



BATCH : 107-128
LESSON : **Git-Github**
DATE : 19.12.2022
SUBJECT : **Git-3**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





Git - Github



Git - Github

Fundamentals

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



Git-Github nedir

Git-Github versiyon kontrol sistemidir



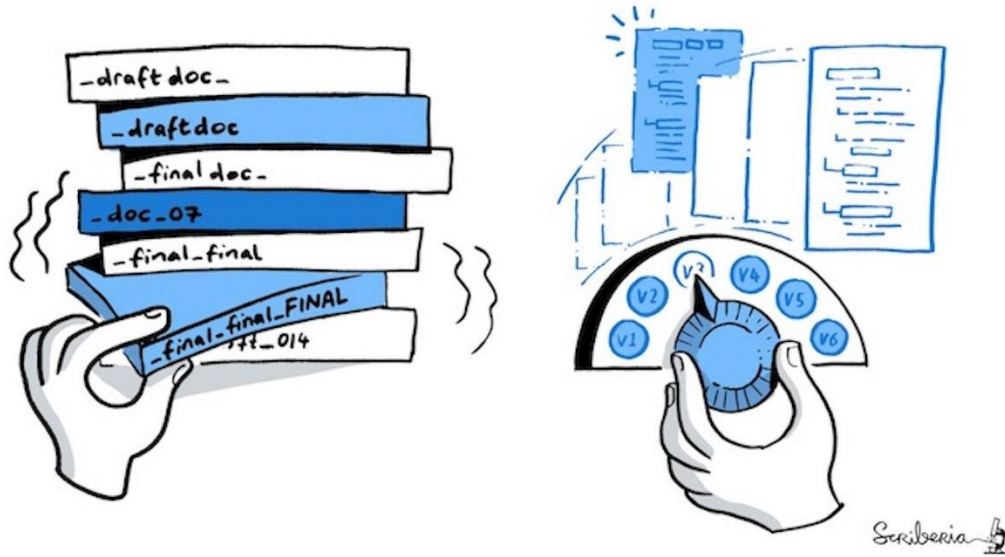
git





VKS nedir

TRACK PROJECT HISTORY

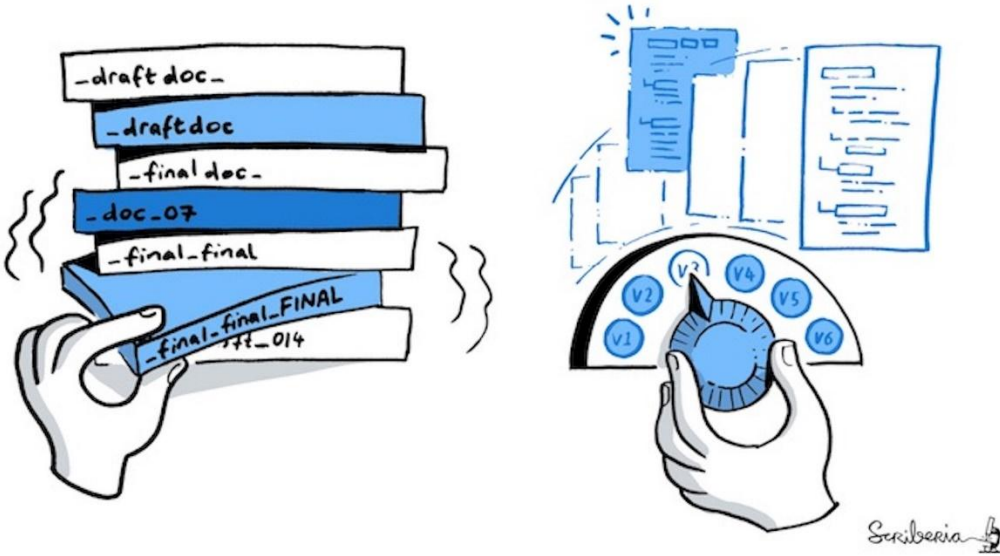


Versiyon kontrol sistemi, projede dosya ve klasör yapısındaki değişiklikleri kaydeden bir sistemdir.



VKS nedir

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 çeşitleri

3 tip Versiyon Kontrol Sistemi vardır.

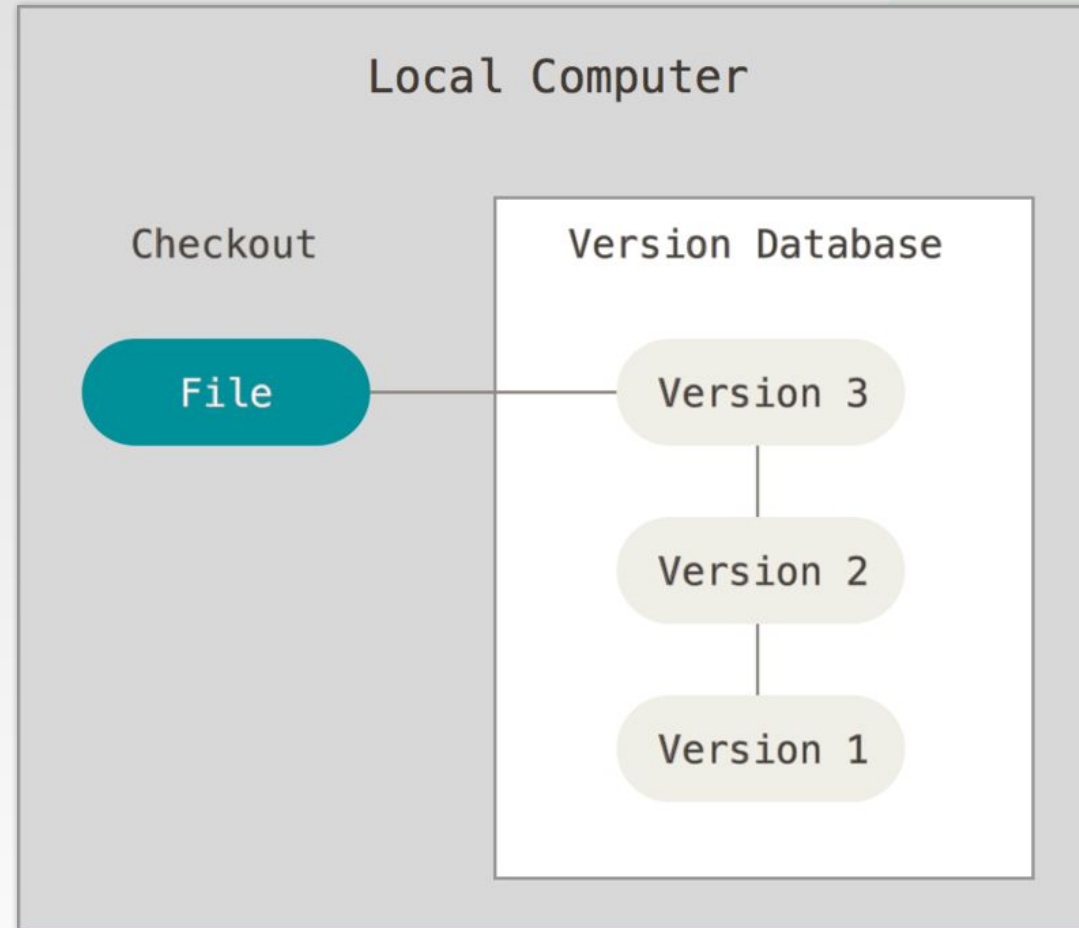
YEREL

MERKEZİ

DAĞITIK

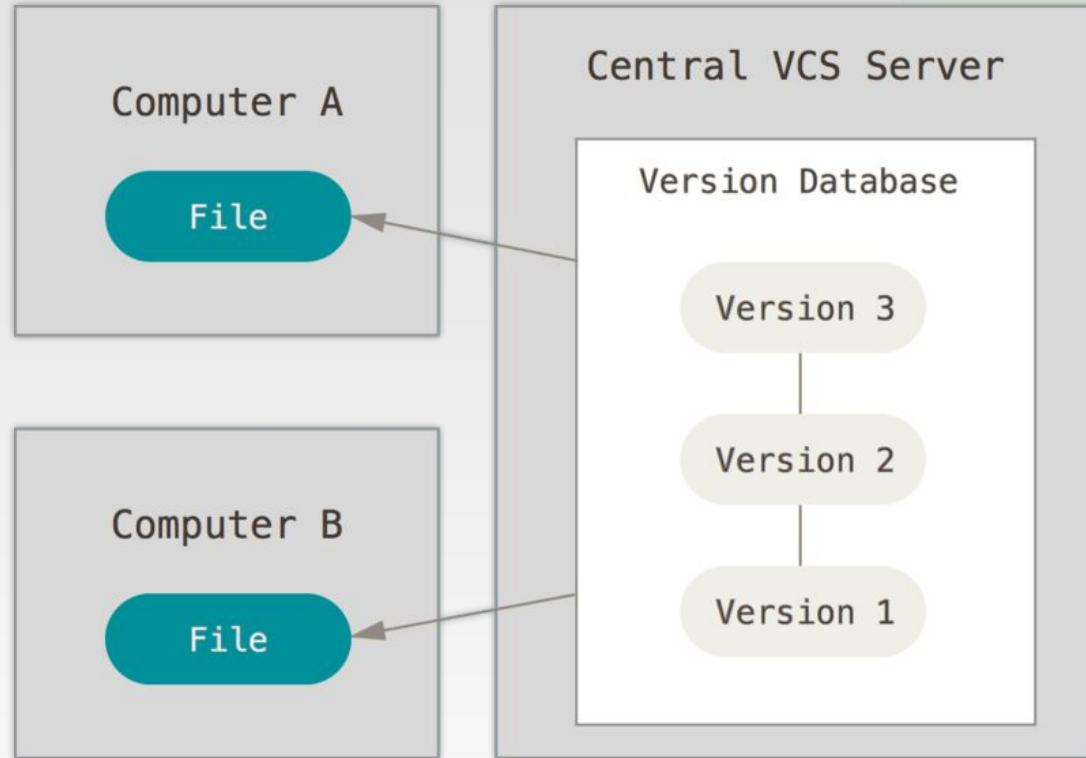


Yerel VKS nedir



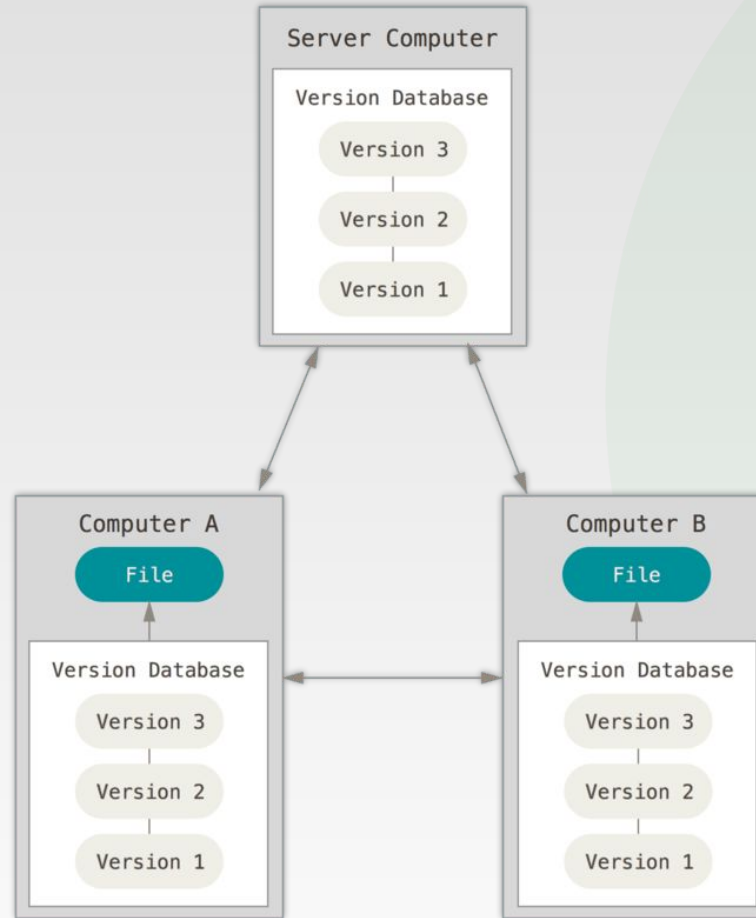


Merkezi VKS nedir





Dağıtık VKS nedir





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 (collaboration)



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



git

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"
```

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

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

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

Working
Space

git add

Staging
Area

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

Oluşturulan versiyonları görmek için bu komut kullanılır

git log --oneline

git commit

Commit
Store

```
git commit -m ''bir mesaj''  
git commit -amend -m ''yeni 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

LOCALE REPOSITORY

Commit Store

Commit 32

Commit 31

Commit 30

Commit 29

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

`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

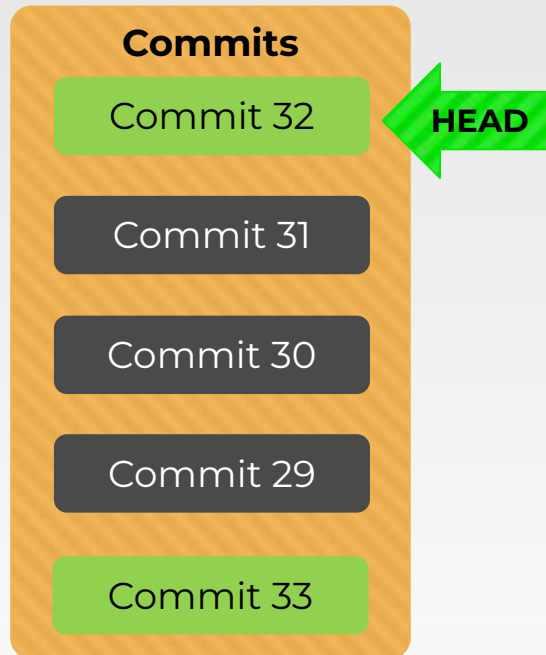
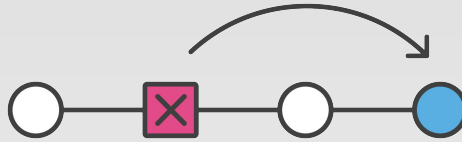
`git checkout [hash] .`

Hash değeri verilen versiyona döner



Önceki versiyonlara dönmek

1.Yöntem: CHECKOUT



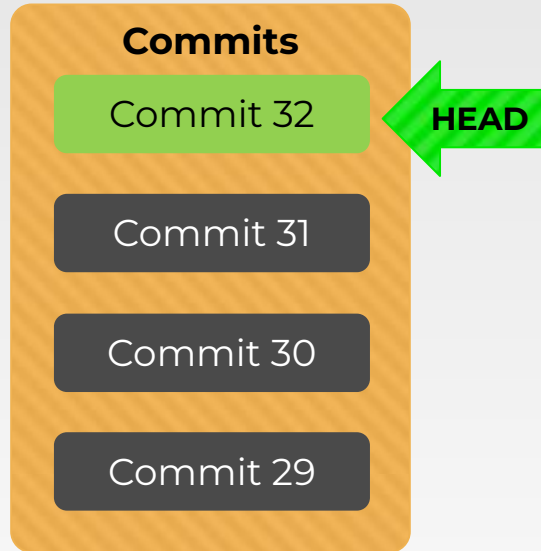
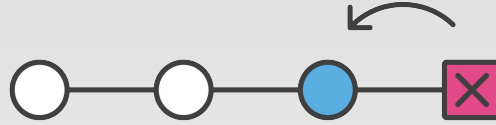
Önceki veriyonu incelemek için
git checkout *[hash]*.

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



Önceki versiyonlara dönmek

2.Yöntem:
RESET

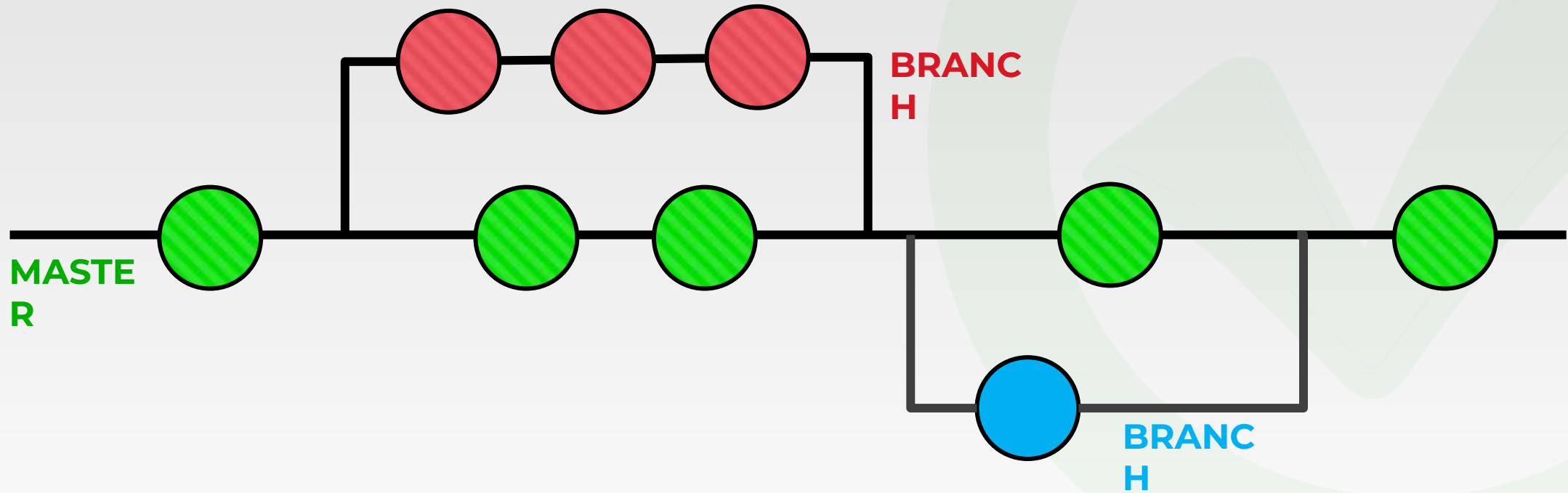


Geri alınamayacak şekilde önceki versiyona dönmek

git reset --hard *[hash]*

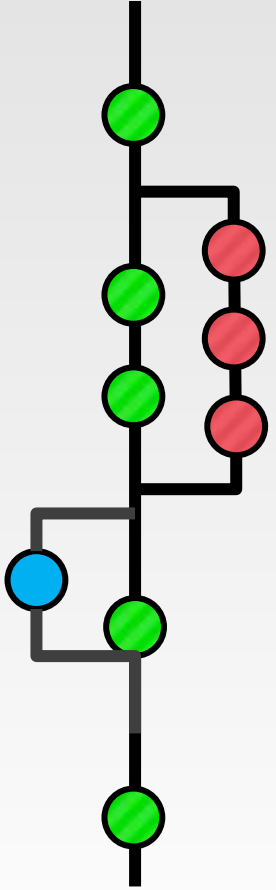


Branch (Dal)





Branch lerin faydaları



- Orijinal 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]`

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 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.

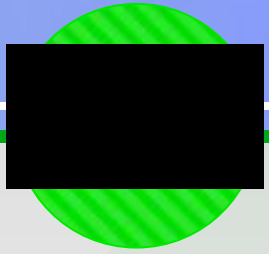


Git - Github

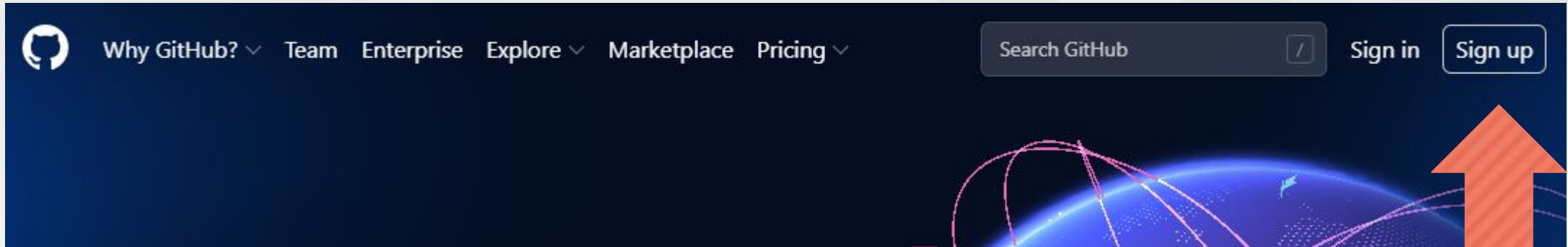
- Hesap oluşturma
- Repo oluşturma
- Genel kavramlar
- Github çalışma prensibi
- Clone
- Push & Pull
- Gitignore
- Merge & Conflicts



Github hesabı oluşturma



Github.co
m





Github hesabı oluşturma



Welcome to GitHub!
Let's begin the adventure

Enter your email

✓ techproed11@gmail.com

Create a password

✓

Enter a username

✓ techproed11

Would you like to receive product updates and announcements via email?

Type "y" for yes or "n" for no

✓ n

Eposta adresinizi giriniz

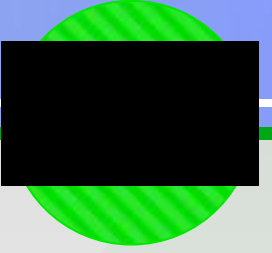
Şifre belirleyiniz

Bir kullanıcı adı belirliyoruz

Ürün güncelleştirmeleri ve tanıtımlardan email yoluyla haberdar olmak istemiyorsak n yazıyoruz



Github hesabı oluşturma



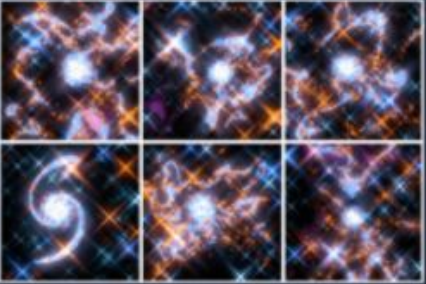
Verify your account

Doğrulama

Gerçek bir kişi olduğunuzu anlamamız için lütfen bu bulmacayı çözün

Doğrula

Sarmal galaksiyi seçin



Create account

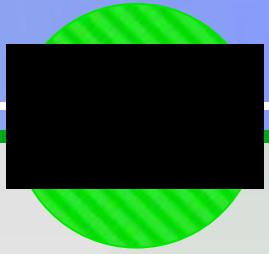
Doğrula butonuna basıyoruz

Doğrulama adımlarını geçiyoruz

Create Account butonuna basıyoruz



Github hesabı oluşturma



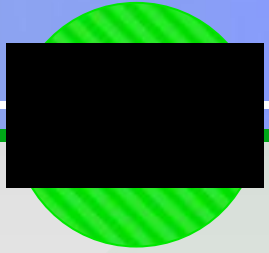
You're almost done!
We sent a launch code to `techproed11@gmail.com`

→ Enter code

Eposta adresine gönderilen
kodu girerek işlemi
tamamlıyoruz



Github repo oluşturma



Pull requests Issues Marketplace Explore

Overview Repositories 15 Projects Packages

Find a repository... Type Language Sort New

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

techproeducation1 /

Great repository names are short and memorable. Need inspiration? How about [scaling-meme?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Create repository

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

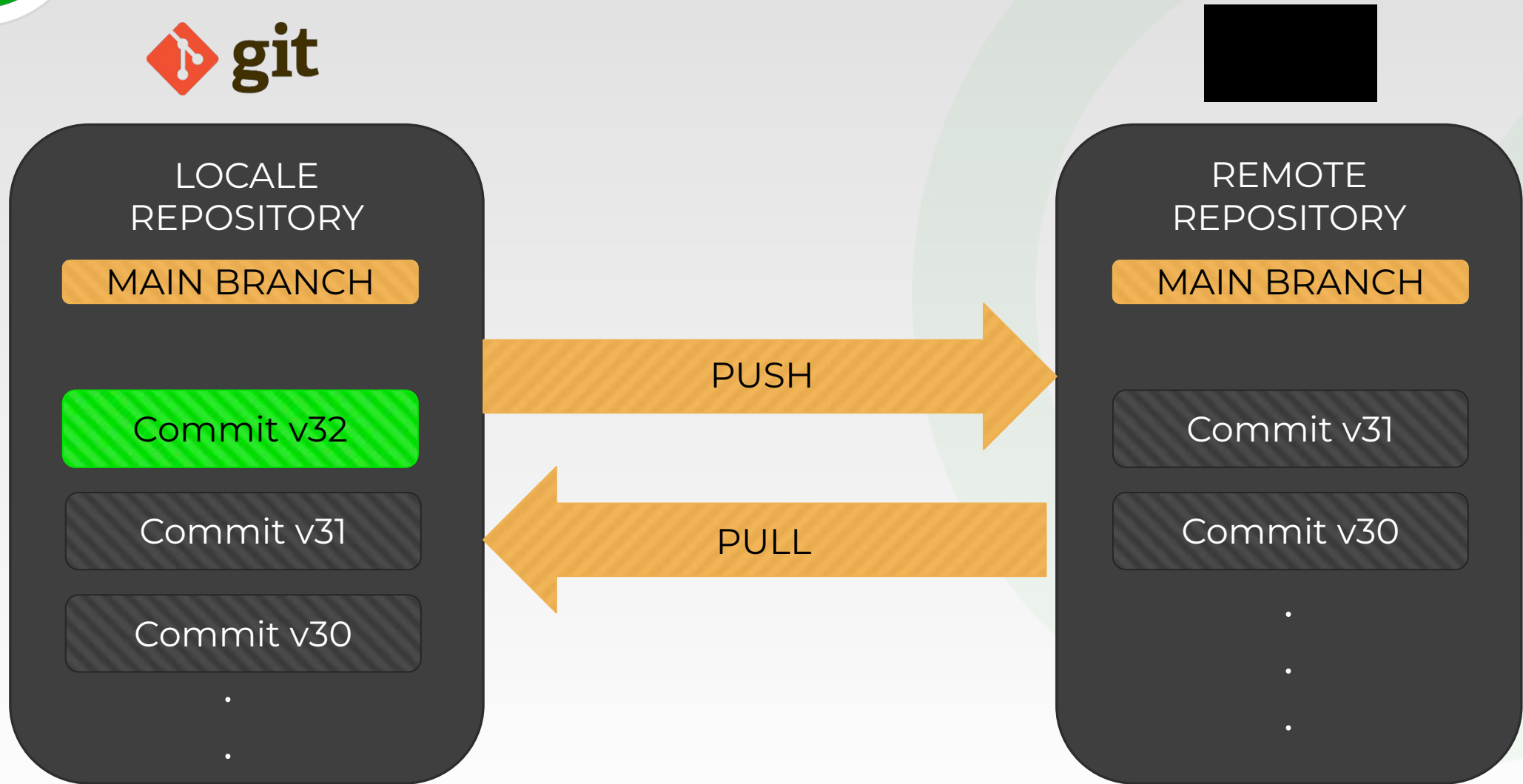
Lokalde oluşturulan commit lerin github a gönderilmesi işlemidir.

Pull

Fetch ve Merge işlemini tek başına yapar



Github Çalışma Prensipleri





Cloning

| **git clone** |

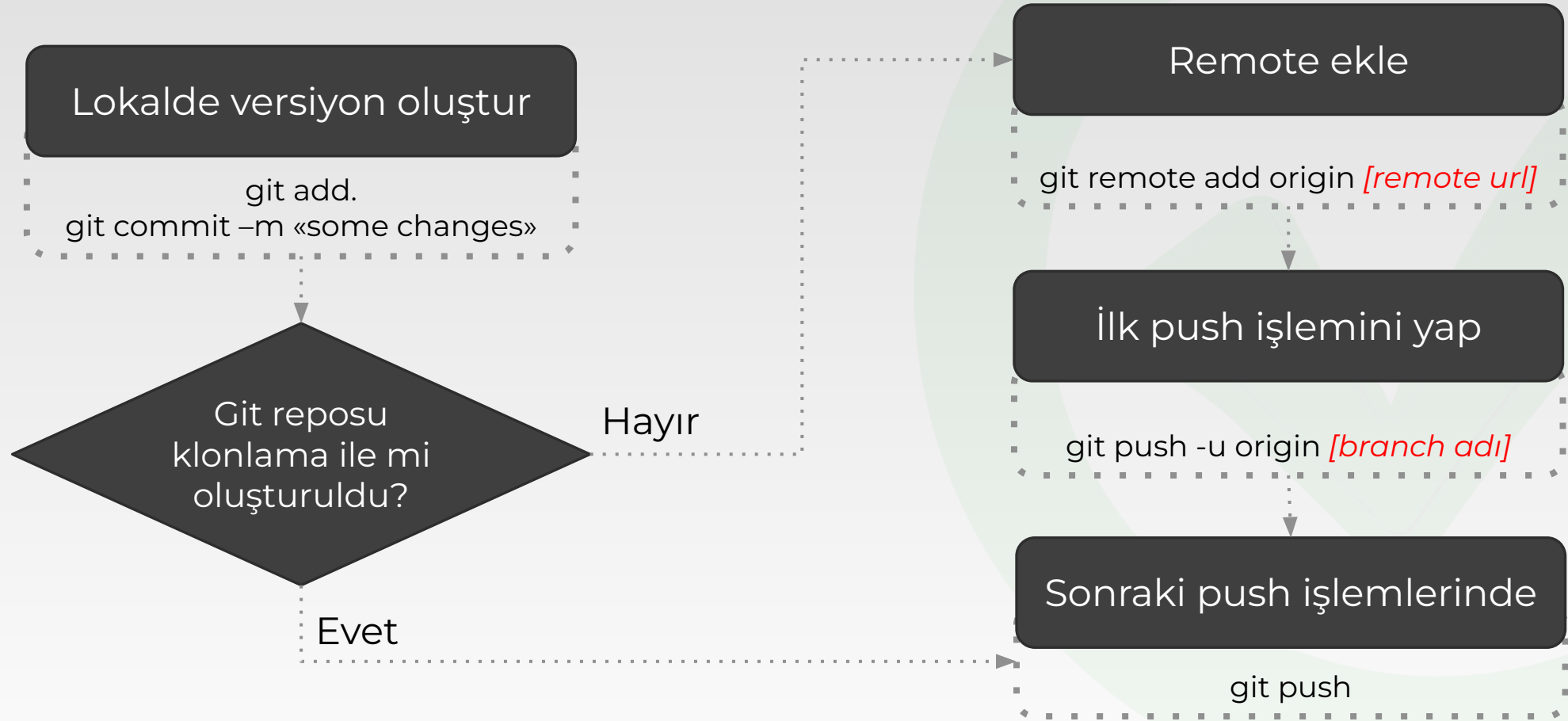
Github daki bir reponun lokale indirilmesi işlemine klonlama denir. Public olan veya gerekli izinlere sahip olunan private repolar klonlanabilir.

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

**VEY
A**

`git pull`

fetch & merge



.gitignore

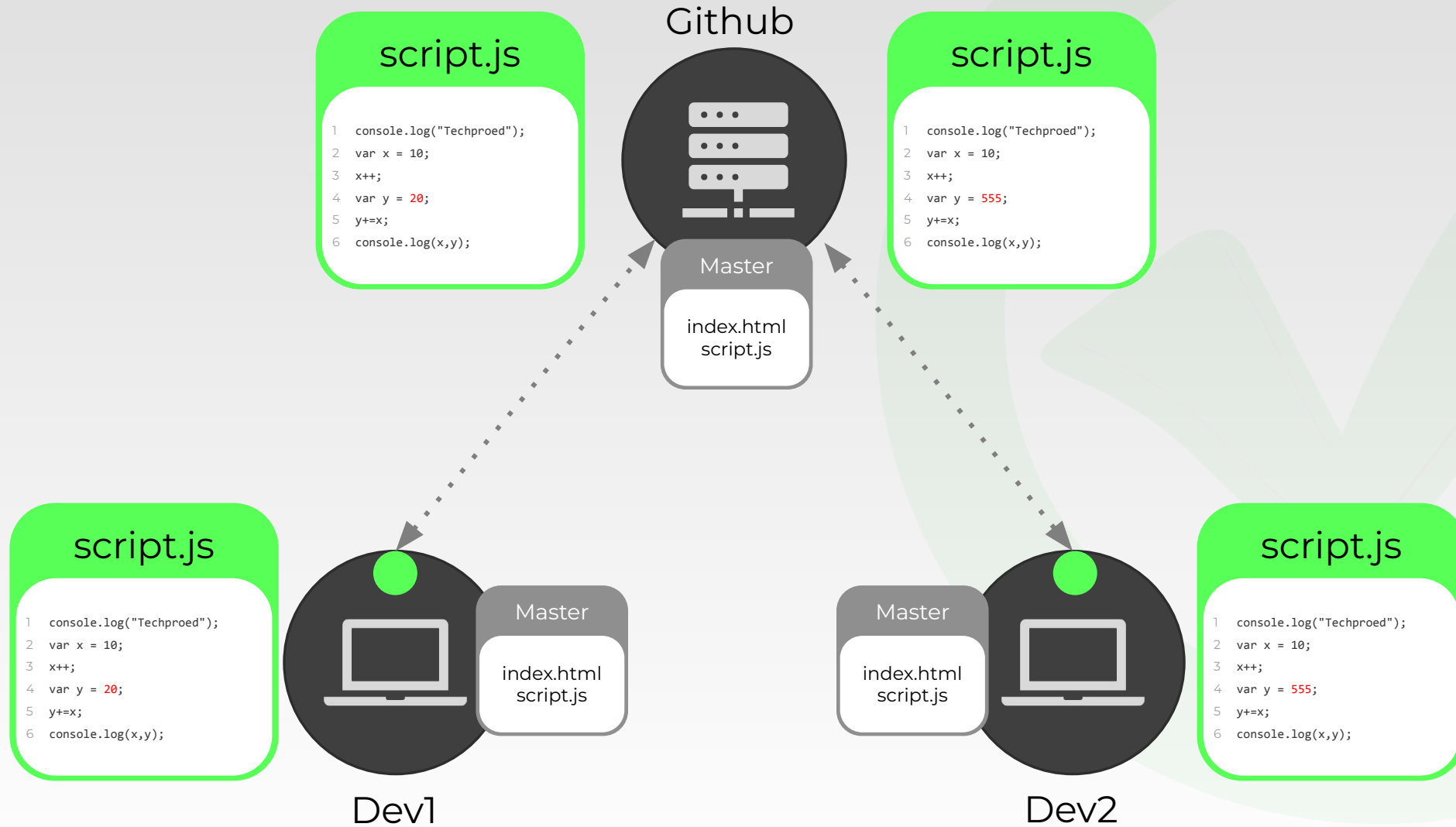
Staging area'ya gitmesini istemediğimiz, yani versiyon kontrol sistemine dahil etmek istemediğimiz dosya ve klasörlerimizi tanımladığımız özel bir dosyadır.

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

.gitignor
e



Merge Conflict





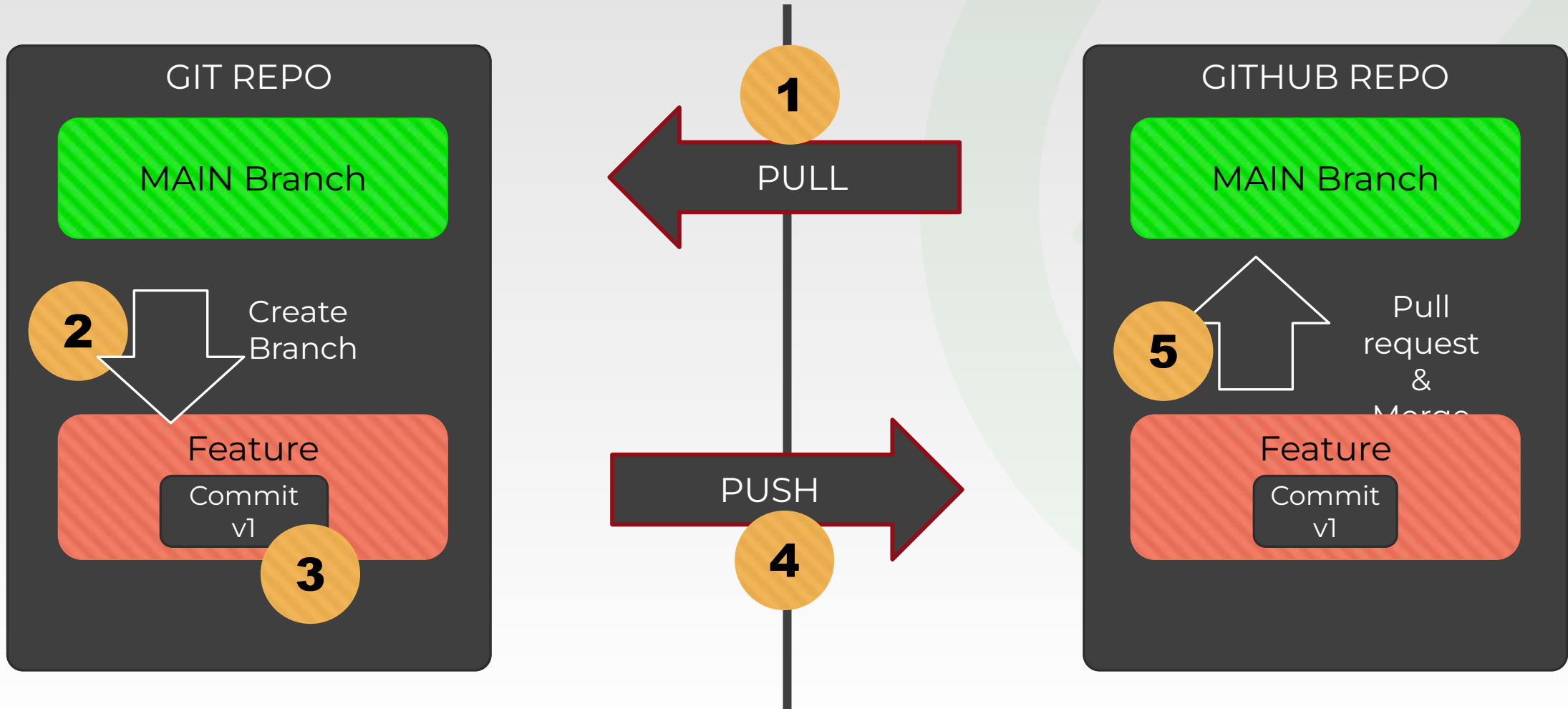
```
Auto-merging script.js
CONFLICT (content): Merge conflict in script.js
Automatic merge failed; fix conflicts and then
commit result.
```

[illegible]



Git – Github Çalışma döngüsü

**BEST
PRACTISE**





**Visual Studio Code ile Git – Github
Entegrasyonu**



VS Code & Git-Github

VS Code' da Git in Aktif Hale Getirilmesi

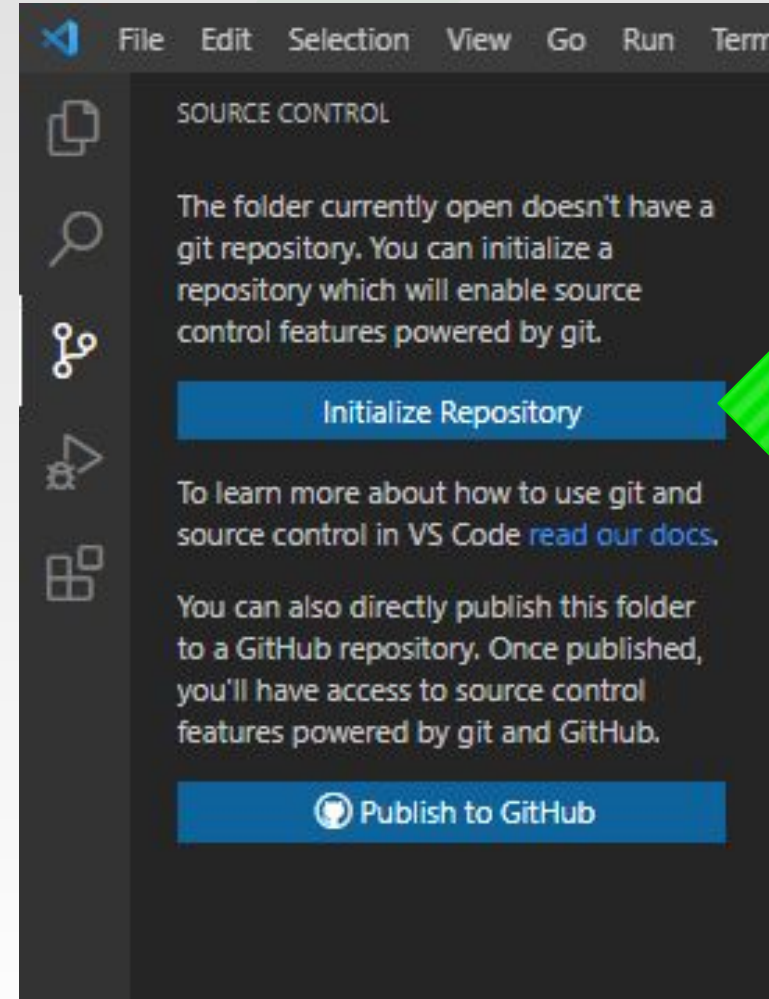
- 1- Ayarlar simgesine tıklanır
- 2- Settings seçilir
- 3- Arama bölümüne «git enabled» yazılır
- 4- Alt tarafa gelen ayarlardaki Git Enabled kutusu işaretlenir.

The image shows two screenshots of the VS Code interface. The first screenshot shows the Command Palette with 'Settings' selected. The second screenshot shows the Settings window with the search bar containing 'git enabled' and the 'Git: Enabled' checkbox checked. Red circles with numbers 1 through 4 and green arrows indicate the sequence of steps.



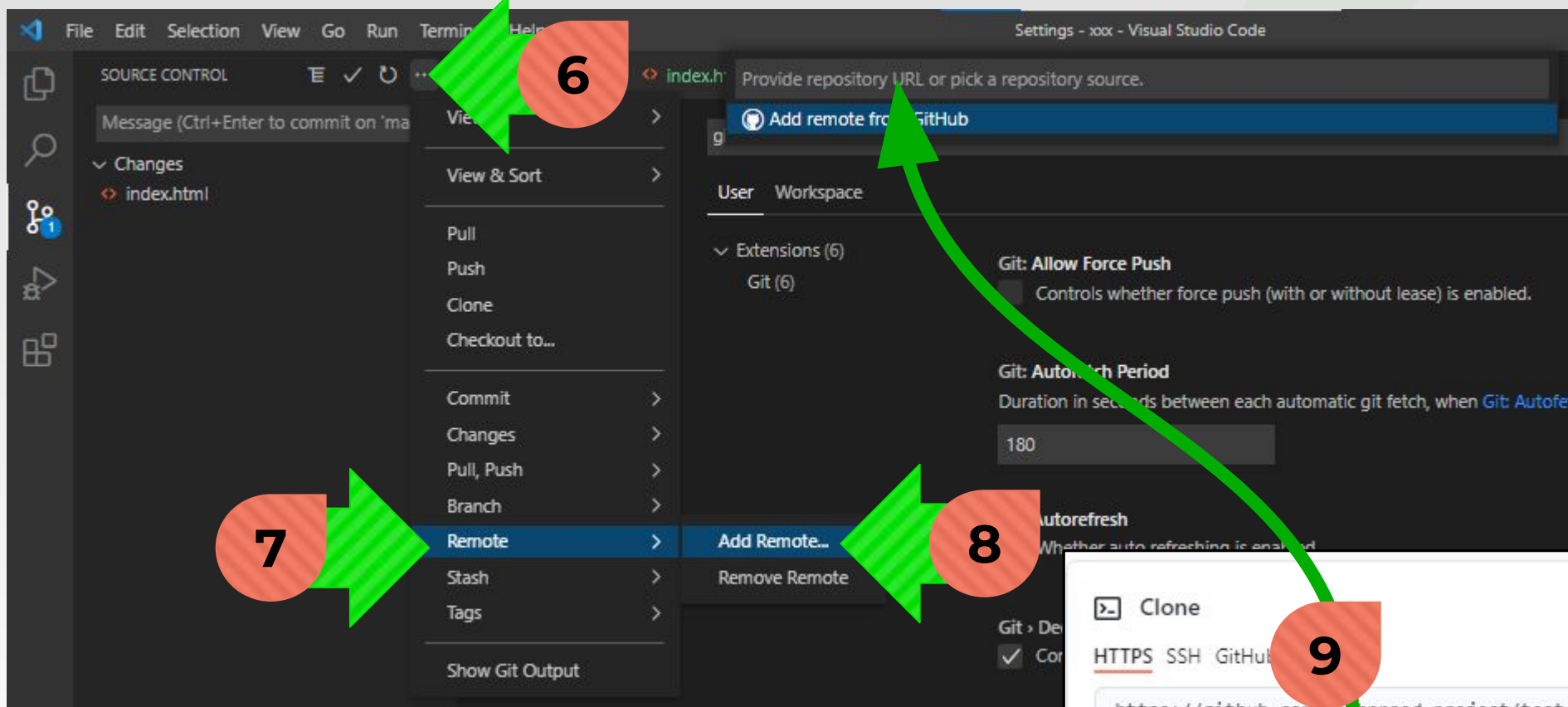
VS Code & Git-Github

5- Bir proje VS Code ile açıldıktan sonra Initialize Repository ile repository oluşturulur. Burada VS Code **git init** komutunu çalıştırır. Bu aşamada local version kontrol sistemi kurulmuş olur.



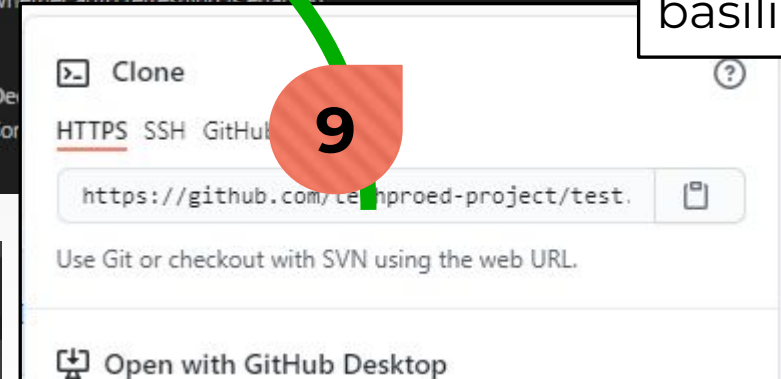
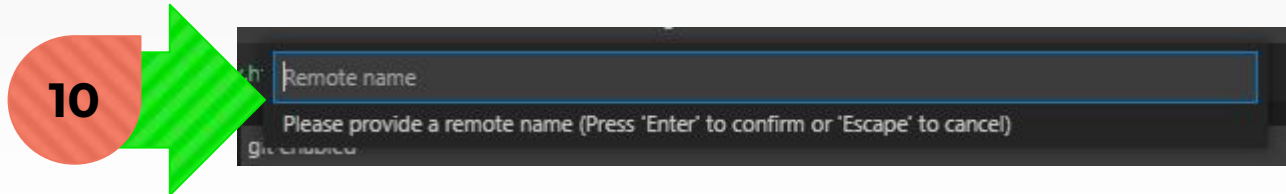


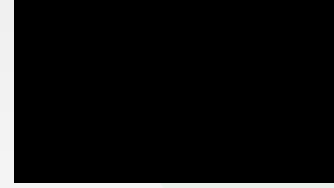
VS Code & Git-Github



Github bağlantısı

6- Üç noktaya basılarak menü açılır
7- Remote seçilir
8- Add remote seçilir.
9- Açılan kutuya github hesabından alınan repo url adresi yapıştırılır
10- Reponun ismi yazılır ve enter a basılır





**IntelliJ ile Git – Github
Entegrasyonu**

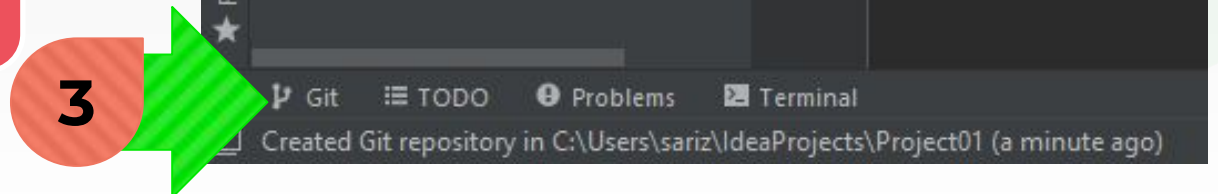
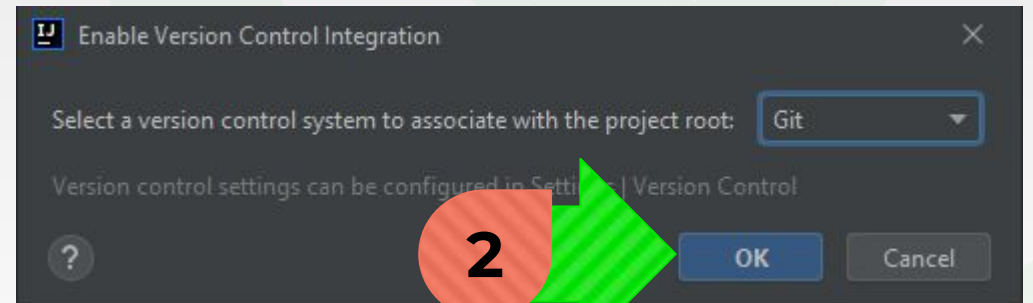
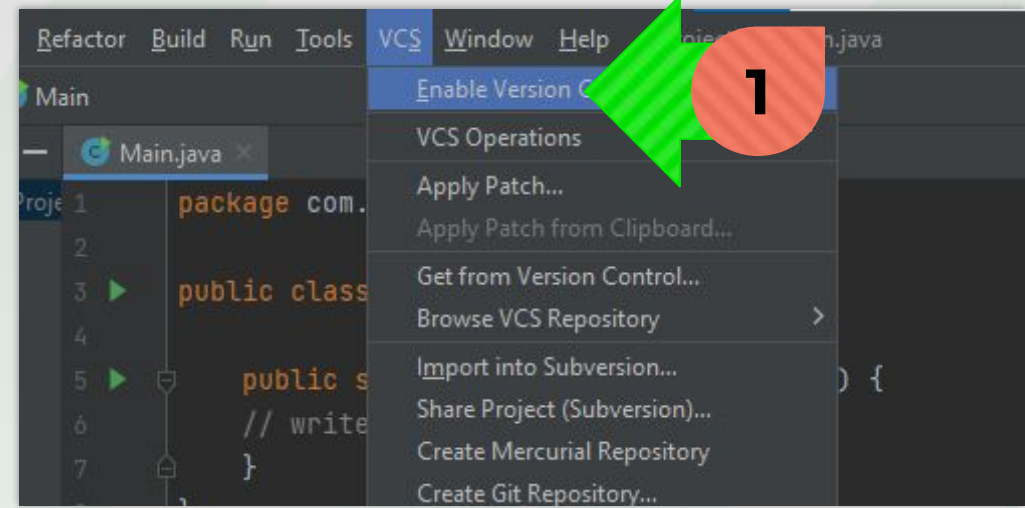


Intellj ile Git Github

- 1- IntelliJ üzerinde VCS kullanmak için VCS menüsündeki «Enable Version Control Integration» komutu seçilir.
- 2- Açılan pencerede Git seçilerek OK butonuna basılır
- 3- Böylece sol alt tarafta Git sekmesi ve üst tarafta Git menüsü görülecektir. (Bu aşamada git init çalıştırılmış oldu)



IntelliJ, üzerinde git ile ilgili oluşabilecek sorunlarda File/Invalidate Caches komutu kullanılabilir





Intellj ile Git Github

Github bağlantısı

4- Git menüsüne tıklanır

5- Manage Remotes seçilir

6- Açılan pencerede + butonuna basılır

7- Github adresi yapıştırılır ve OK butonuna basılır

Ardından github hesabı için authorization işlemleri başlayacaktır.

