DATE : 2.12.2024

DT/NT : TN

LESSON : GIT - GITHUB

SUBJECT: GIT

BATCH : 304 – 305 - 306

TECHPRO EDUCATION

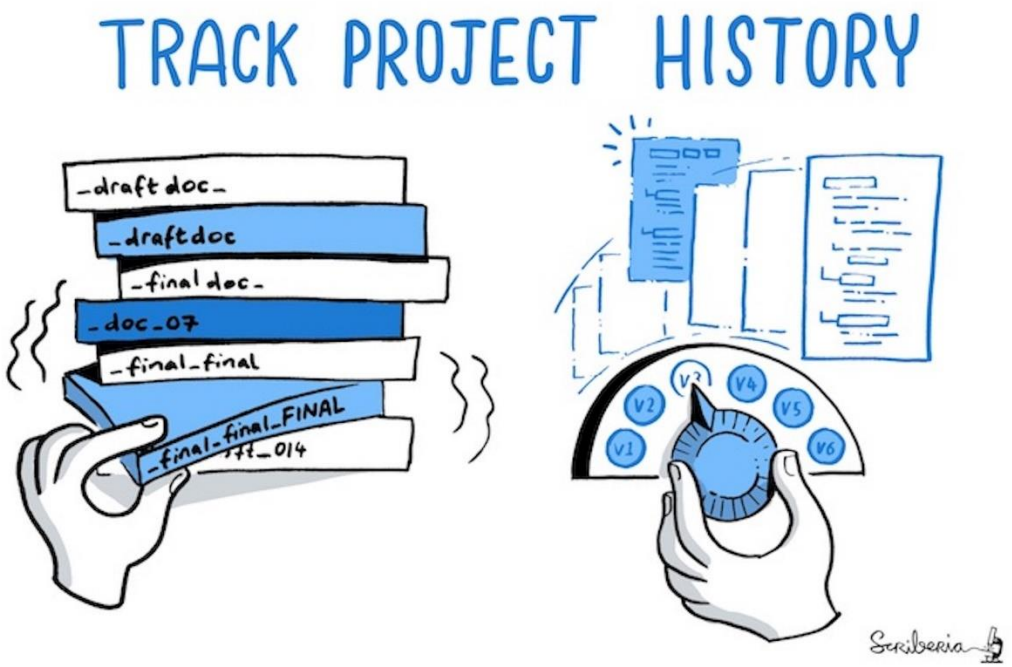
- Git – Github nedir?
- VKS Nedir?
- Ne amaçla kullanılır

Fundamentals

GIT-GITHUB NEDIR

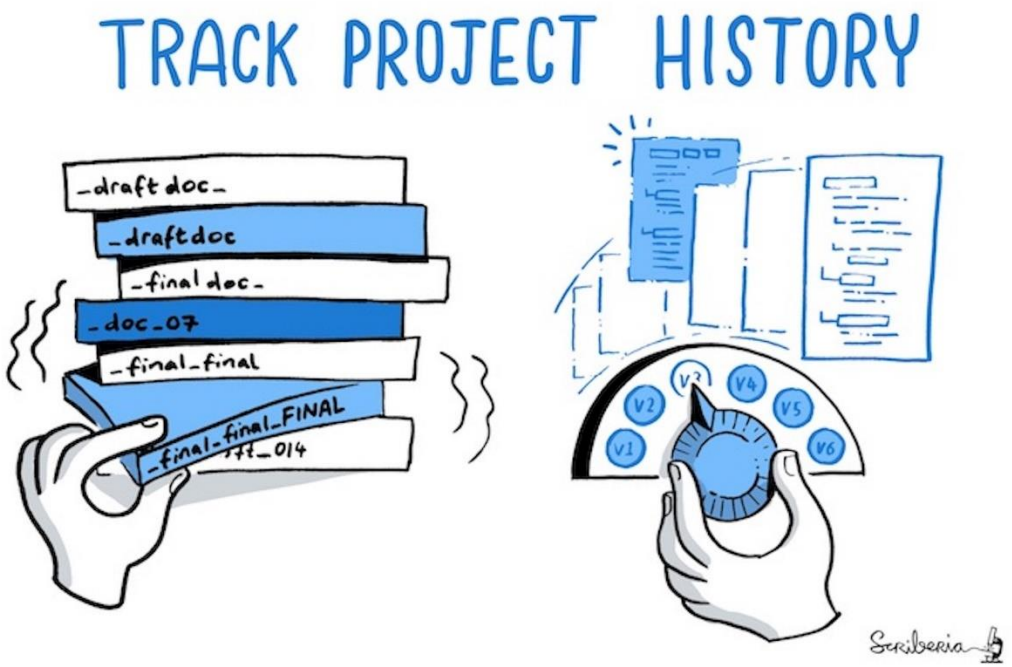
Git-Github versiyon kontrol sistemidir

VKS NEDIR



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

VKS NEDIR



- 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 ÇEŞITLERI

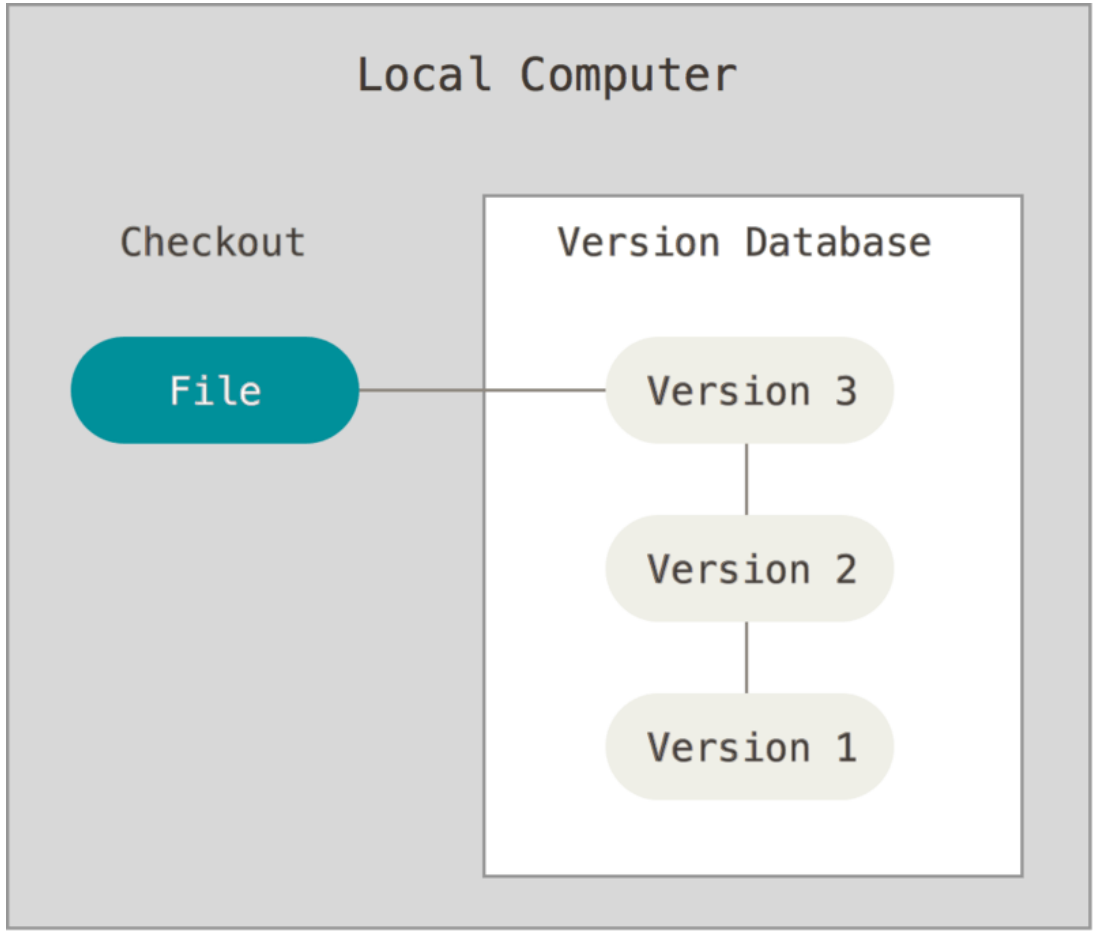
3 tip Versiyon Kontrol Sistemi vardır.

YEREL

MERKEZİ

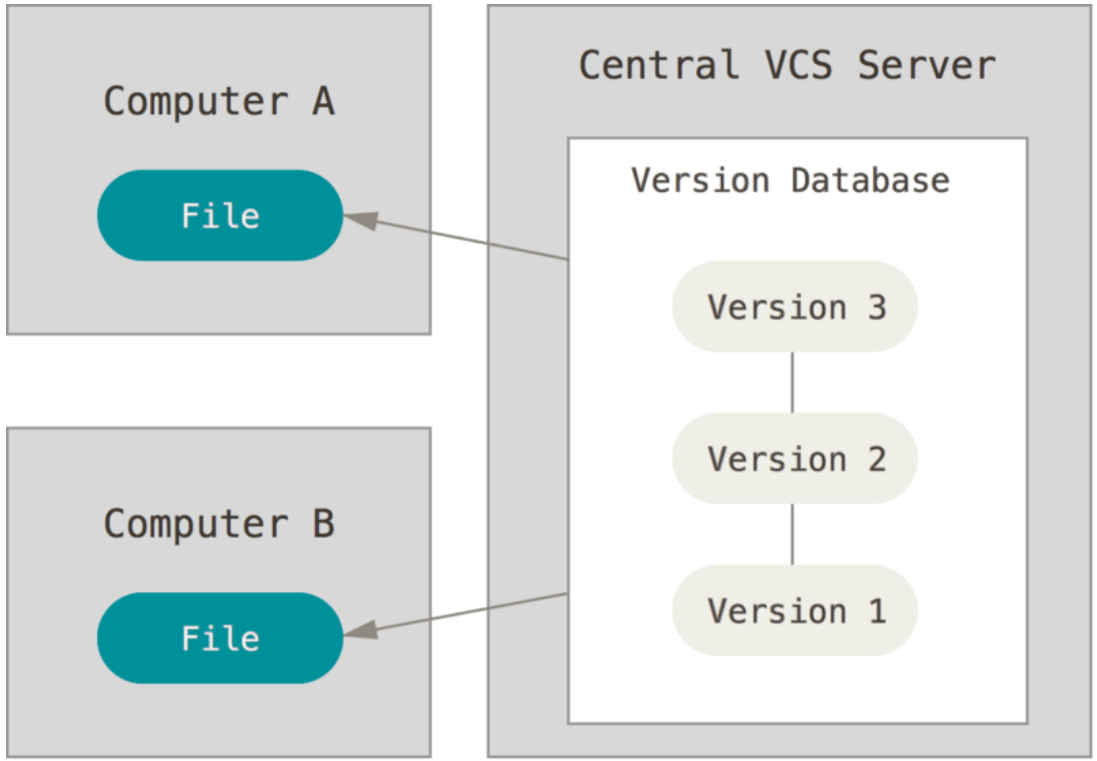
DAĞITIK

YEREL VKS NEDIR



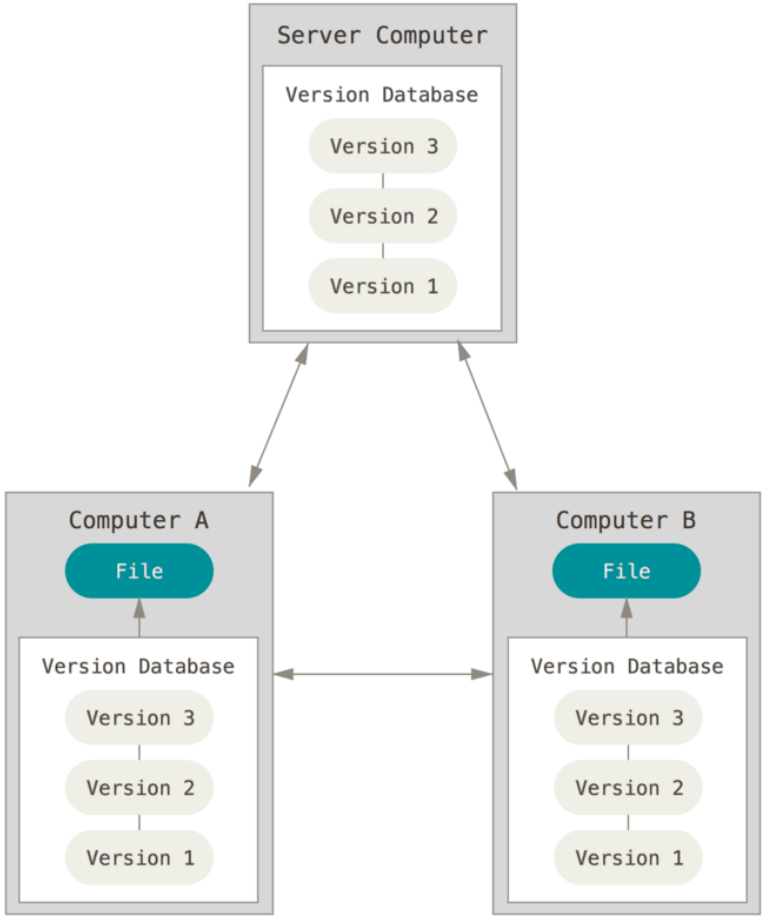
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

MERKEZI VKS NEDIR



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

DAĞITIK VKS NEDIR



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

GIT-GITHUB NE AMAÇLA KULLANILIR

LOCALE

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

REMOTE

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

- 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

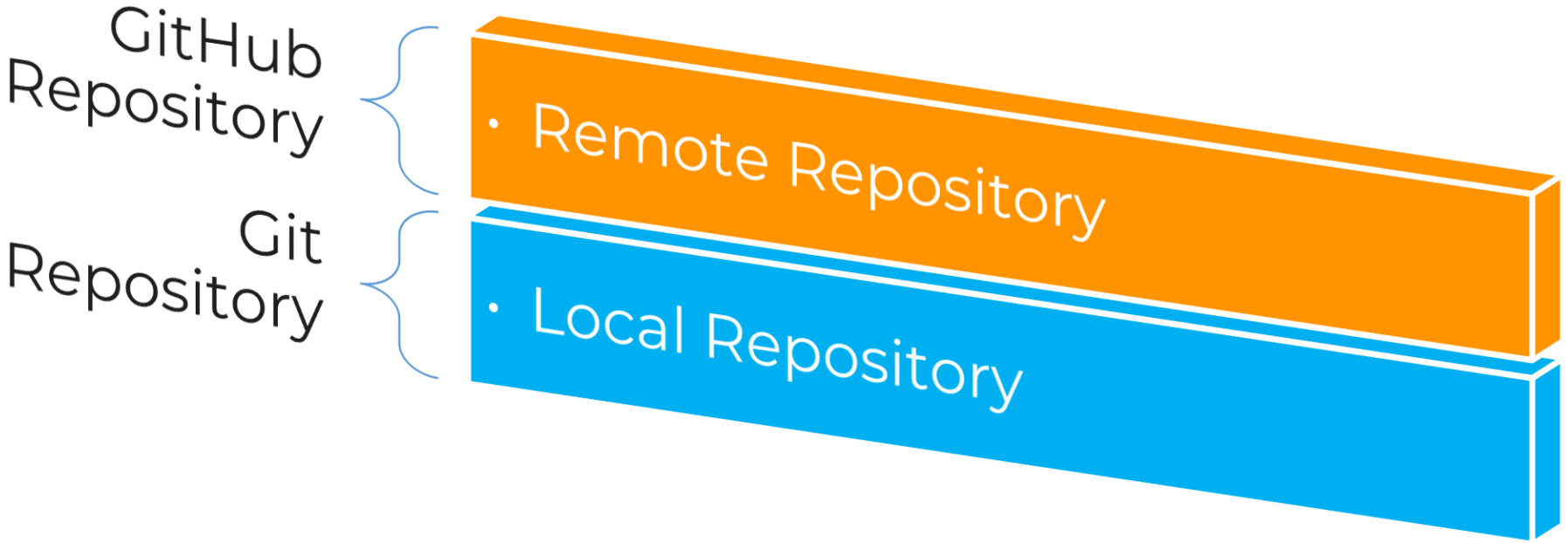
- **System** parametresi kullanıldığında tüm kullanıcılar ve tüm repolar üzerinde etkili olur
- **Global** parametresi geçerli kullanıcının tüm repolar üzerinde etkili olur
- **Local** parametresi ise sadece geçerli repo üzerinde etkili olur

GENEL KAVRAMLAR

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 VERSIYONLAR OLUŞTURMA

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

git status

Oluşturulan versiyonları görmek için bu komut kullanılır

git log


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

```
git commit –m"bir mesaj"
```

VERSIYON 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 IÇIN KODLAR

Ana komutlar

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

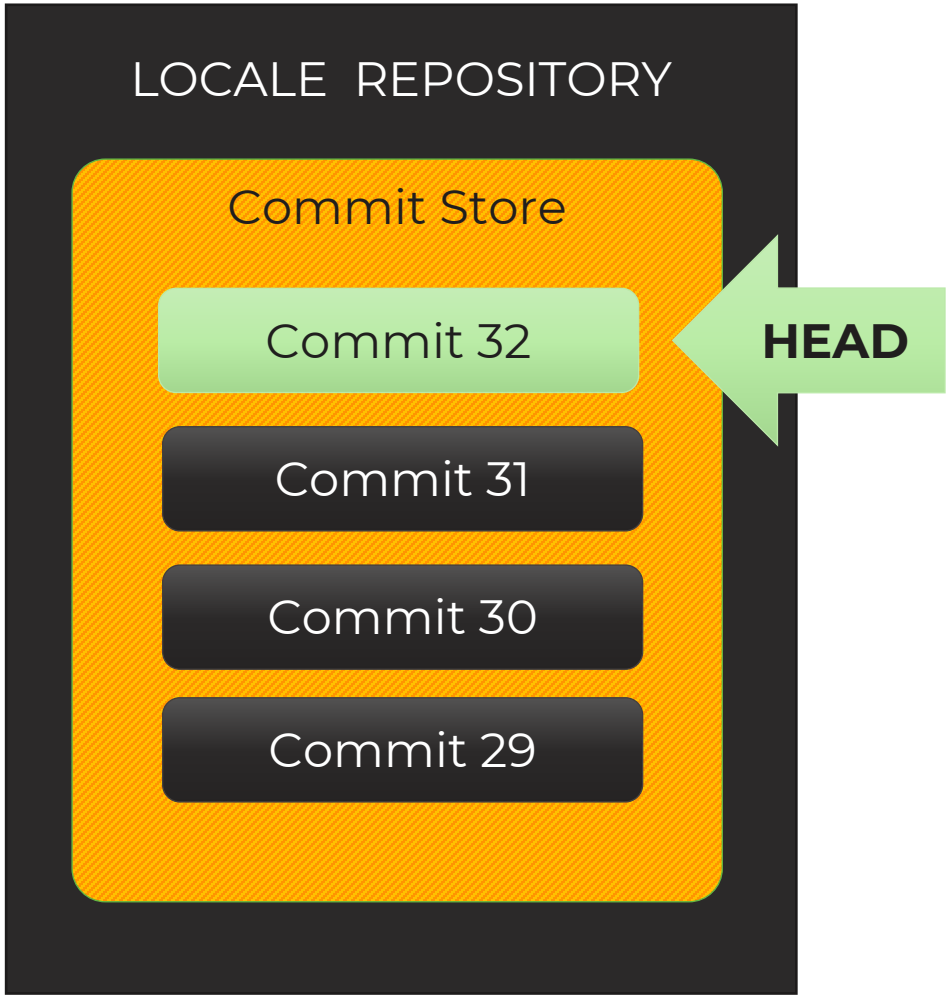
- `git init` → Repo oluşturur. Her projede en başta bir kere kullanılır.
- `git add .` → Dosyaları staging area ya gönderir
- `git commit –m "versiyon metni"` → Versiyon oluşturur

Yardımcı komutlar

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

- `git status` → Genel durum ile ilgili bilgi verir
- `git log` → Versiyonların listesini verir
- `git show [hash_kodu]` → Versiyondaki değişiklikleri gösterir

COMMIT STORE & HEAD

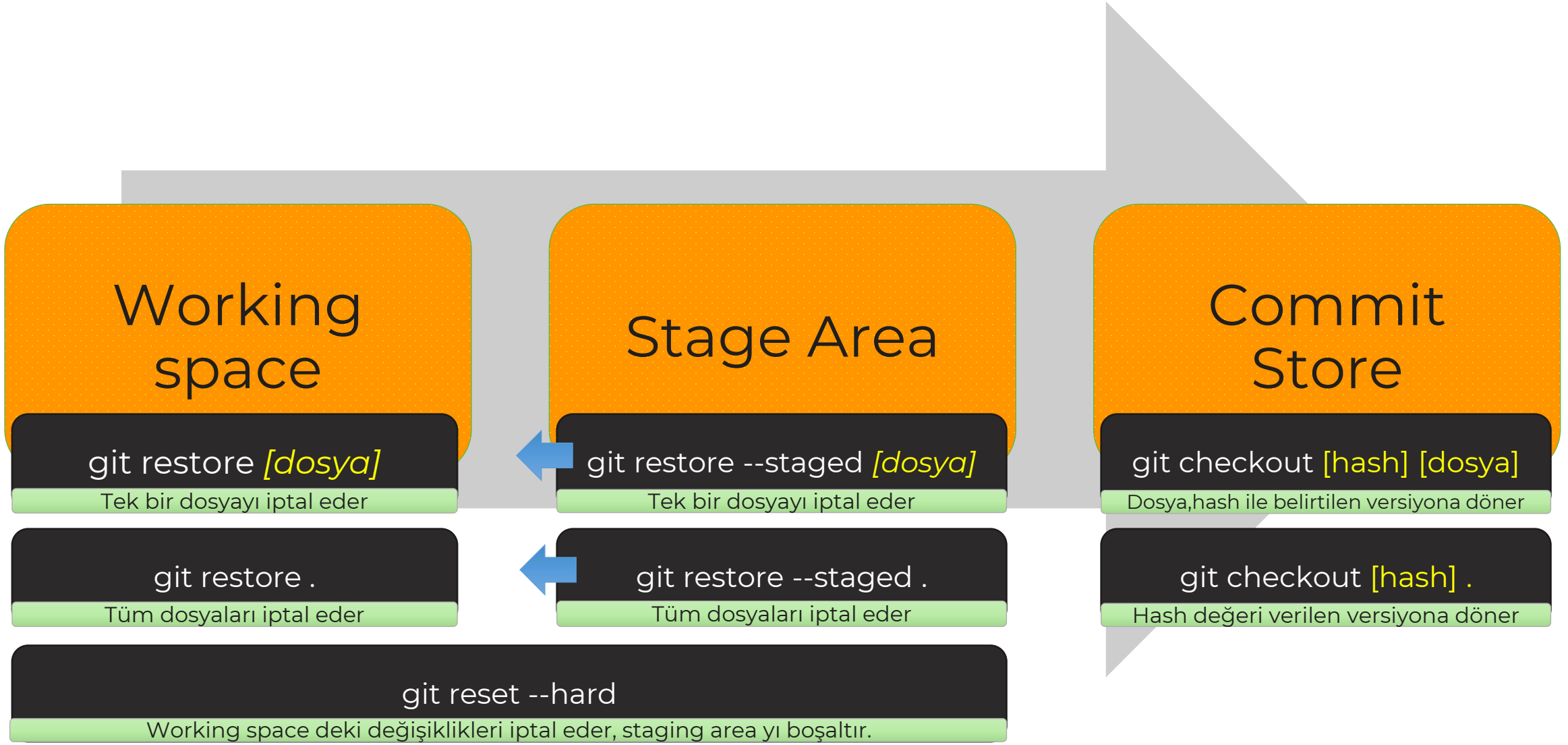


- Bir repo içinde birden fazla commit olabilir. Bunlardan en son alınan commit' e **HEAD** denir.
- Bu HEAD değiştirildiğinde önceki versiyonlara dönüş yapılabilir.

```
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
```

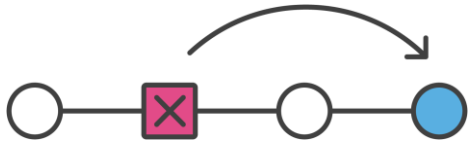
HEAD

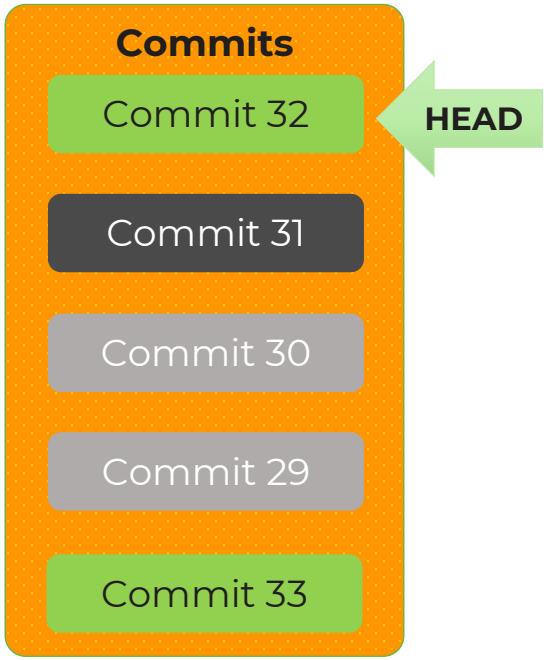
DEĞIŞIKLIKLERI IPTAL ETMEK



ÖNCEKI VERSIYONLARA DÖNMEK

1.Yöntem: CHECKOUT



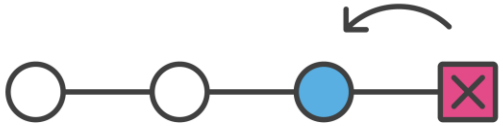


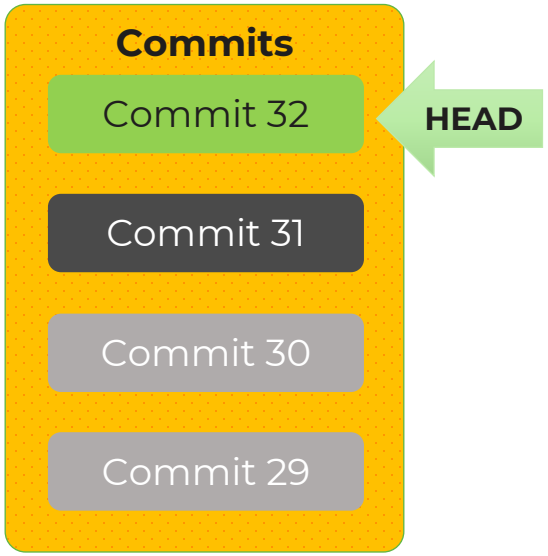
Önceki veriyonu incelemek için
git checkout *[hash]* .

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

ÖNCEKI VERSIYONLARA DÖNMEK

2.Yöntem: RESET

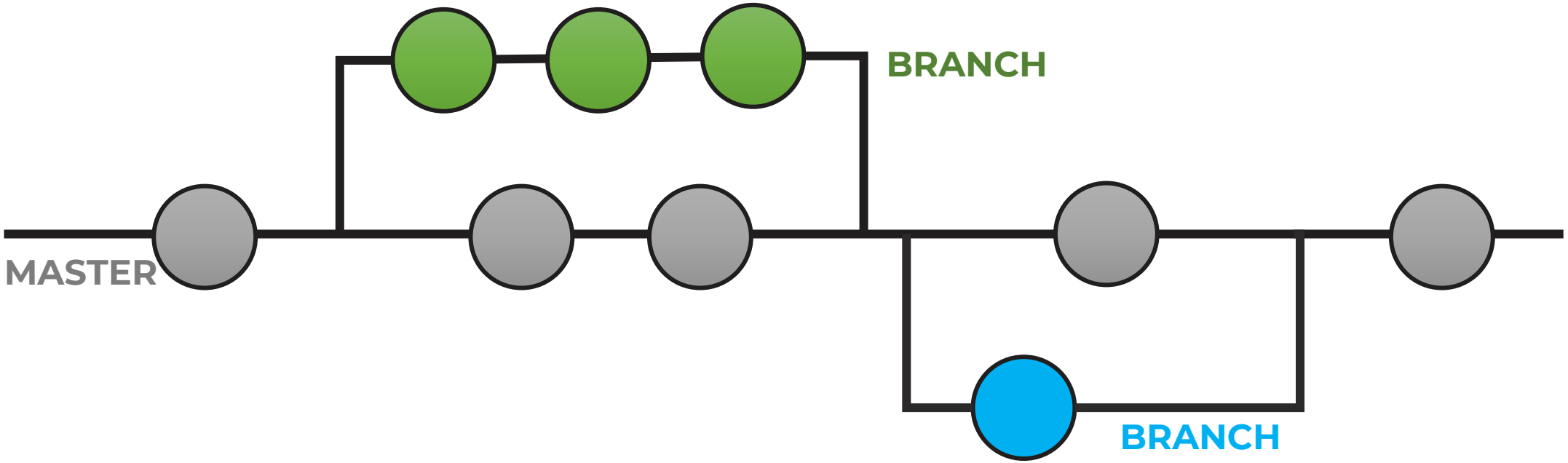




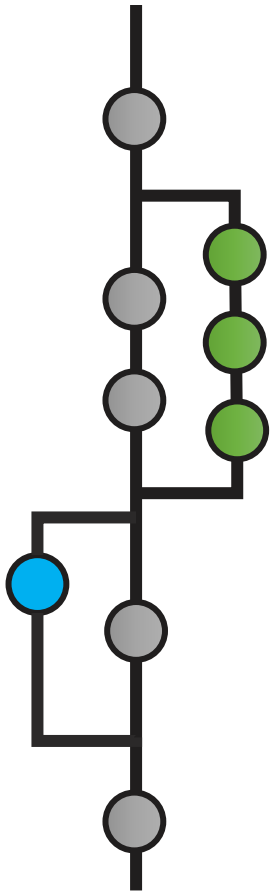
Geri alınamayacak şekilde önceki versiyona dönmek

git reset --hard *[hash]*

BRANCH (DAL)



BRANCH LERIN FAYDALARI



- Orginal 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]*	git checkout *[isim]*	git branch -m *[isim]*
Yeni branch oluşturur	**Branch aktif hale gelir**	**Branch ismini değiştirir.**
git branch	git merge *[isim]*	git branch –d *[isim]*
Mevcut branch leri listeler	**İki branch i birleştirir**	**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	git stash list	git stash pop
Working space ve staging area daki değişiklikleri geçici olarak hafızaya alır ve bu bölgeleri temizler	Hafızaya alınan değişiklikleri görmek için kullanılır	Hafızaya alınan değişiklikleri geri uygulamak için kullanılır.

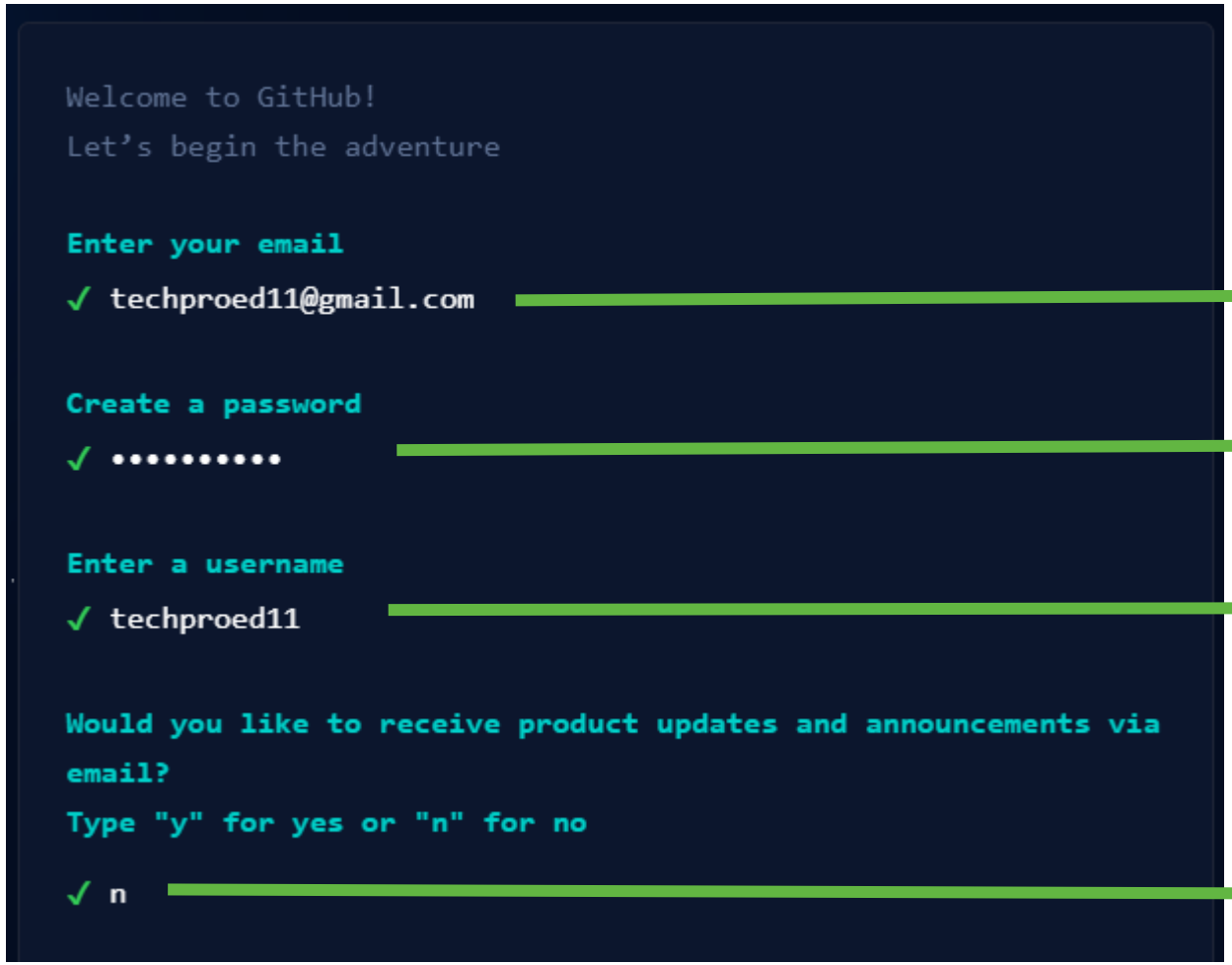
- Hesap oluşturma
- Repo oluşturma
- Genel kavramlar
- Github çalışma prensibi
- Clone
- Push & Pull
- Gitignore
- Merge & Conflicts

Github

GITHUB HESABI OLUŞTURMA

Github.com

GITHUB HESABI OLUŞTURMA



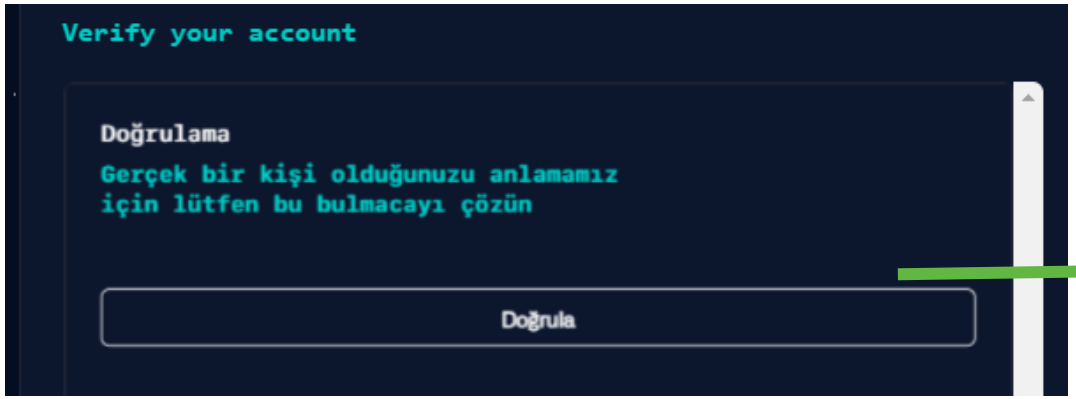
Eposta adresinizi giriniz

Şifre belirleyiniz

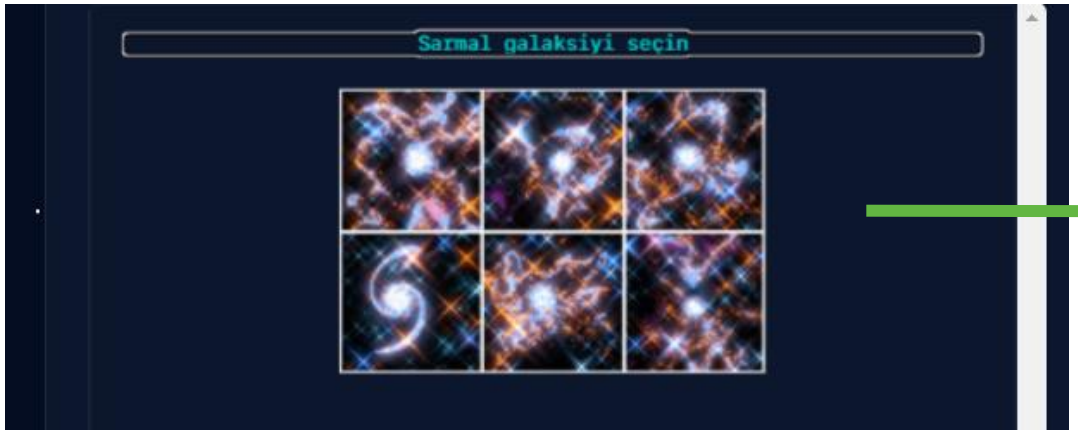
Bir kullanıcı adı belirliyoruz

Ürün güncelleştirmeleri ve tanıtımlardan email yoluyla haberdar olmak istemiyorsak n yazıyoruz

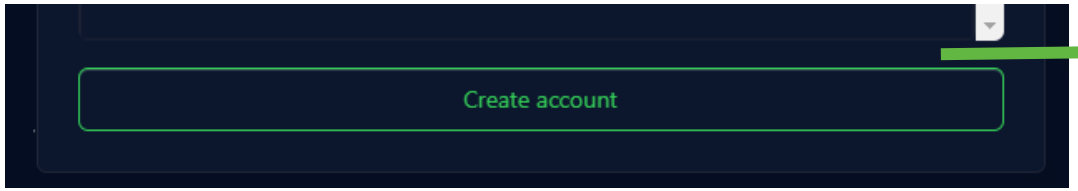
GITHUB HESABI OLUŞTURMA



Doğrula butonuna basıyoruz

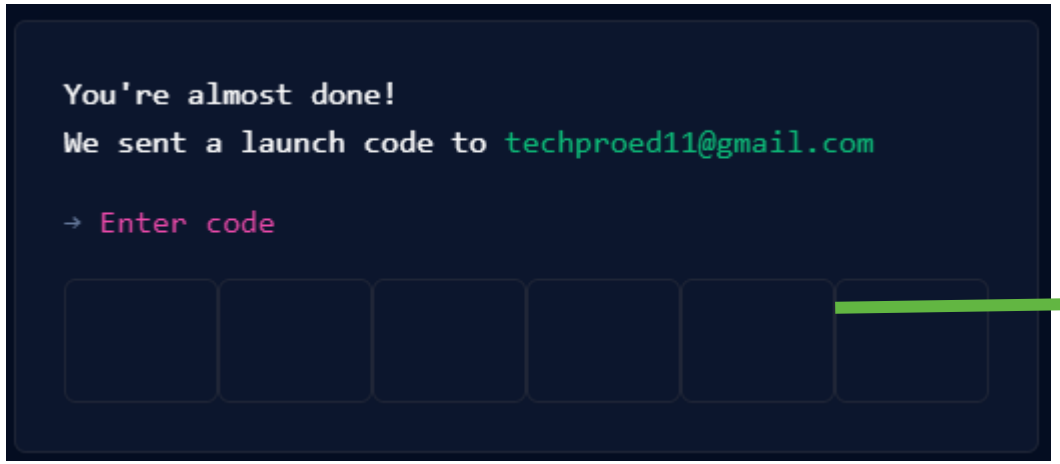


Doğrulama adımlarını geçiyoruz



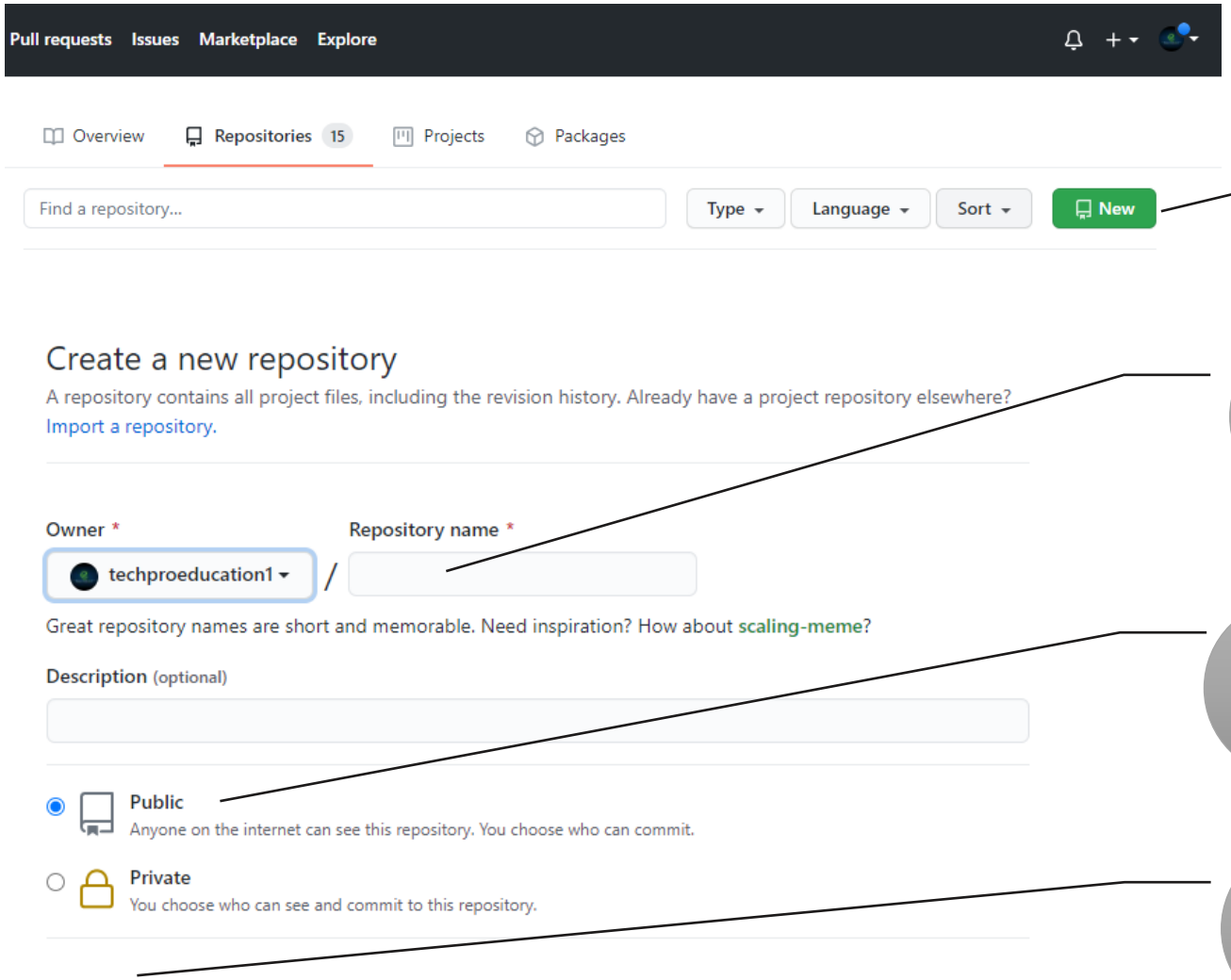
Create Account butonuna basıyoruz

GITHUB HESABI OLUŞTURMA



Eposta adresine gönderilen kodu girerek işlemi tamamlıyoruz

GITHUB REPO OLUŞTURMA



1 New butonuna basılır

2 Repository için bir isim veriyoruz

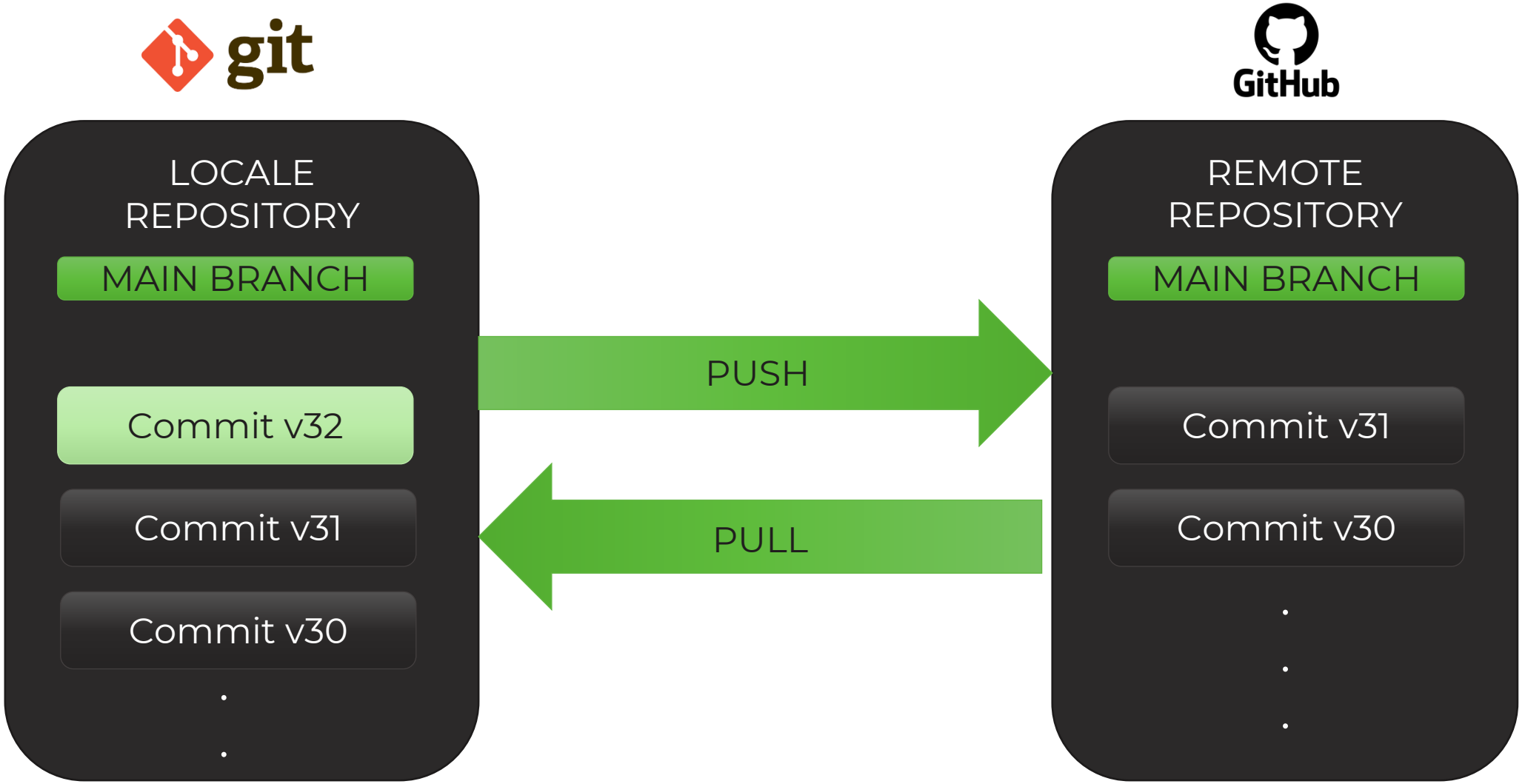
3 Herkes tarafından ulaşılabilir mi olsun, yoksa sadece bizim belirlediğimiz kullanıcılar mı ulaşabilsin

4 Create repository butonuna basılır

KAVRAMLAR

Clone	Github daki bir repoyu lokale indirme işlemidir
Push	Lokalde oluşturulan commit lerin github a gönderilmesi işlemidir.
Pull	Fetch ve Merge işlemini tek başına yapar

GITHUB ÇALIŞMA PRENSIBI



CLONING

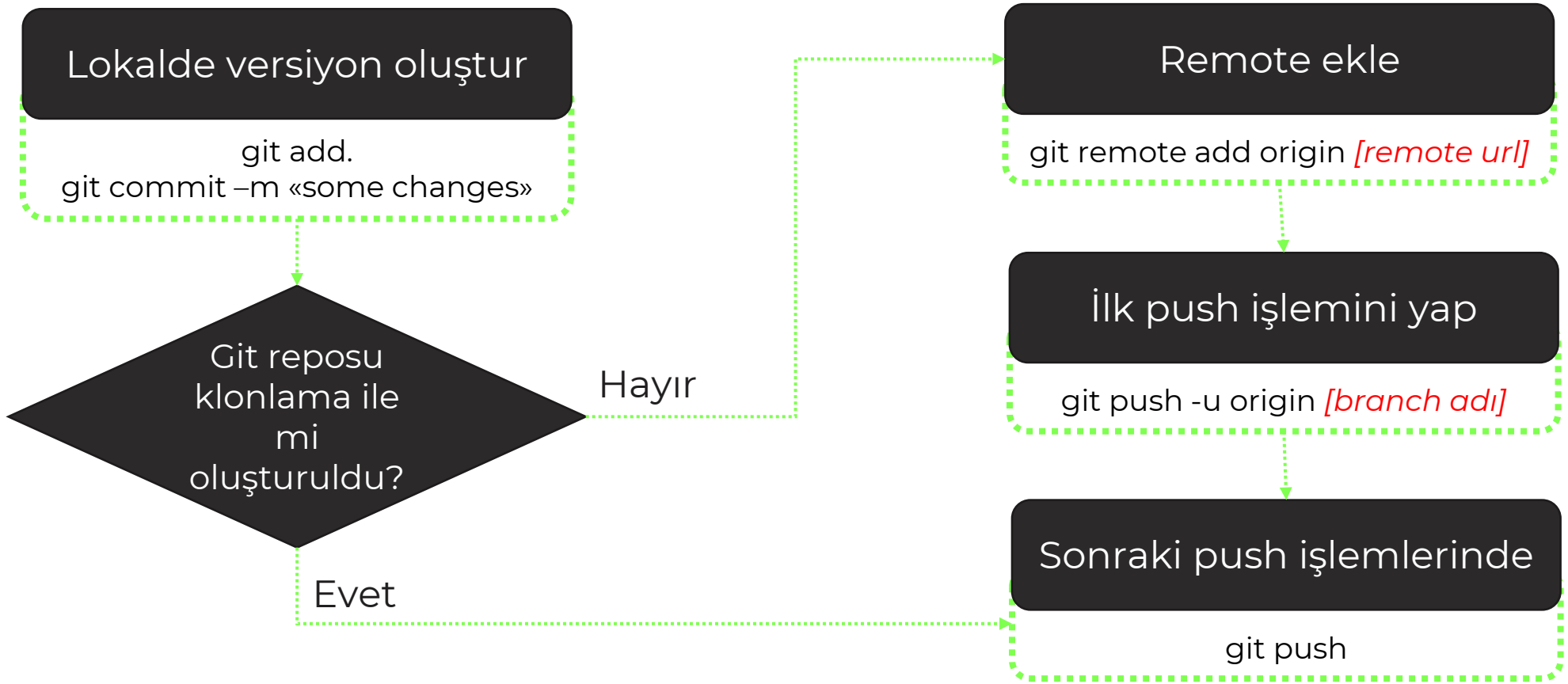
git clone

Github daki bir reponun lokale indirilmesi işlemine klonlama denir. Public olan veya gerekli izinlere sahip olunan private repolar klonlanabilir.

Bunun için git clone komutu kullanılır.

git clone https://github.com/techproeducation-batchs/B-71-FED-TR.git

GITHUB'A YÜKLEME (PUSHING)



GITHUB DAN COMMIT ÇEKME (PULLING)

Github üzerinden local repo güncellenmek istenirse aşağıdaki komutlar kullanılır

git fetch

Değişiklikleri remote'dan local'e indirir

git merge

İndirilen değişiklikleri local repoya uygular

VEYA

git pull

fetch & merge

TECHPRO EDUCATION

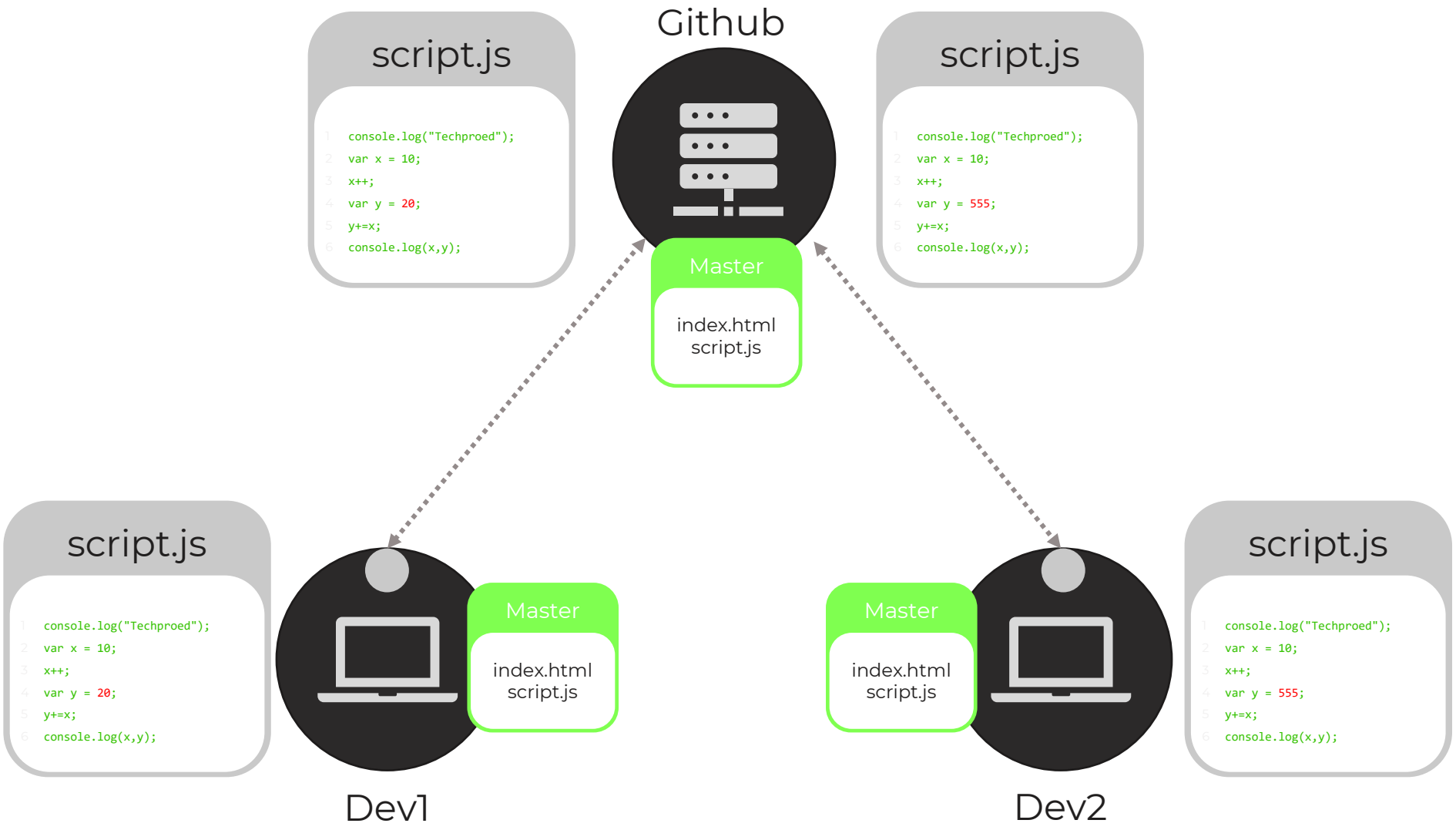
GITIGNORE

Staging area' ya gitmesini istemediğimiz, yani versiyon kontrol sistemine dahil etmek istemediğimiz dosya ve klasörlerimizi tanımladığımız özel bir dosyadır.

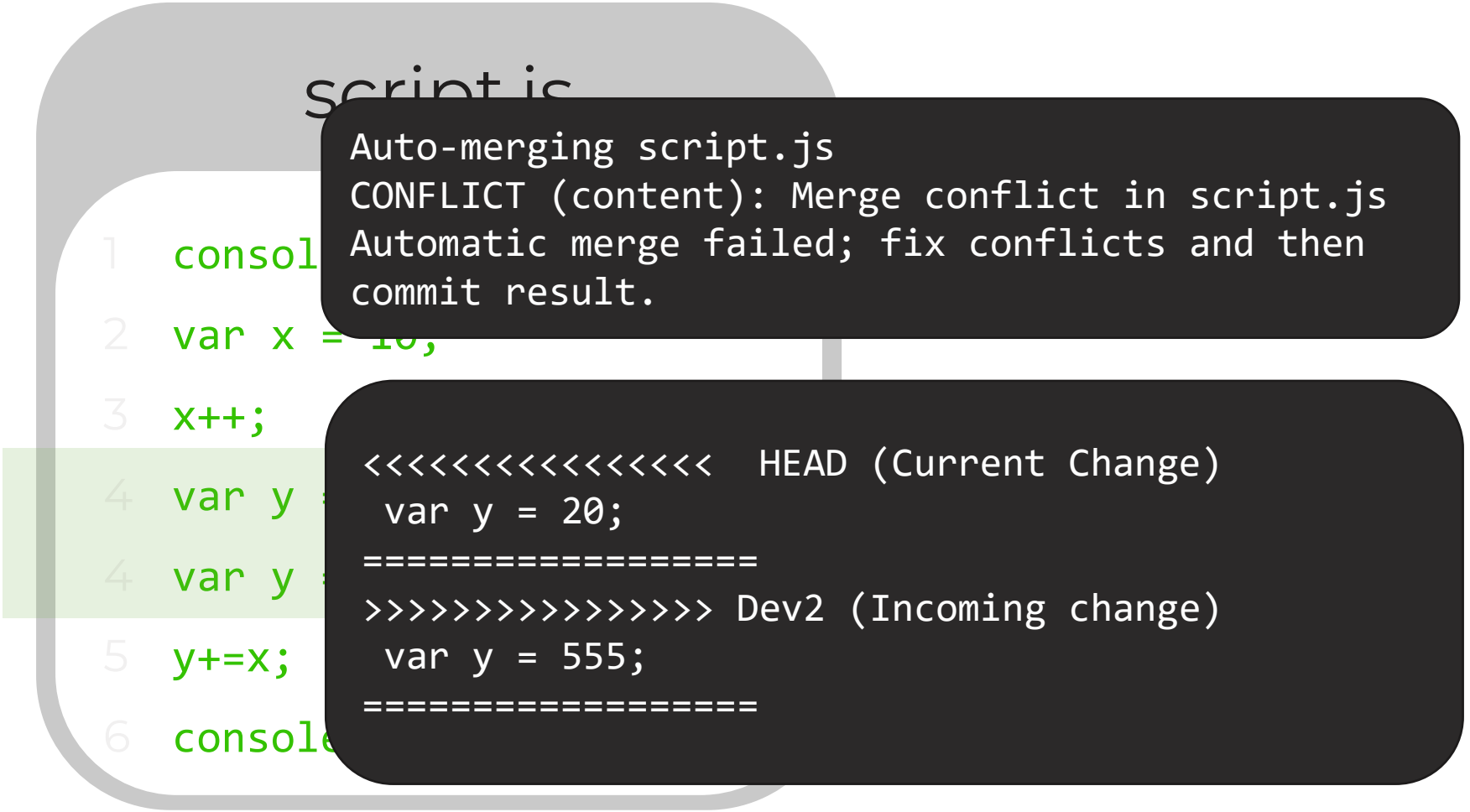
```
out/
.idea/
.idea_modules/
*.iml
*.ipr
*.iws
```

.gitignore

MERGE CONFLICT

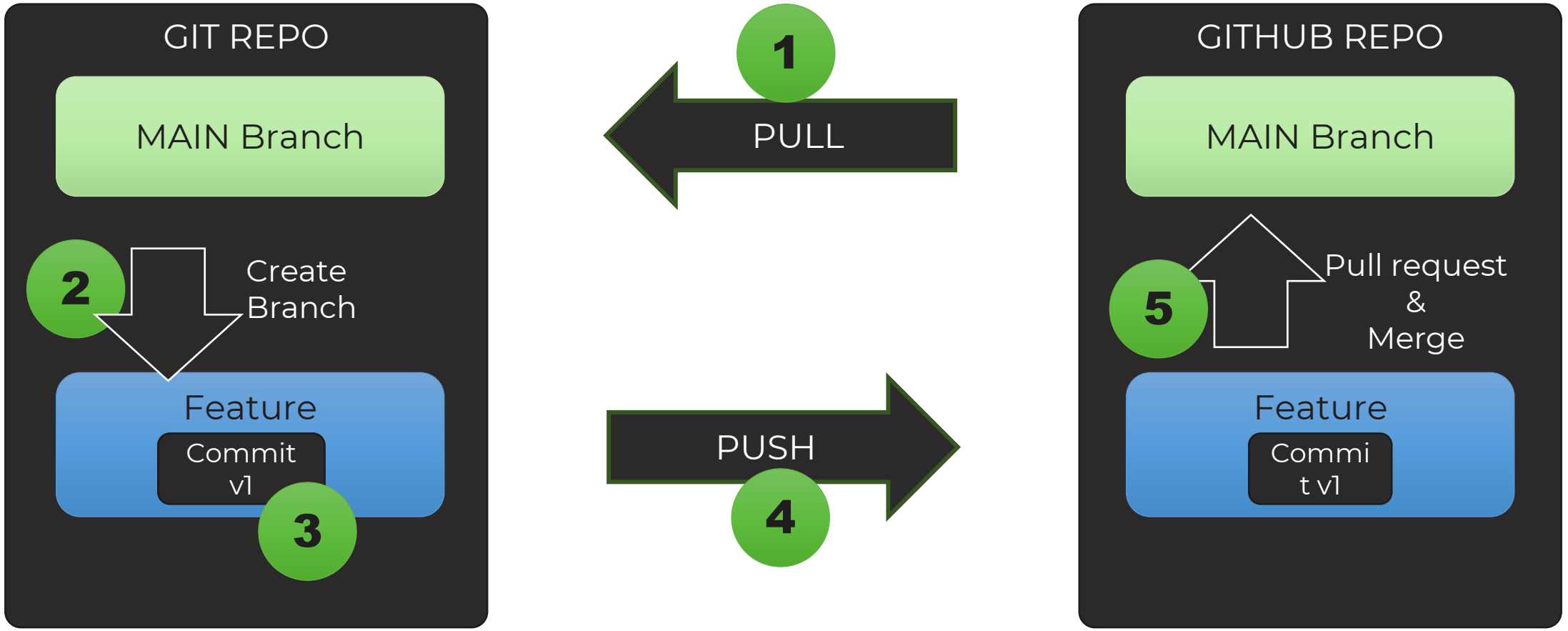


MERGE CONFLICT



GIT – GITHUB ÇALIŞMA DÖNGÜSÜ

BEST PRACTISE



TECHPRO EDUCATION

KAHOOT

Kahoot!


INTERVIEW