BATCH : 106-116

LESSON : Git-Github

DATE : 01.11.2022

SUBJECT : Git-Github

techproeducation

techproeducation

techproeducation

techproeducation

techproedu

techproeducation.com

info@techproeducation.com

+1 (917) 768-7466

Git - Github

Git - Github

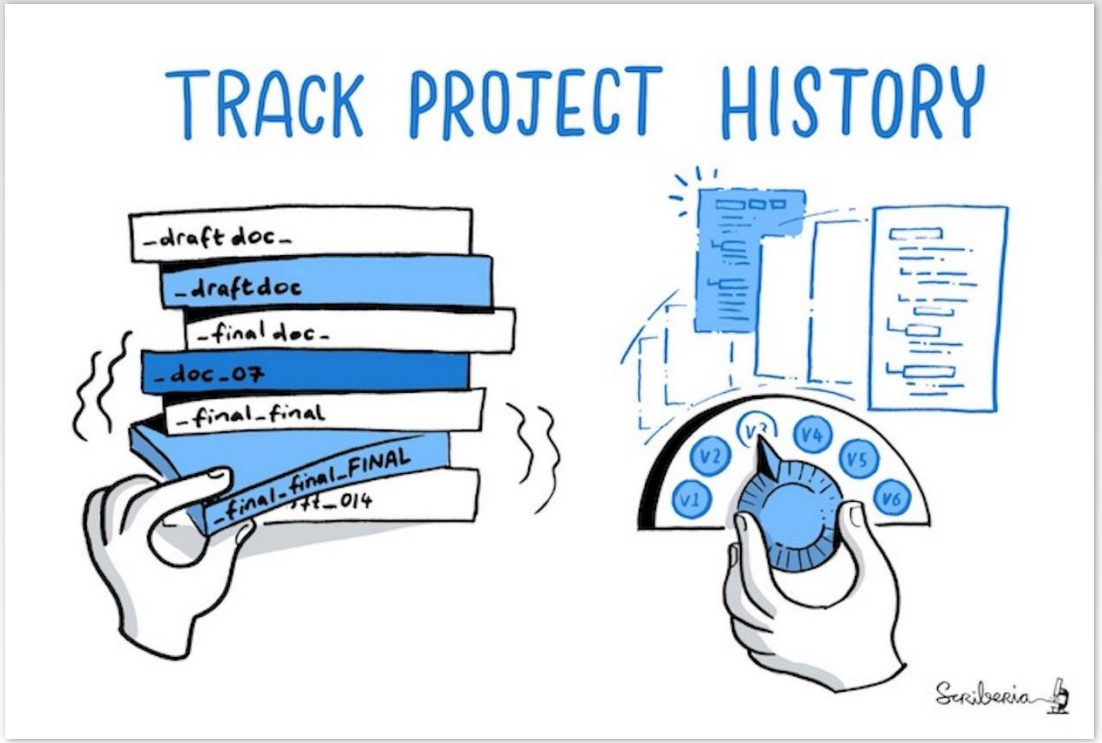
- Git - Github nedir?
- VKS Nedir?
- Ne amaçla kullanılır

Fundamentals

Git-Github nedir

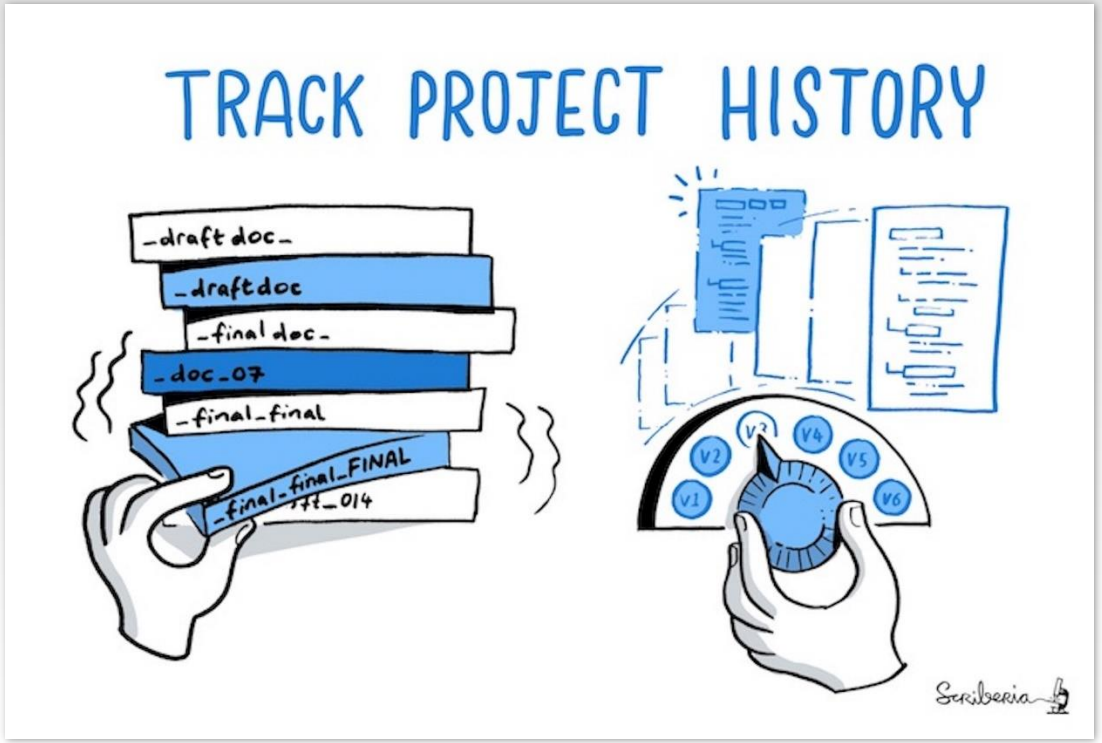
Git-Github versiyon kontrol sistemidir

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

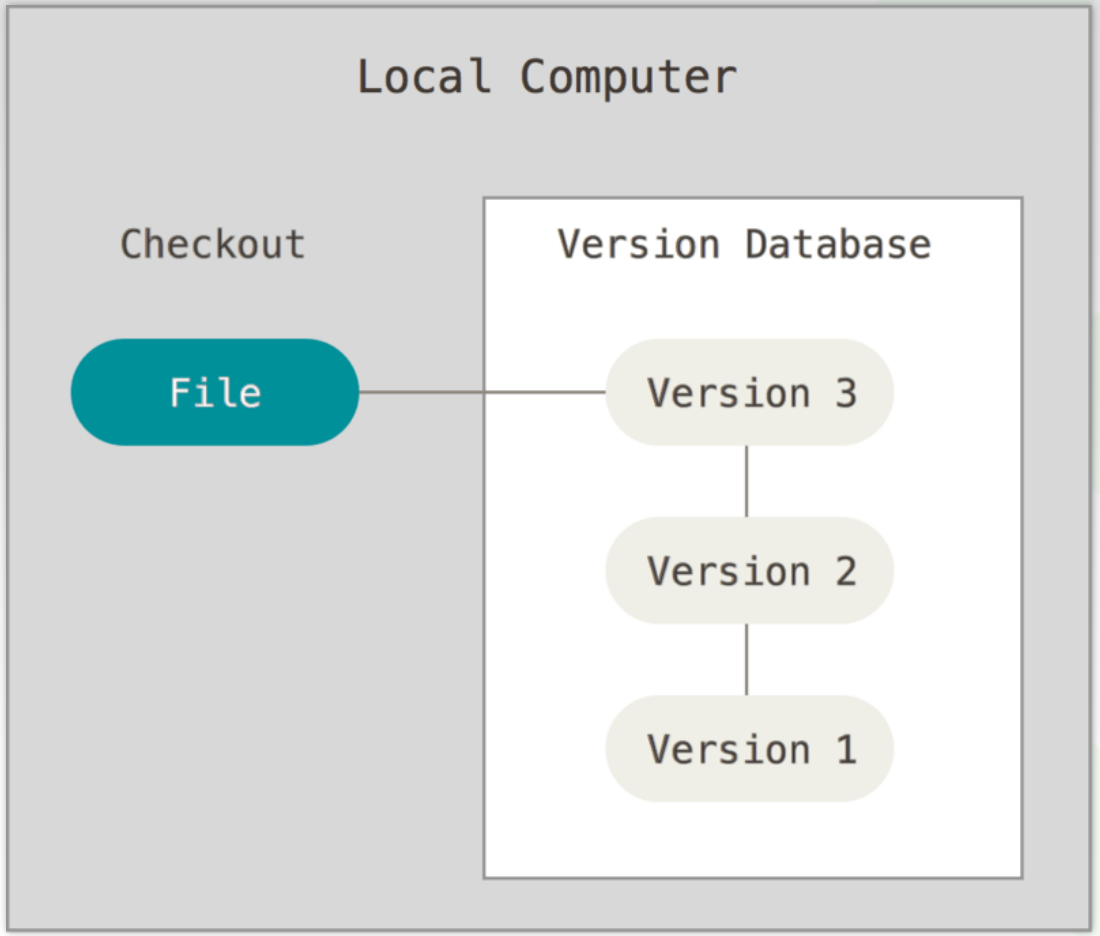
3 tip Versiyon Kontrol Sistemi vardır.

YEREL

MERKEZİ

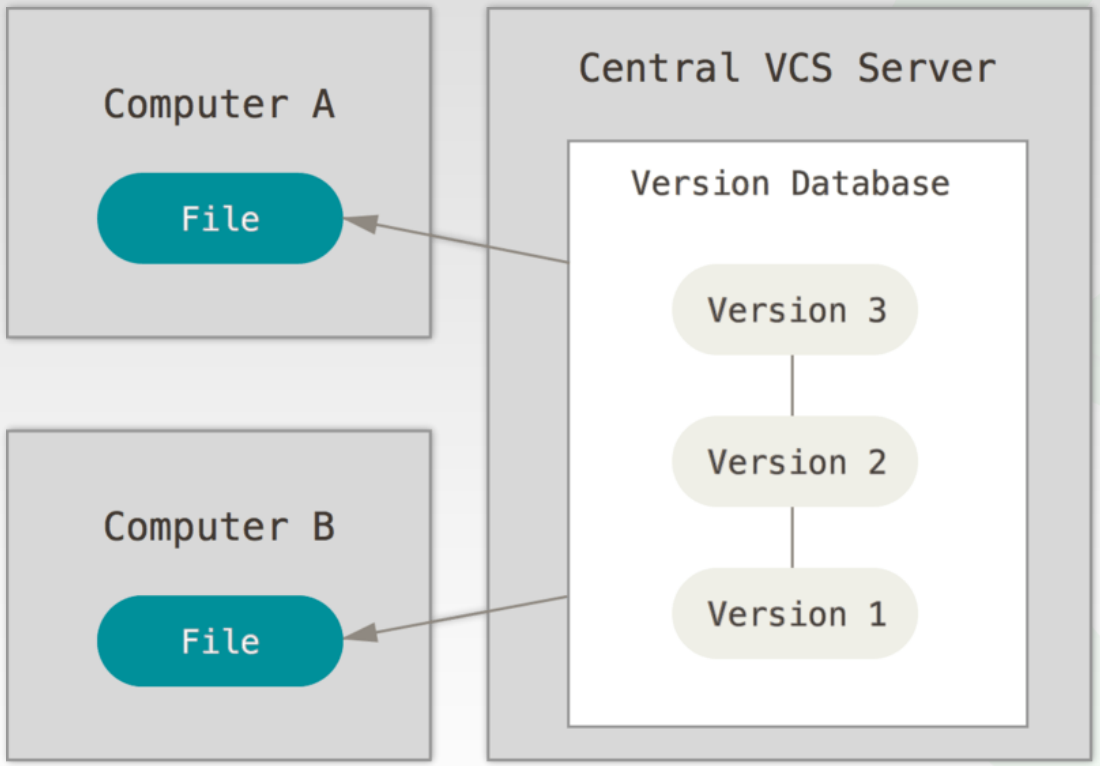
DAĞITIK

Yerel VKS nedir



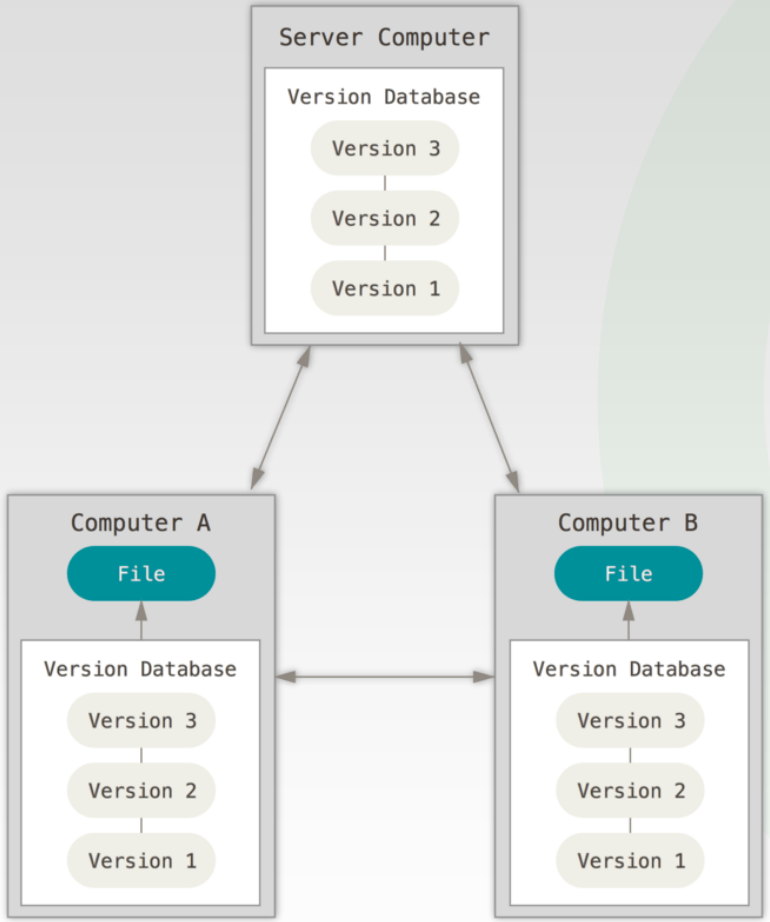
https://git-scm.com/book/en/v2/Getting-Started-About-Version-C

Merkezi VKS nedir



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

Dağıtık VKS nedir



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

Git-Github Ne Amaçla Kullanılır

git LOCALE

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

GitHub REMOTE

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

Git - Github

- Kurulum ve ilk ayarlar
- Repository
- Lokal repo oluşturma
- Working space, staging area, commit store
- 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.

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 / git diff(Değişiklik görme)
Git diff --staged

Oluşturulan versiyonları görmek için bu komut kullanılır

git log --oneline


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

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

Versiyon detaylarını görme

git show

Bir versiyon içinde, hangi değişikliklerin olduğunu görmek için kullanır.

`git show` *`[hash kodun ilk 7 karakteri]`*

`git log --oneline`

```
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"
```

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
```

HEAD

Değişiklikleri iptal etmek

Working space

`git restore [dosya]`

Tek bir dosyayı iptal eder

`git restore .`

Tüm dosyaları iptal eder

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] .`

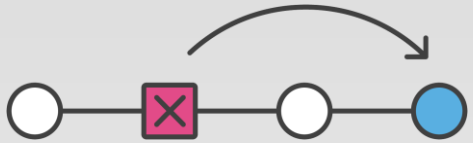
Hash değeri verilen versiyona döner

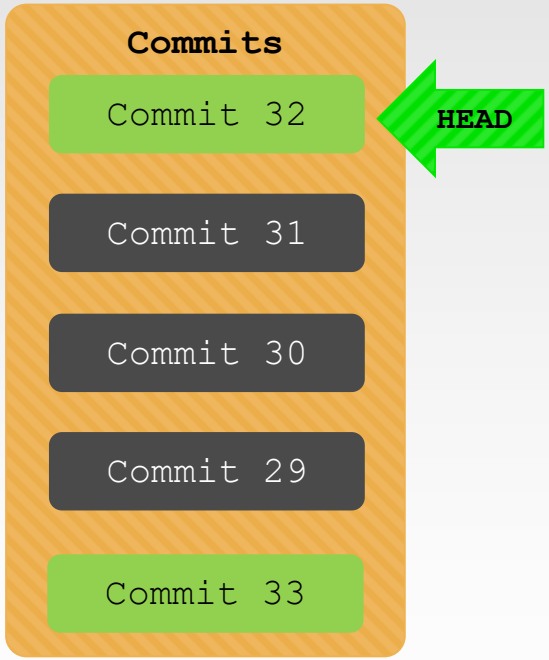
`git reset --hard`

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

Önceki versiyonlara dönmek

1.Yöntem: CHECKOUT





Önceki veriyonu incelemek için

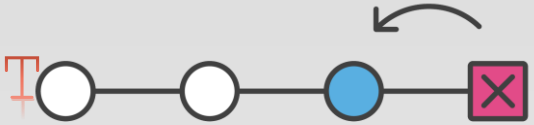
git checkout *[hash]* .

Bu işlemi kalıcı hale getirmek için

git commit –m"…"

Önceki versiyonlara dönmek

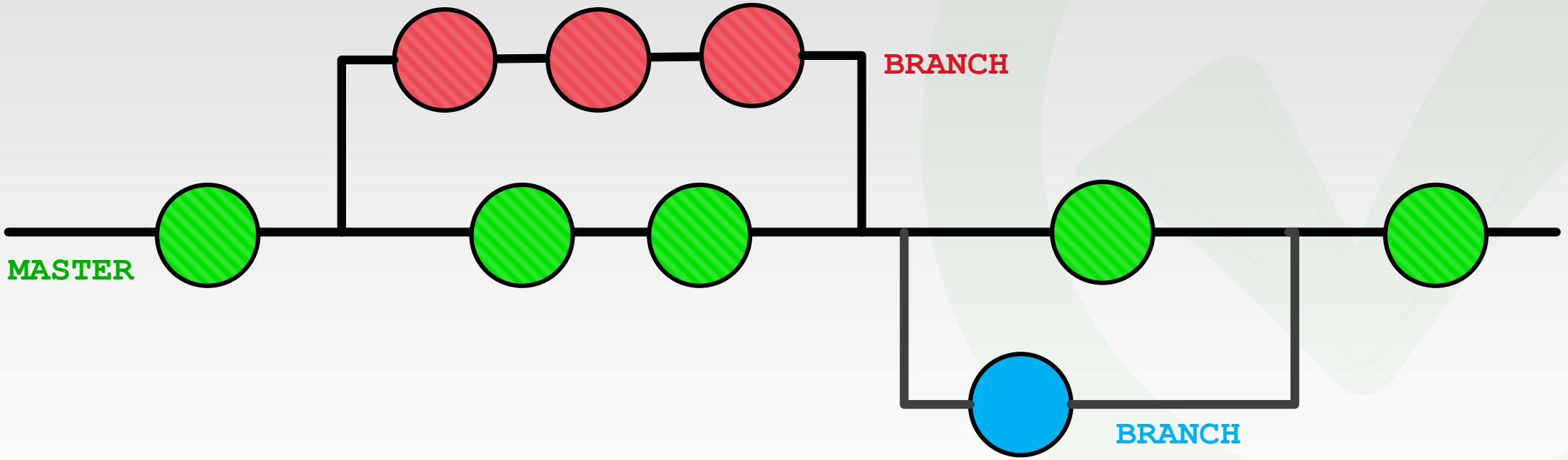
2.Yöntem: RESET



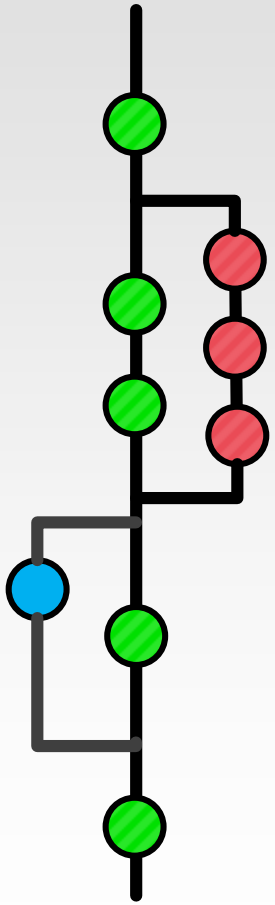
Geri alınamayacak şekilde önceki versiyona dönmek

git reset --hard
[hash]

Branch (Dal)



Branch lerin faydaları



- 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 Komutları

`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 space ve staging 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

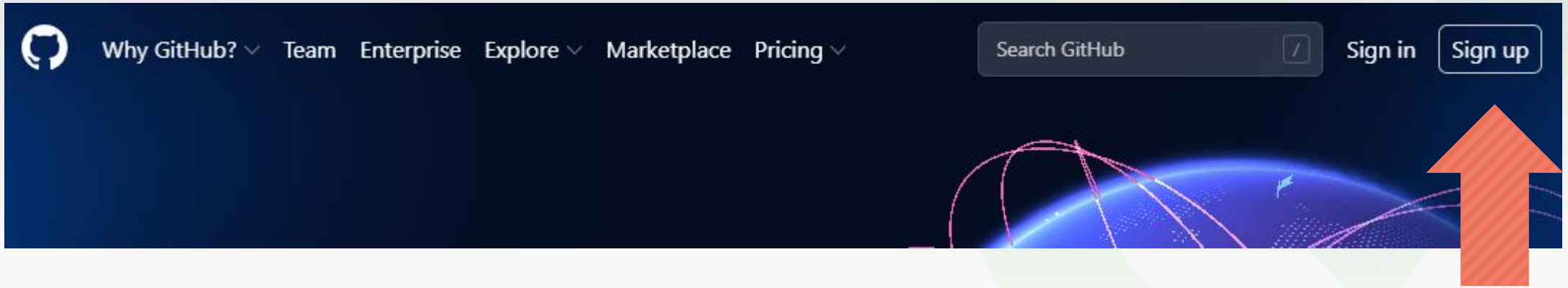
Hafızaya alınan değişiklikleri geri uygulamak için kullanılır.

Git - Github

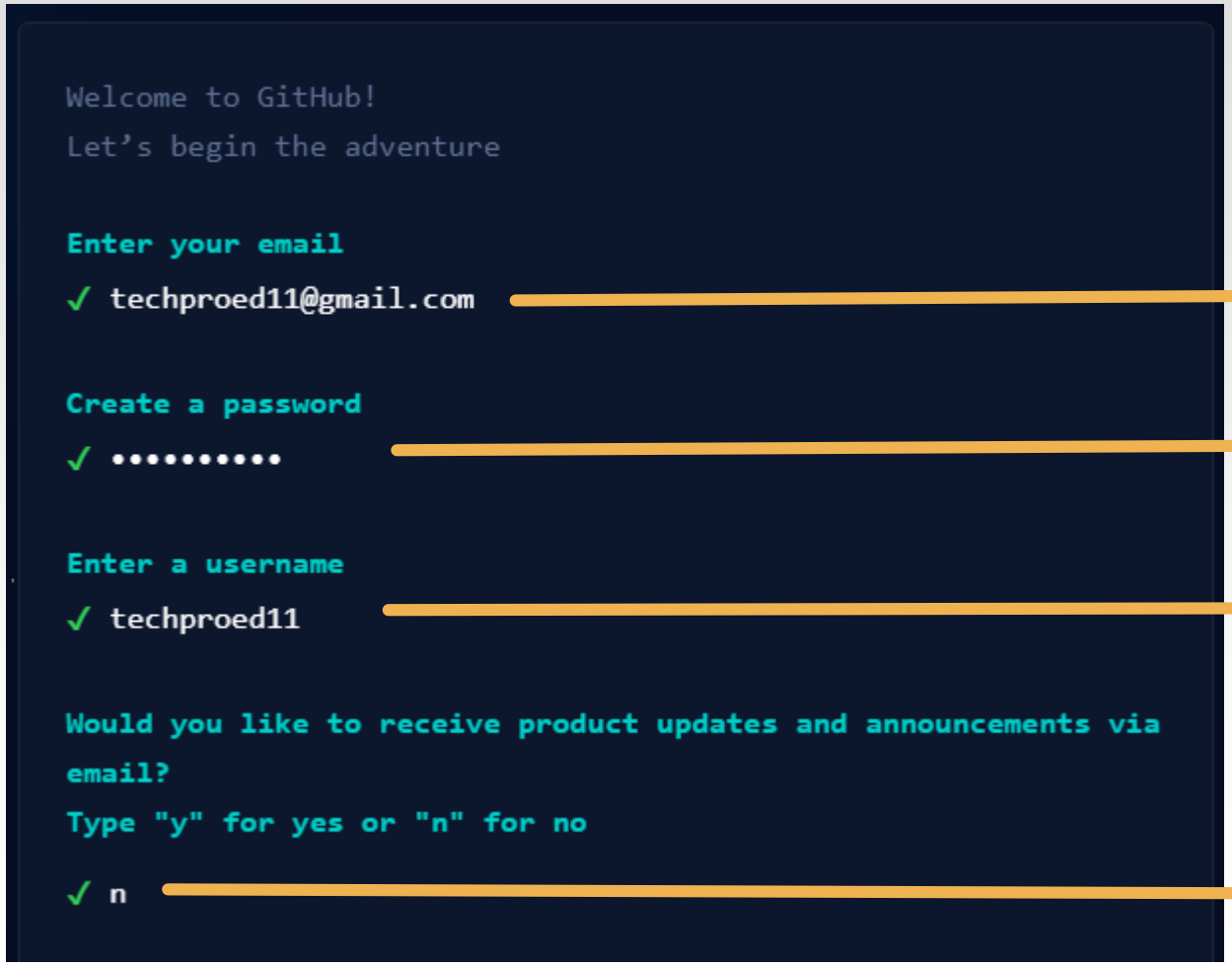
- Hesap oluşturma
- Repo oluşturma
- Genel kavramlar
- Github çalışma prensibi
- Clone
- Push & Pull
- Gitignore
- Merge & Conflicts

Github hesabı oluşturma

Github.com



Github hesabı oluşturma



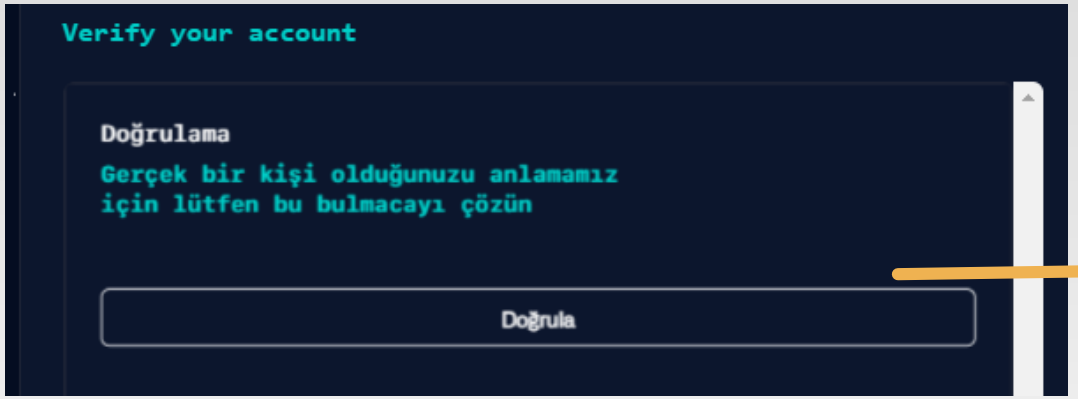
Eposta adresinizi giriniz

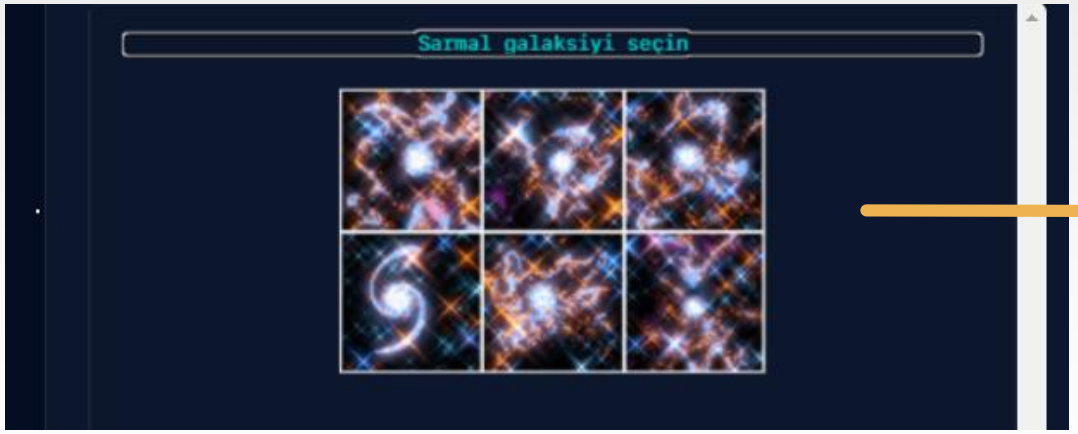
Şifre belirleyiniz

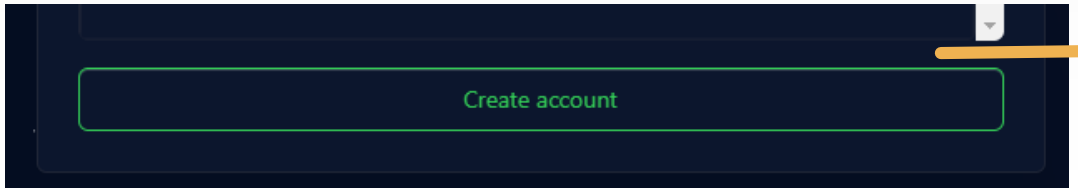
Bir kullanıcı adı belirliyoruz

Ürün güncelleştirmeleri ve tanıtımlardan email yoluyla haberdar olmak istemiyorsak n yazıyoruz

Github hesabı oluşturma





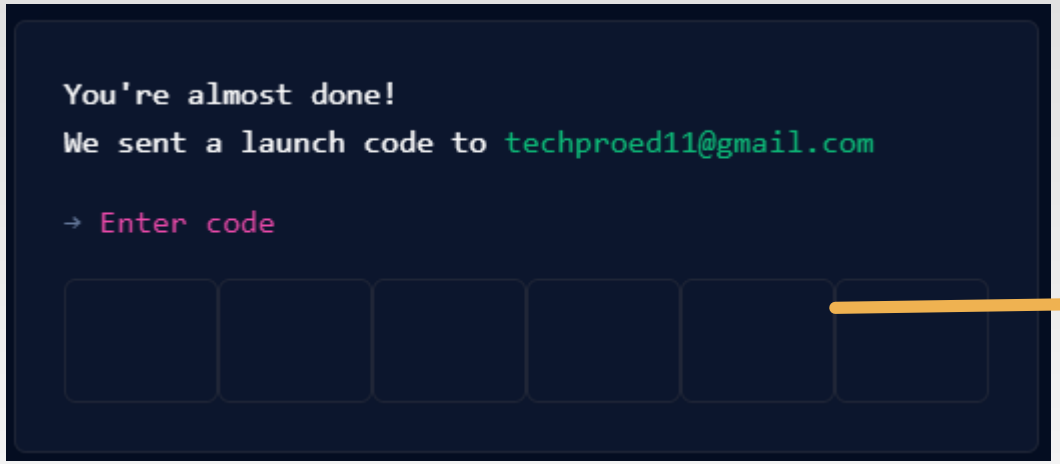


Doğrula butonuna basıyoruz

Doğrulama adımlarını geçiyoruz

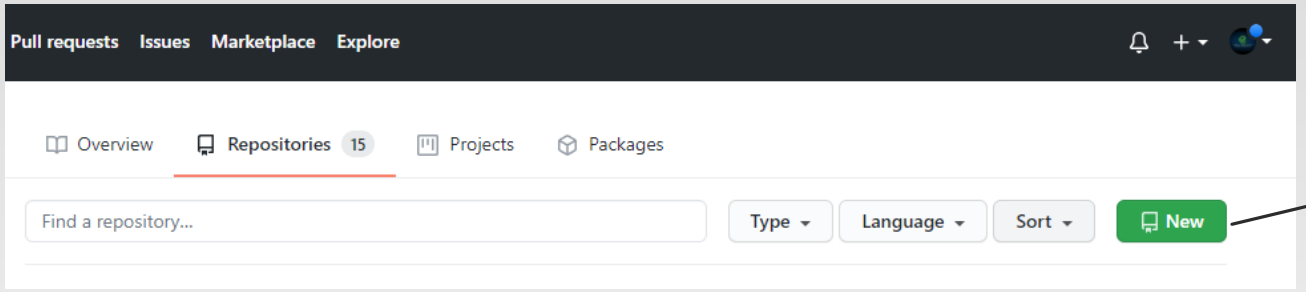
Create Account butonuna basıyoruz

Github hesabı oluşturma

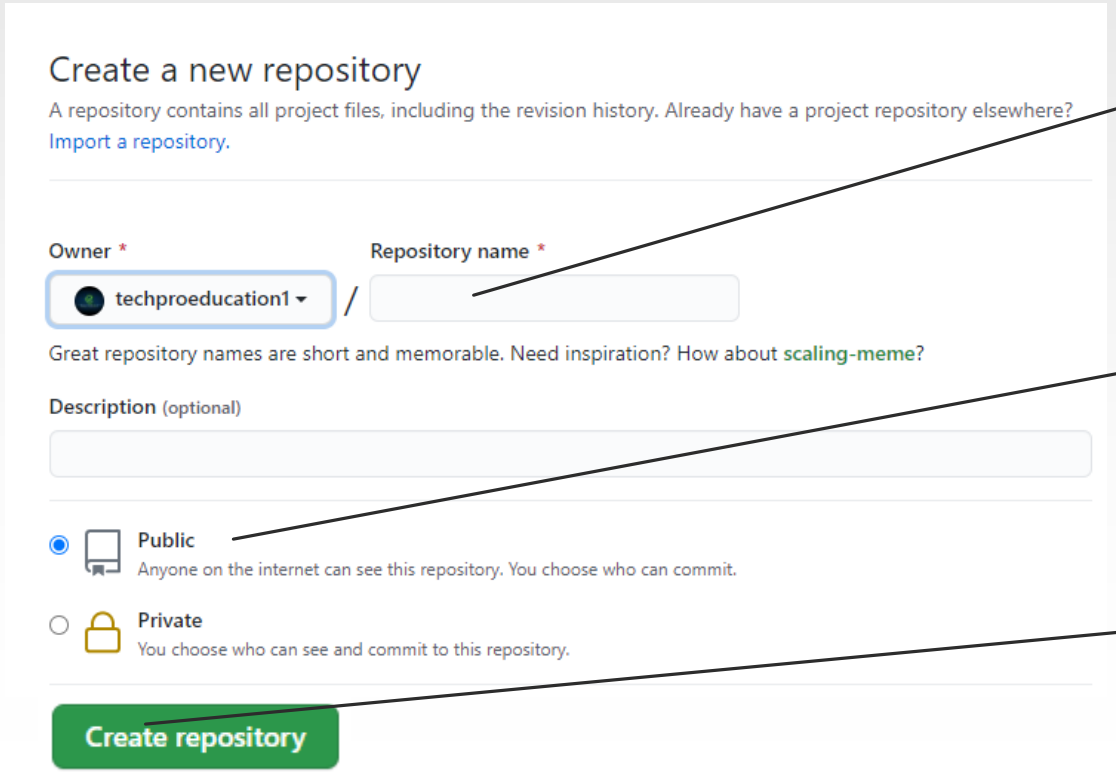


Eposta adresine gönderilen kodu girerek işlemi tamamlıyoruz

Github repo oluşturma



1 New butonuna basılır



2 Repository için bir isim veriyoruz

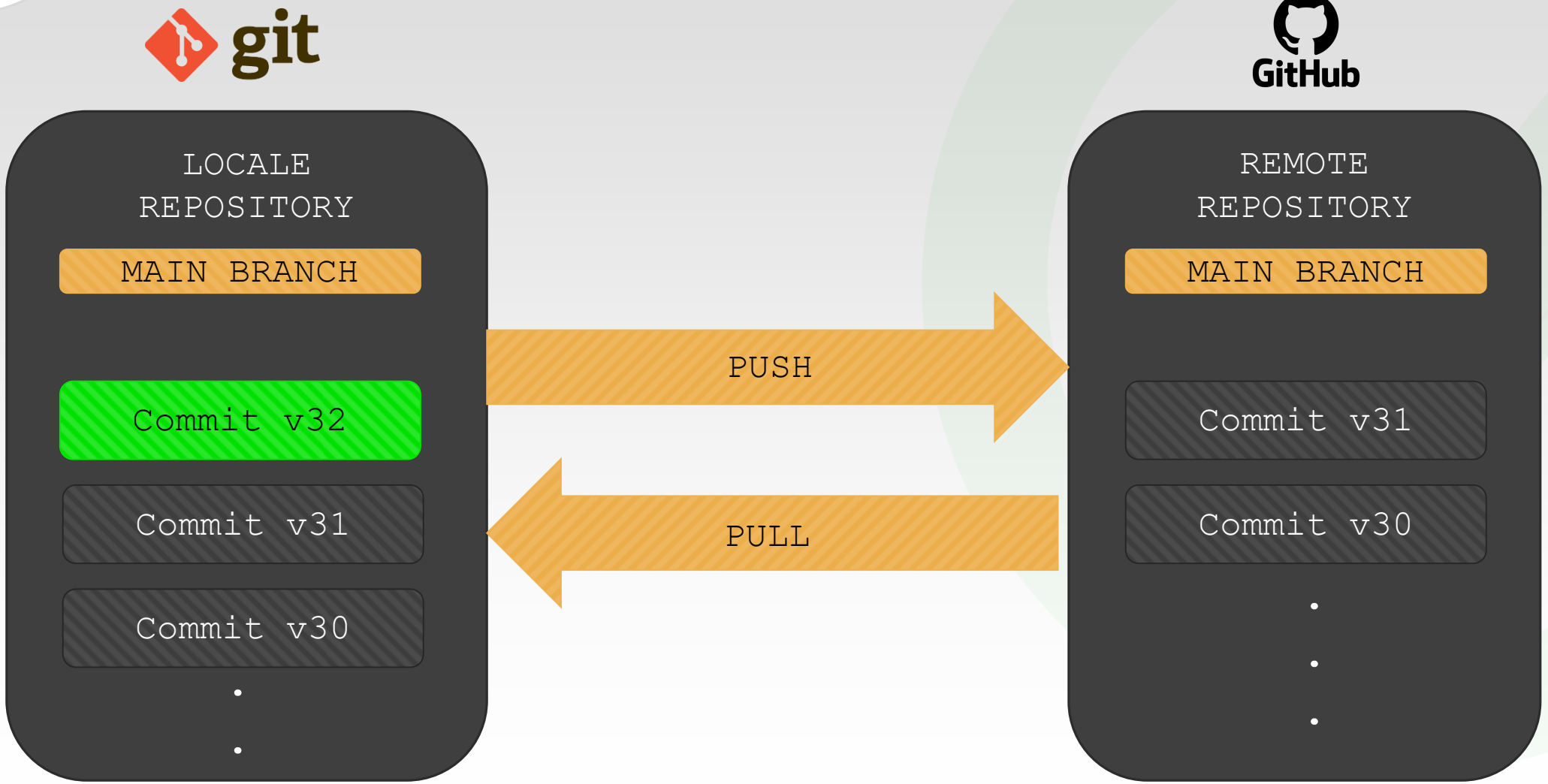
3 Herkes tarafından ulaşılabilir mı olsun, yoksa sadece bizim belirlediğimiz kullanıcılar mı ulaşabilsin

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 Çalışma Prensibi



Cloning

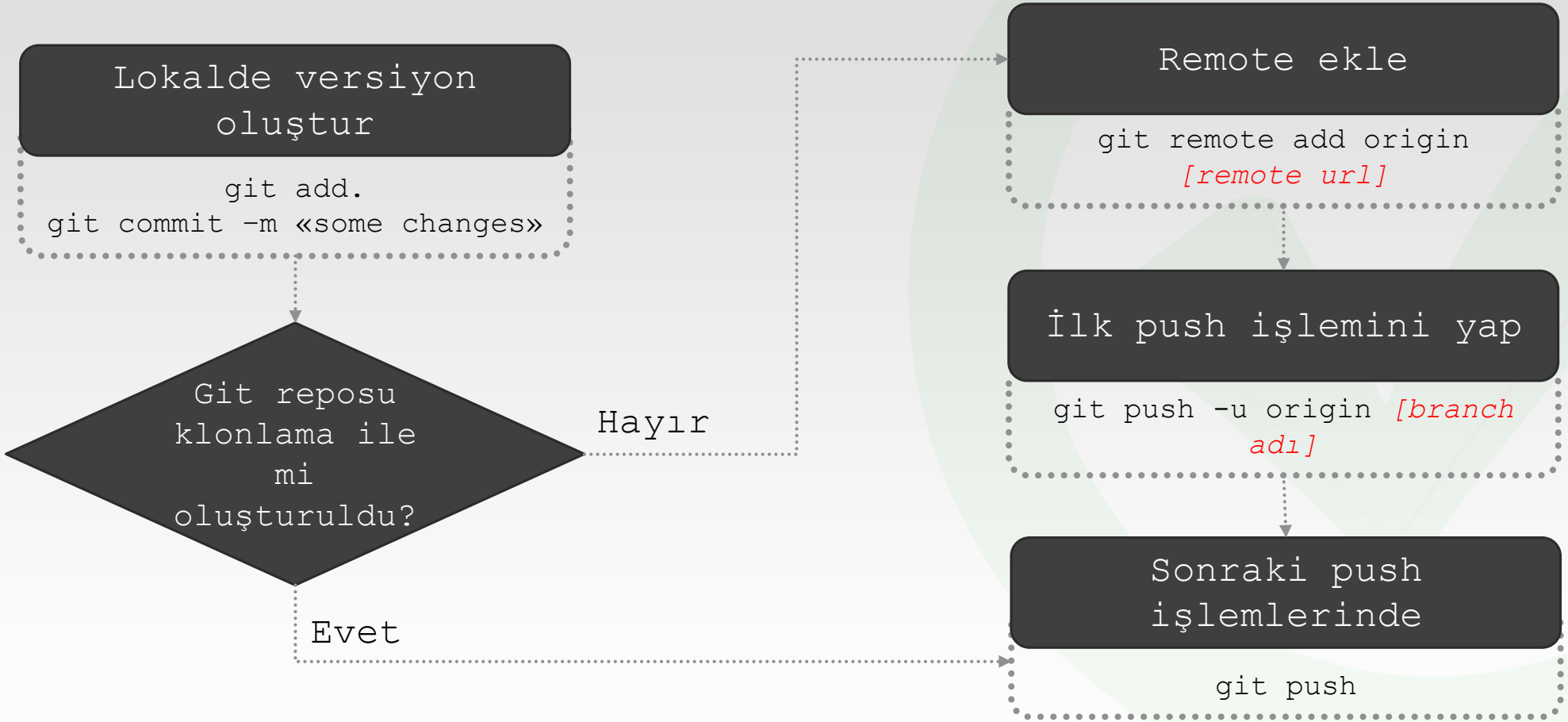
git clone

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

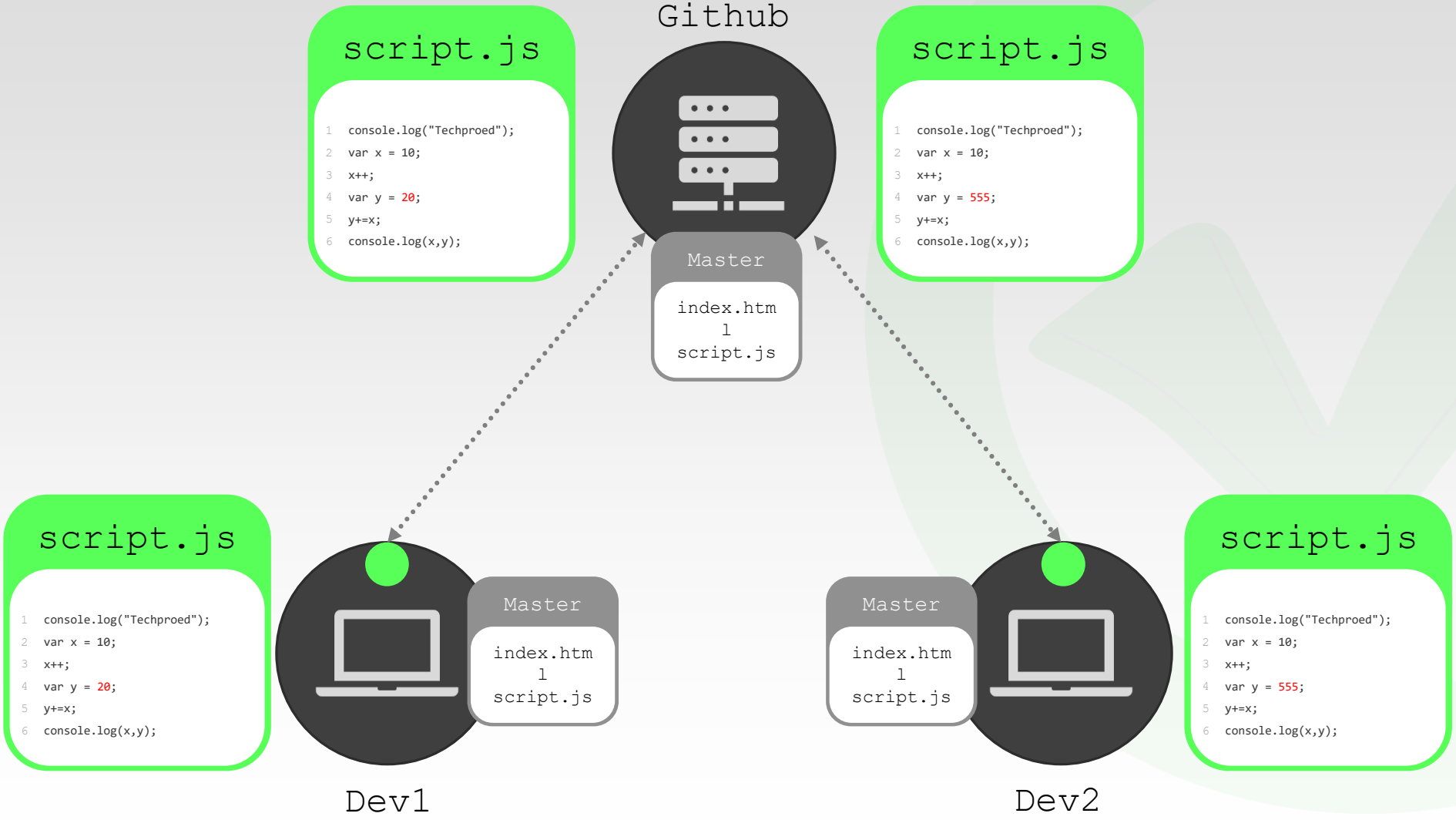
.gitignore

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

script.js

```
1  consol
2  var x =
3  x++;
4  var y
4  var y
5  y+=x;
6  console
```

```
Auto-merging script.js
CONFLICT (content): Merge conflict in script.js
Automatic merge failed; fix conflicts and then
commit result.
```

```
<<<<<<<<<<<<<<<  HEAD (Current Change)
 var y = 20;
==================
>>>>>>>>>>>>>>>> Dev2 (Incoming change)
 var y = 555;
==================
```

Git - Github Çalışma döngüsü

BEST PRACTISE