış dosyaları ifade eder.

### **git config**

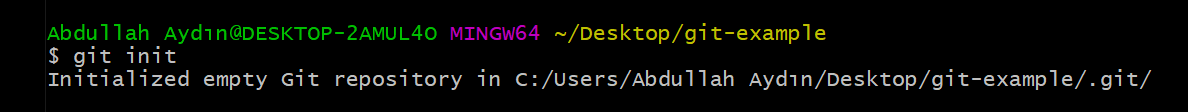
GIT’in bir çok konfigürasyon ve ayarı vardır, bunlardan ikisi user.name ve user.email olanıdır. Bu ayarları yapılandırmak için aşağıdaki komutları kullanırız. GIT'i ilk kurduğumuzda genellikle aldığımız ilk hata bu configurasyon ayarlarını yapmadığımız için gelir. Burada yazdığınız isim ve email ileride GitHub benzeri bir plat forma commit attığınızda da görüneceği için bunu bilerek isimlendirme yapmak yararlı olur. Ayrıca görüldüğü gibi bu ayarlar --globalyani sistem genelinde geçerli ayarlardır. Proje bazlı bu ayarları değiştirebiliriz.

* $ git config --global user.name "Name Surname"
* $ git config --global user.email "test@email.com"
* **Bu ayarların bütününü görüntülemek için:**
* $ git config --list

### **git init**

Henüz versiyon kontrolü altında olmayan bir projenin dizininde, boş bir git deposu oluşturmak için kullanılır.

* $ git init



### **git add**

Yeni eklenen veya üzerinde değişiklik yapılan dosyaları **staged** ortamına göndermek için kullanılır.

$ git add <dosya veya klasor\_name>

**Tek seferde bütün dosyaları eklemek için ise:**

$ git add . veya $ git add \* veya $ git add -A .

Buradaki -A (all) tümü anlamındadır. . ise tüm dosya uzantılarını ifade eder.

$ git add -u

Yalnızca aşamada(stage area) değiştirilmiş ve silinmiş dosyalar

git commit

*Commit*, **staged** ortamına alınan dosyaların *Local Repository*’e gönderilmesidir. En iyi uygulama yöntemi her kayıt sırasında yapılan değişiklikleri açıklayıcı bir mesaj eklemektir. Ayrıca her commit benzersiz bir kimliğe (unique ID) sahip olur. Bu sayede eski bir commit'e geri dönebilirsiniz ve herhangi bir kayıp yaşama ihtimaliniz kalmaz.

$ git commit -m "ilk commit mesajı"

Buradaki **-m** (message) mesaj anlamındadır.

$ git commit –amend Taahhüt mesajını değiştir

$ git commit -am ”message”

### Tüm izlenen dosyaları eklemek ve onaylaymak için

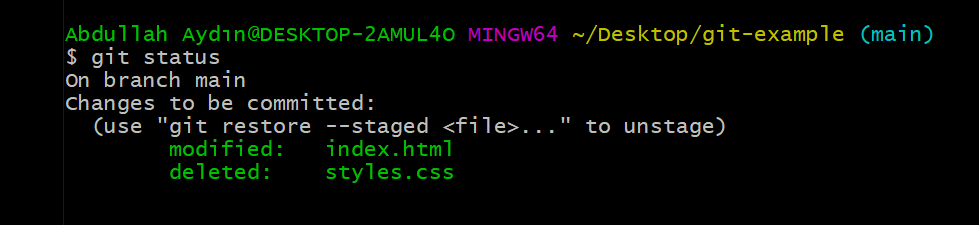
### **git push**

* Projemizde aldığımız commit'leri, remote repository'e gönderir.
* $ git push origin main
* ***Burada bahsi geçen origin remote repository’nin kök dizinini belirtir ve sabit bir isimdir. main ise sizin çalıştığınız branch (dal)’ı belirtir.***
* Henüz remote repository’niz yoksa aşağıdaki komut ile local deponuzu uzak sunucudaki depoya bağlayabilirsiniz.
* $ git remote add origin http://uzak\_deponun\_adresi.git

### **git status**

Üzerinde çalışılan projenin o anki durumu hakkında bilgi verir. Yapılan değişiklikler, eklenen ve silinen dosyalar gibi bilgiler listelenir.

$ git status

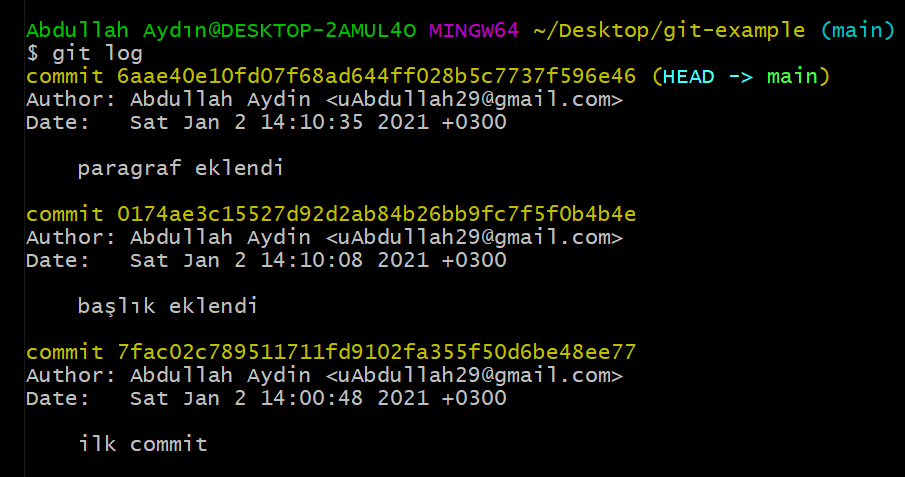


* On branch main -> Main branch'ınde olduğumuzu,
* Changes to be commited -> Commit'lenmeye hazır değişiklikler olduğunu,
* Modified: index.html -> Index.html dosyasında **değişiklik** yaptığımızı,
* Deleted: styles.css -> styles.css dosyasını **sildiğimizi**,

### **git log**

Projedeki commit geçmişini görüntülememizi sağlar. Bütün commit'ler, id'si, yazarı, tarihi ve mesajı ile beraber listelenir.

$ git log



**Git log --oneline**

### **Not: git log ekranından q harfine basmalısınız.**

### **git checkout**

* Branch’ler arası veya commit'ler arası geçiş yapmak istediğimizde kullanılır.
* **Mevcutta var olan branch'a geçiş yapmak için;**
* $ git checkout <branch\_name>
* **Yeni bir branch oluşturup, bu branch'a geçiş yapmak için;**
* $ git checkout -b <branch\_name>
* **Commitler arası geçiş yapmak için:** (Eski bir versiyona dönmek istediğimiz zaman)
* $ git checkout <commit\_ID>

### **git branch**

* Local veya remote repository üzerinde yeni bir branch (dal) eklemek, silmek veya listelemek için kullanılır.
* git Branch (yerel şubeleri göster)
* git branch -r (uzak dalları göster)
* git branch -a (tüm yerel ve uzak dalları gösterir)
* **Projenize yeni bir branch eklemek için;**
* $ git branch <branch\_name>
* \*Tüm uzak ve yerel branch'lari listelemek için;
* $ git branch -a
* **Bir branch'ı silmek için;**
* $ git branch -d <branch\_name>

git merge

Başka bir branch'da olan değişiklikleri, bulunduğumuz branch ile birleştirmek istediğimizde kullanılır.

$ git merge <branch\_name>

git clone

Mevcut bir Remote Repository'de bulunan dosyaların bilgisayarımızda bir kopyasının oluşturulmasını sağlar.

$ git clone <remote\_URL>

**Git pull**

$ git pull origin master

*remote repository*’deki güncel datayı *local repository*’mize çeker.

**git diff**

Repository üzerinde yapılan değişikliklerden sonra dosyalar arasında oluşan farklılıkları gösterir. **Çalışma dizini ile repository (HEAD) arasındaki farklılıkları görmek için:**

$ git diff HEAD

**İki commit arasındaki farklılıkları görmek için:**

$ git diff <commit\_id\_1>..<commit\_id\_2>

**Çalışma dizini ve staged ortamı arasındaki farkları görmek için:**

$ git diff --staged

Yerel depodaki değişiklikleri görmek için

git diff main origin/main

**Git rm**

$ git rm file.txt dosyayı **deleted** statüsüne taşır ve local’den siler.

$ git rm -r dizin-ismi dizini **deleted** statüsüne taşır ve local’den siler.

$ git rm --cached file3.txt unstage yapmak için

**Git fetch**

**git fetch** komutu, uzak bir depodan commit'leri, dosyaları, dalları ve etiketleri alır. Komut için genel sözdizimi şöyledir

git fetch <options> <remote name> <branch name>