

DATE : 26.10.2024

DT/NT : DT

LESSON : GIT - GITHUB

SUBJECT: GIT

BATCH : 304 - 305 - 306



techproeducation.com



+1 (585) 304 29 59



TECHPRO
EDUCATION





- Git – Github nedir?
- VKS Nedir?
- Ne amaçla kullanılır



Fundamentals

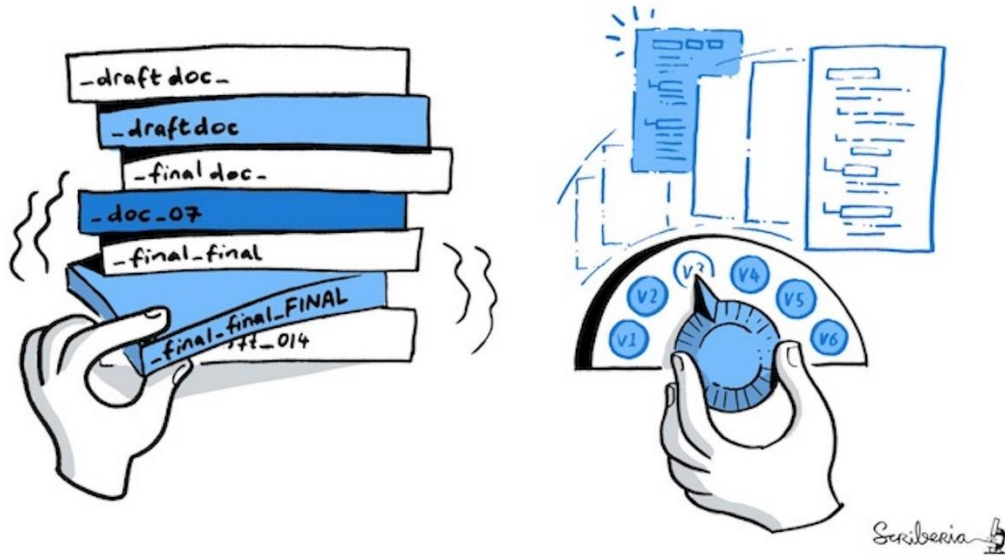
GIT-GITHUB NEDİR

Git-Github versiyon kontrol sistemidir



VKS NEDİR

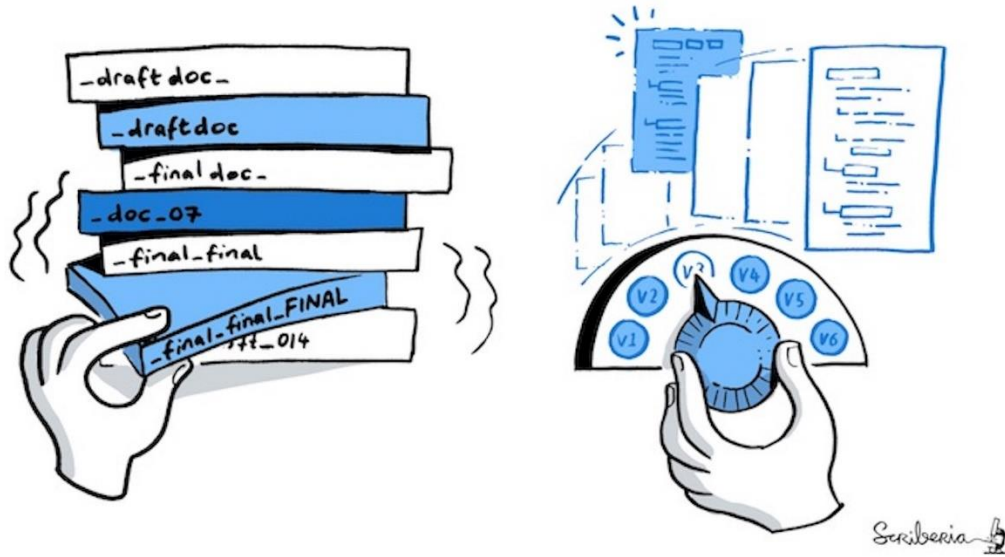
TRACK PROJECT HISTORY



Versiyon kontrol sistemi, projede dosya ve klasör yapısındaki değişiklikleri kaydeden bir sistemdir.

VKS NEDİR

TRACK PROJECT HISTORY



- Bazı dosyaların veya projenin tamamının bir önceki versiyona döndürülmesi,
- Zaman içerisinde yapılan değişikliklerin karşılaştırılması,
- Probleme neden olabilecek değişikliklerin en son kimin tarafından yapıldığının tespiti

VKS ÇEŞİTLERİ

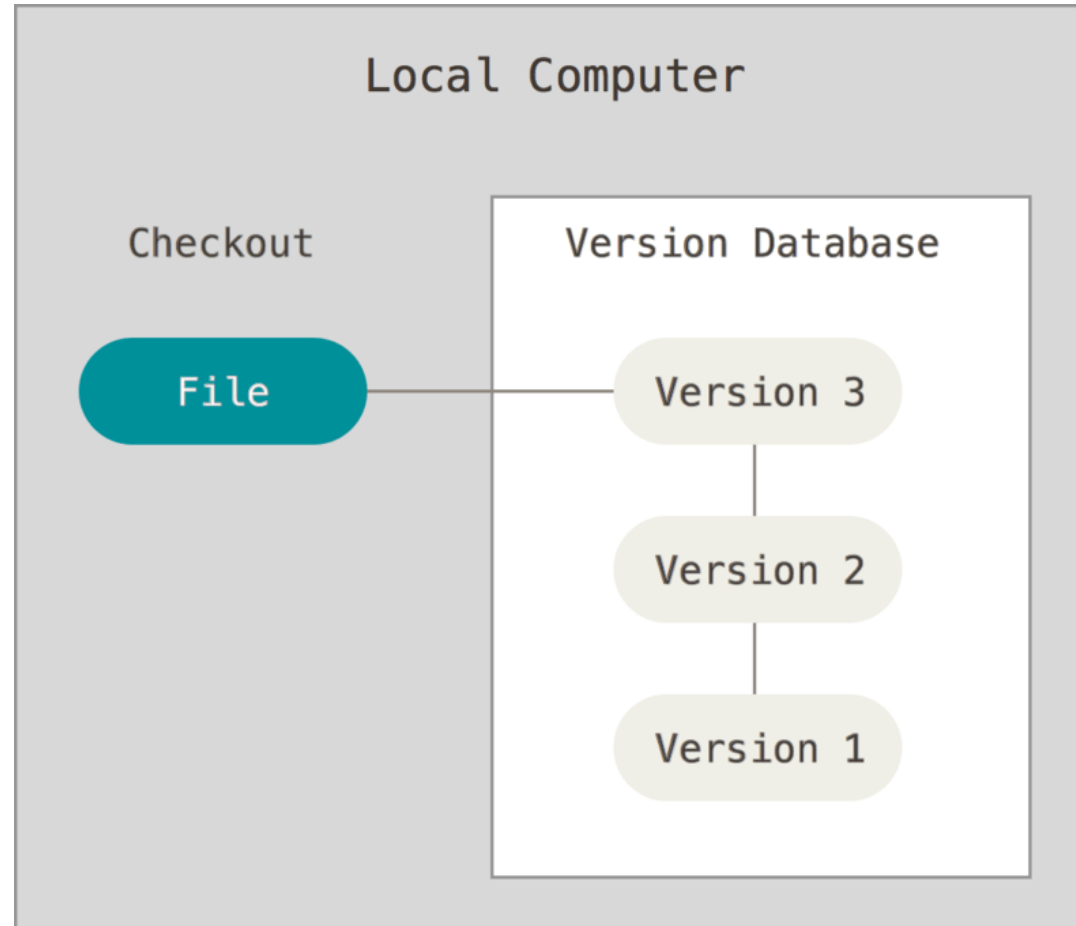
3 tip Versiyon Kontrol Sistemi vardır.

YEREL

MERKEZİ

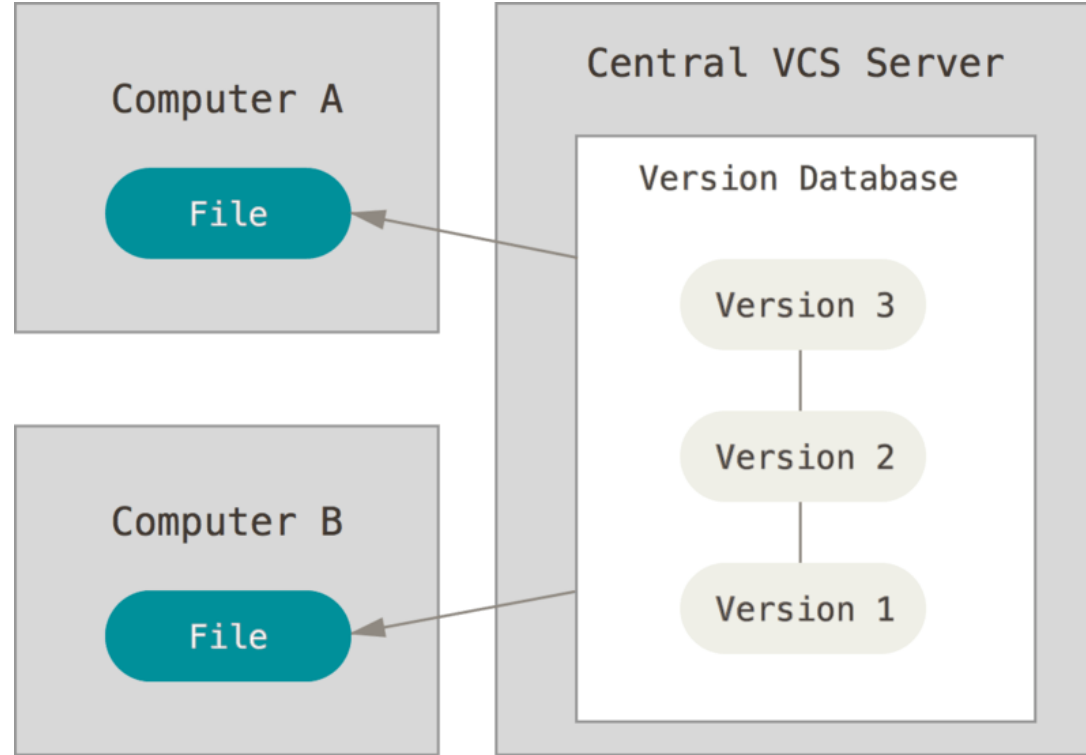
DAĞITIK

YEREL VKS NEDİR



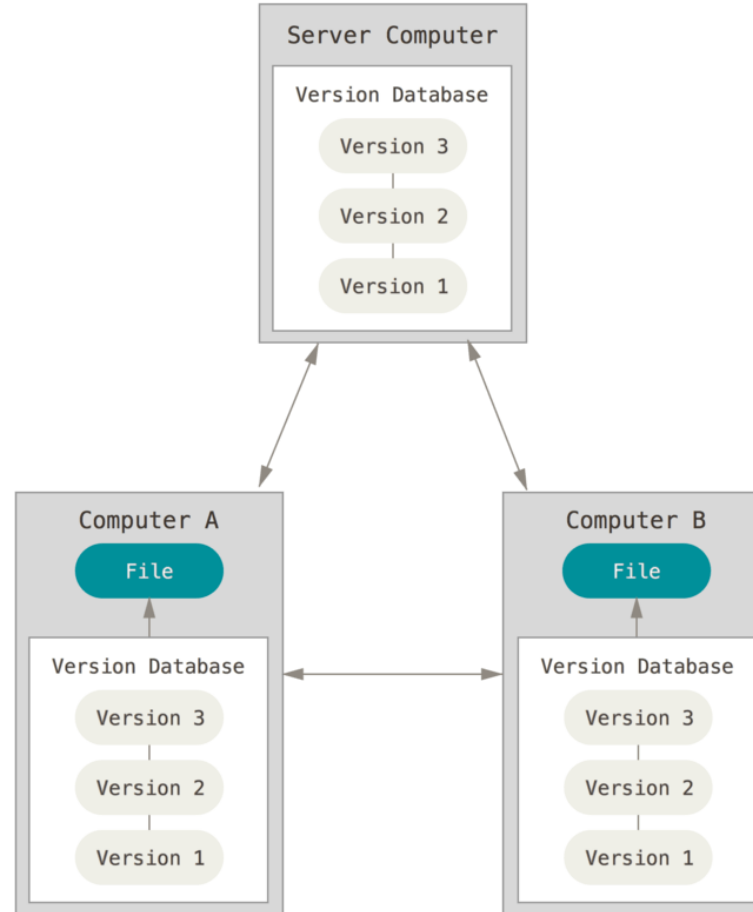
<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

MERKEZI VKS NEDİR



<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

DAĞITIK VKS NEDİR



<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

GIT-GITHUB NE AMAÇLA KULLANILIR



- ⦿ Lokalde versiyon yönetimi yapmak
- ⦿ Offline çalışabilmek
- ⦿ Hataları geri alabilmek
- ⦿ Versiyonlar arasında geçiş yapabilmek



- ⦿ Yedekleme (backup)
- ⦿ Proje paylaşımı (share)
- ⦿ Proje yayınlama (deploy)
- ⦿ Ortak çalışma (collaboration)



- Kurulum ve ilk ayarlar
- Repository
- Lokal repo oluşturma
- Working directory, staging area, commit changes
- Değişiklikleri iptal etme
- Önceki versiyonlara dönme
- Branchs



KURULUM VE İLK AYARLAR



Version Control System

- Git altyapısını oluşturmak ve git komutlarını kullanabilmek için Git kütüphanesinin kurulması gerekmektedir

[<https://git-scm.com/downloads>]

- *git --version*

KURULUM VE İLK AYARLAR

Git configuration

```
git config --global user.name "Ali Gel"  
git config --global user.email "ali@gel.com"
```

Yapılan commit leri burada belirtilen isim ve eposta ile ilişkilendirir. Repo da çalışan diğer kişiler bu isim ve epostayı görür.

```
git config --global color.ui true
```

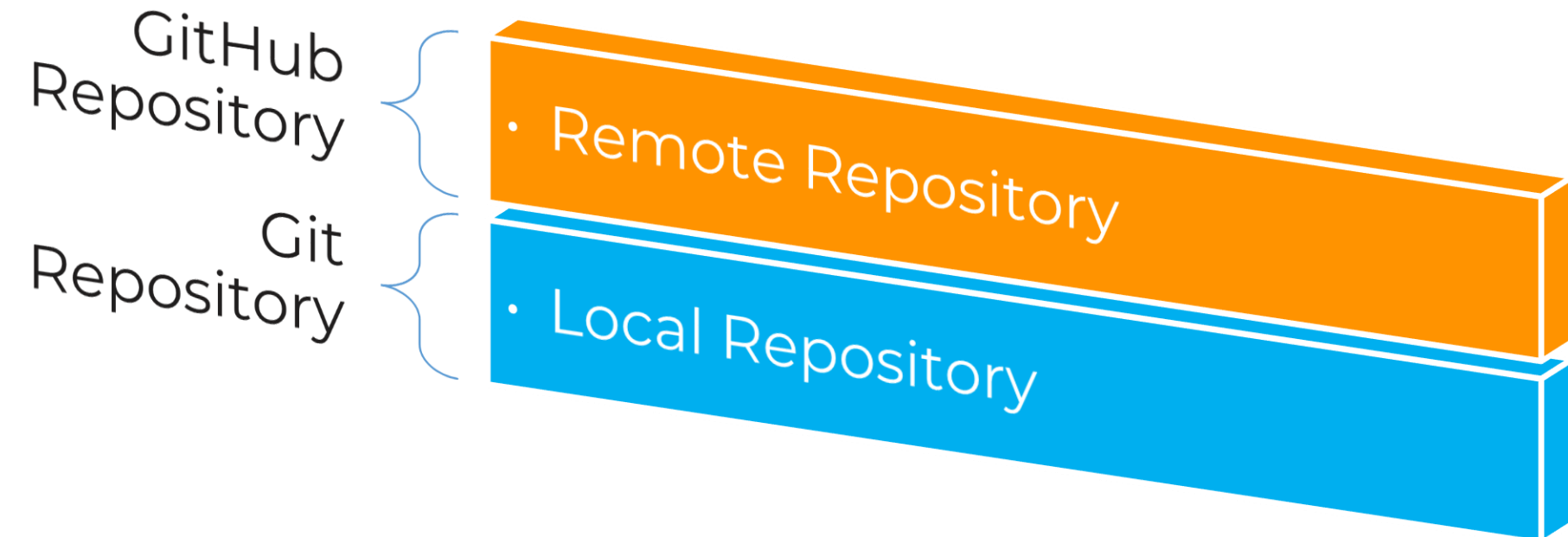
Terminal de komutların renklendirilmesini sağlar

- **System** parametresi kullanıldığında tüm kullanıcılar ve tüm repolar üzerinde etkili olur
- **Global** parametresi geçerli kullanıcının tüm repolar üzerinde etkili olur
- **Local** parametresi ise sadece geçerli repo üzerinde etkili olur

| Repository |

Versiyon kontrol ve birlikte çalışma altyapısını ayrı tutmak istediğimiz her bir bağısız yapıya repository denir. Genellikle her proje için ayrı bir repository tanımlanır.

GENEL KAVRAMLAR



LOCAL REPO OLUŞTURMA

| **git init** |

Local bilgisayarımızda bir projeyi versiyon sistemine alabilmek için git init komutu kullanılır. Bu komut kullanılınca proje klasöründe .git klasörü oluşturulur. Bu, local repomuzu saklayacaktır.

GENEL KAVRAMLAR



Working Space

.git klasörünün bulunduğu çalışma alanıdır. Klasörler ve dosyalar üzerinden değişiklik burada yapılır.



Staging Area

Versiyon oluşturulacak olan dosya veya klasörlerin geçici olarak toplandığı yerdir. Versiyon (commit) oluşturulduktan sonra otomatik olarak staging area boşaltılır



Commit Store

Git her bir commit i ayrı bir versiyon olarak tutar. Böylece yapılan çeşitli değişikliklerden sonra projede sorunlar ortaya çıkarsa bir önceki commit e geri dönülebilir.

LOCAL VERSİYONLAR OLUŞTURMA

Working Space veya Staging area'nın durumunu görmek için kullanılır.

git status

Working Space

git add

Staging Area

```
git add dosya_adi  
veya  
git add .
```

Oluşturulan versiyonları görmek için bu komut kullanılır

git log

Commit Store

git commit

```
git commit -m"bir mesaj"
```

VERSİYON DETAYLARINI GÖRME

git show

Bir versiyon içinde, hangi değişikliklerin olduğunu görmek için kullanır.

git show *[hash kodun ilk 7 karakteri]*

git log

```
C:\Users\sariz\Desktop\test>git log
commit c417dfe1afa5deac505808a0a2c8ba05afc8e86d (HEAD -> master)
Author: Your Name <you@example.com>
Date: Sat Aug 7 23:49:17 2021 +0300

    1 satır eklendi

commit 5e063d211454b3bc8846bc0720aef4895b1fdbff
Author: Your Name <you@example.com>
Date: Sat Aug 7 23:40:18 2021 +0300

    first commit
```

VERSIYON 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

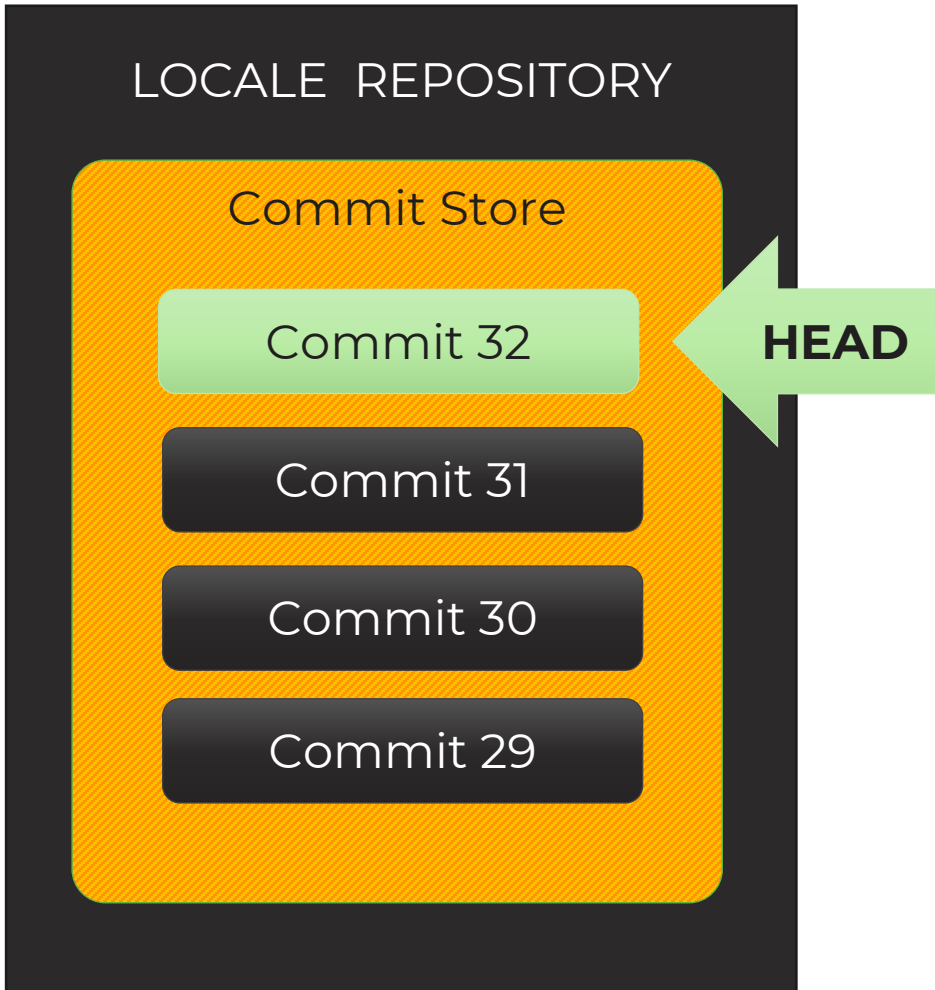
```
git status  
git log  
git show [hash_kodu]
```

Genel durum ile ilgili bilgi verir

Versiyonların listesini verir

Versiyondaki değişiklikleri gösterir

COMMIT STORE & HEAD




- Bir repo içinde birden fazla commit olabilir. Bunlardan en son alınan commit'e **HEAD** denir.
- Bu HEAD değiştirildiğinde önceki versiyonlara dönüş yapılabilir.

```
C:\Users\sariz\Desktop\test>git log
commit c417dfe1afa5deac505808a0a2c8ba05afc8e86d (HEAD -> master)
Author: Your Name <you@example.com>
Date: Sat Aug 7 23:49:17 2021 +0300

    1 satır eklendi

commit 5e063d211454b3bc8846bc0720aef4895b1fdbff
Author: Your Name <you@example.com>
Date: Sat Aug 7 23:40:18 2021 +0300

    first commit
```



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]`

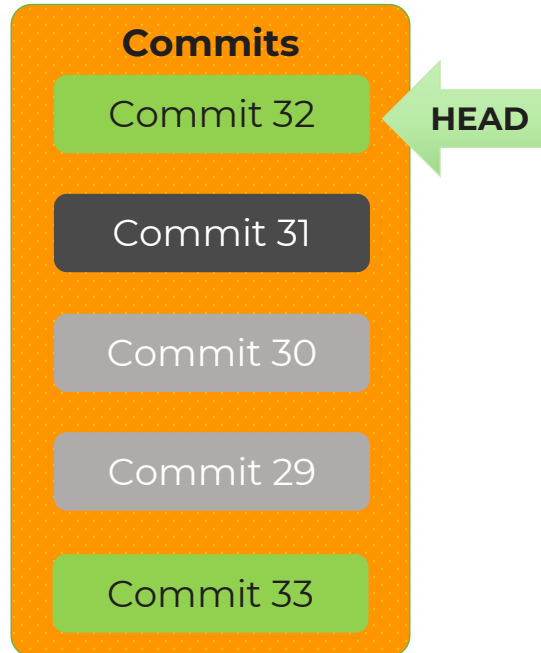
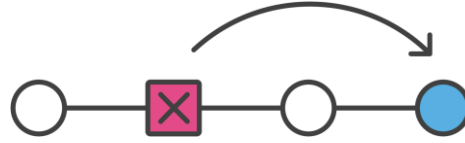
Dosya,hash ile belirtilen versiyona döner

`git checkout [hash] .`

Hash değeri verilen versiyona döner

ÖNCEKİ VERSİYONLARA DÖNMEK

1.Yöntem: CHECKOUT

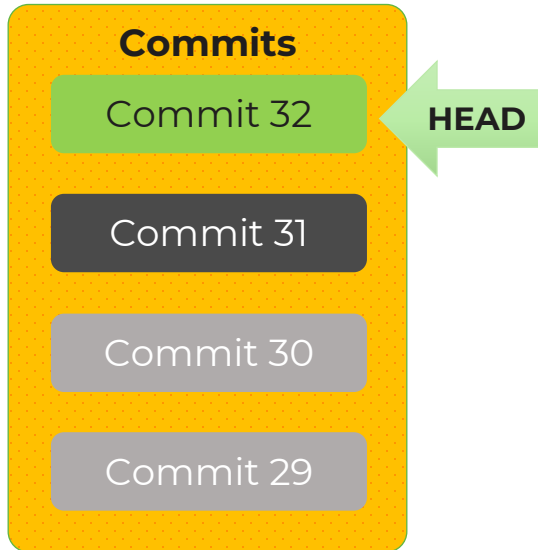
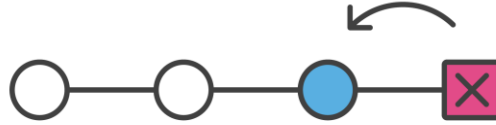


Önceki versiyonu incelemek için
git checkout *[hash]*.

Bu işlemi kalıcı hale getirmek için
git commit -m"..."

ÖNCEKİ VERSİYONLARA DÖNMEK

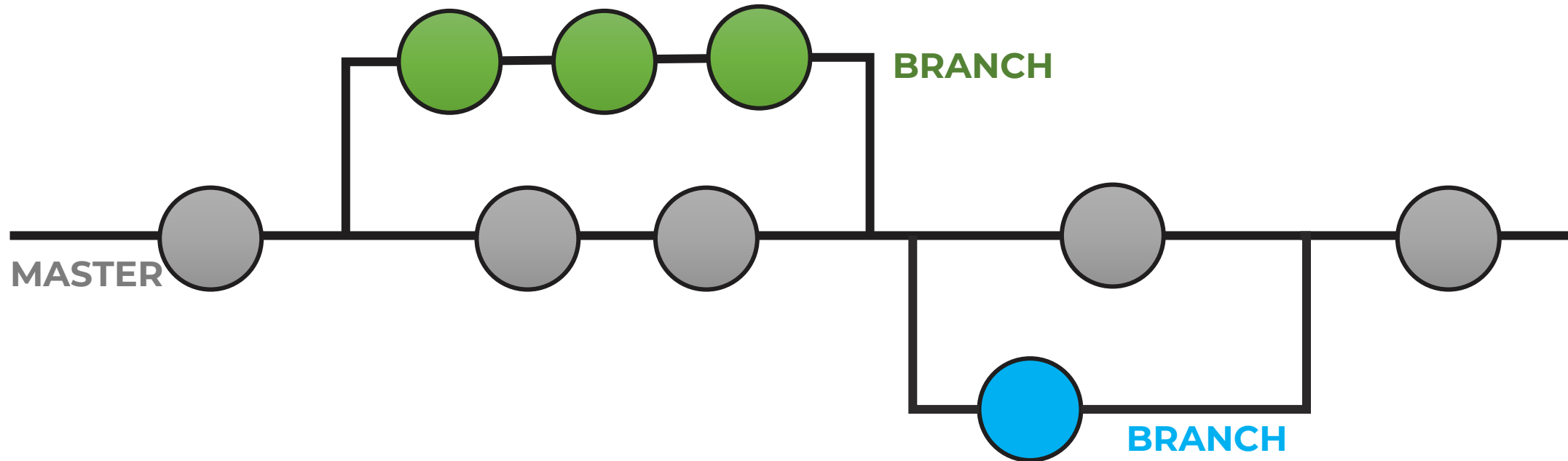
2.Yöntem: RESET



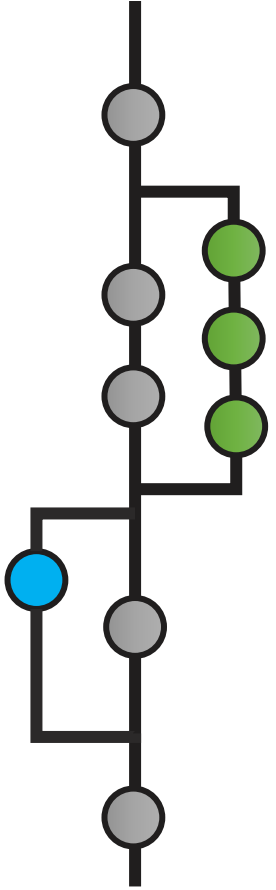
Geri alınamayacak şekilde önceki versiyona dönmek

git reset --hard *[hash]*

BRANCH (DAL)



BRANCH LERIN FAYDALARI



- Original kodların güvenliği sağlanır
- Her developer kendi bölümünden sorumlu olur
- Daha hızlı geliştirme yapılır
- Daha az hata oluşur
- Sorunlar daha hızlı düzeltilir.
- Organize kod yapısı sağlanır
- Kaos olmaz

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

STASHING

Working directory ve stage area daki *–henüz commit haline gelmemiş–* değişikliklerin **geçici olarak geri alınması** için stashing işlemi yapılır.

git stash

Working space ve staging area daki değişiklikleri geçici olarak hafızaya alır ve bu bölgeleri temizler

git stash list

Hafızaya alınan değişiklikleri görmek için kullanılır

git stash pop

Hafızaya alınan değişiklikleri geri uygulamak için kullanılır.





INTERVIEW