BATCH : BATCH 48

LESSON : GIT

DATE : 19.01.2022

SUBJECT : GIT BASICS

techproeducation

techproeducation

techproeducation

techproeducation

techproedu

techproeducation.com info@techproeducation.com +1 (917) 768-7466

Command Prompt Terminal

- Command prompt veya terminal için komutlar

Komutlar

	Windows	Linux/Mac
Uygulama adı	**Komut İstemi**	**Bash/Terminal**
Konum değiştirme	cd | cd.. | cd/	
Listeleme	dir	Ls
Klasör oluşturma	Mkdir	
Klasör silme	Rmdir	
Dosya oluşturma	echo merhaba > dosya.txt	cat > merhaba.txt
Dosyanın içini görme	more	cat
Dosya silme	del	rm
Klasör ve dosya ismi değiştirme	ren	mv
Ekran temizleme	cls	clear

Giriş

- Versiyon kontrol sistemi

LOCALE

- Git ile yönetilen repoların public veya private olarak **saklandığı** veya **paylaşıldığı** uzak sunucu
- Birden fazla kişi ile işbirliği içinde çalışma imkanı

REMOTE

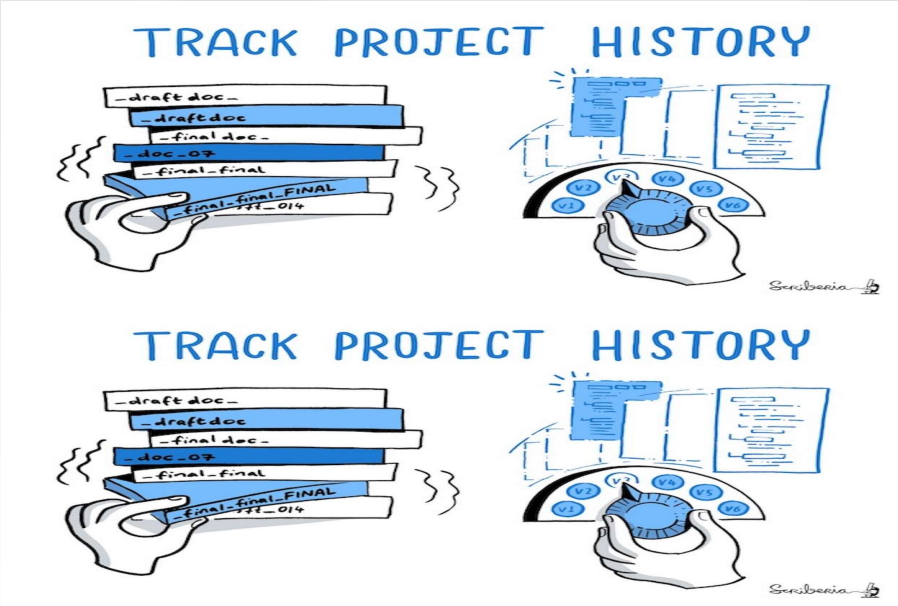
Version Control System

Versiyon Kontrol Sistemi

- VKS, bir uygulamada belli değişikliklerden sonra, o ana kadar ortaya çıkan ürün ile iligli bir versiyon oluşturulması, yeni değişikliklerin ayrı bir versiyona konulması işlemidir.
- Çoğu insanın versiyon kontrol metodu, ilgili dosyaları başka bir yere kopyalamaktır.
- Bu yaklaşım basit olduğundan çok yaygındır fakat aynı zamanda inanılmaz derecede hataya açık bir yaklaşımdır.
- Hangi dizinde bulunduğunuzu unutmak, yanlışlıkla yanlış dosya üzerine yazmak veya istemediğiniz dosyaların üzerine yazmak gibi ihtimallerin gerçekleşmesi çok olasıdır.

VKS nedir

Versiyon kontrol sistemi, belirli versiyonların daha sonra çağrılabilmesi için zaman içerisinde bir dosya veya dosya grubundaki değişiklikleri kaydeden bir sistemdir.

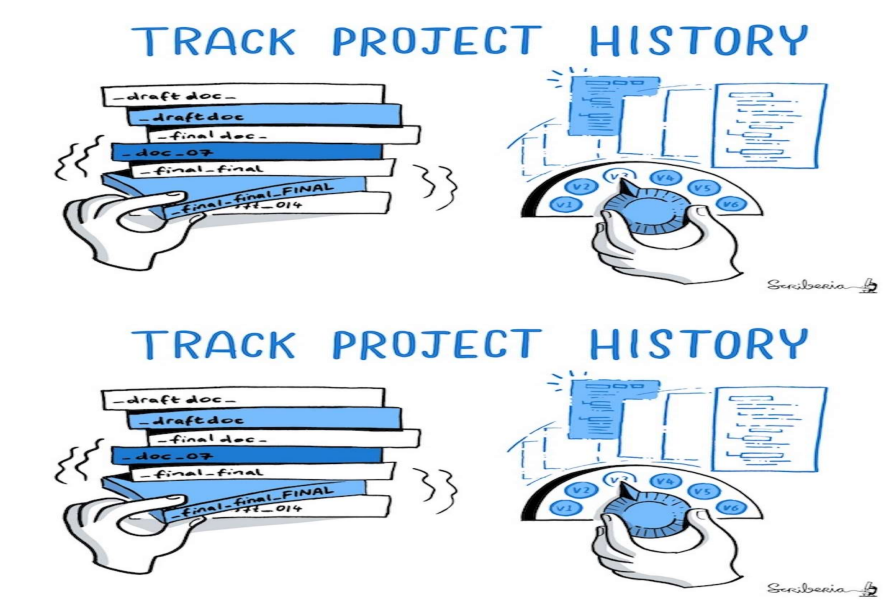


https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

VKS nedir

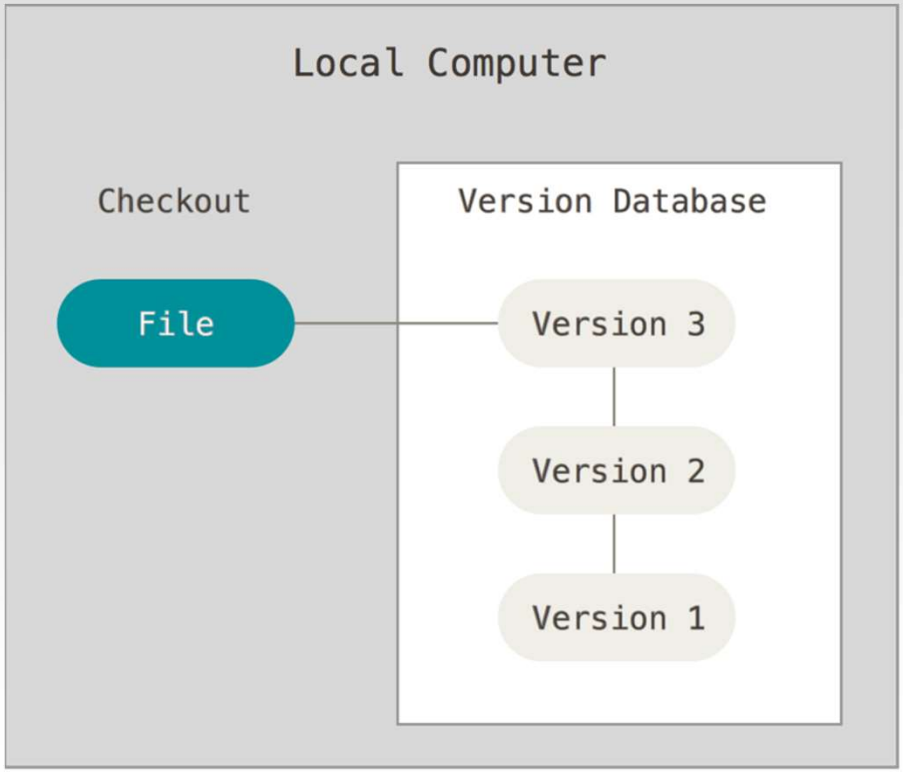
Versiyon Kontrol Sistemi, seçili dosyaların bir önceki versiyona döndürülmesi, projenin tamamının bir önceki versiyona döndürülmesi, zaman içerisinde yapılan değişikliklerin karşılaştırılması, probleme neden olabilecek değişikliklerin en son kimin tarafından yapıldığı gibi bir çok işlemin gerçekleştirilebilmesini sağlar.

Genel olarak VKS kullanmak, değişiklik yaptığınız dosyalar üzerinde bir şeyleri berbat ettiğinizde ya da bir şeyleri kaybettiğinizde kolayca geri getirebilmeniz anlamına gelmektedir.



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

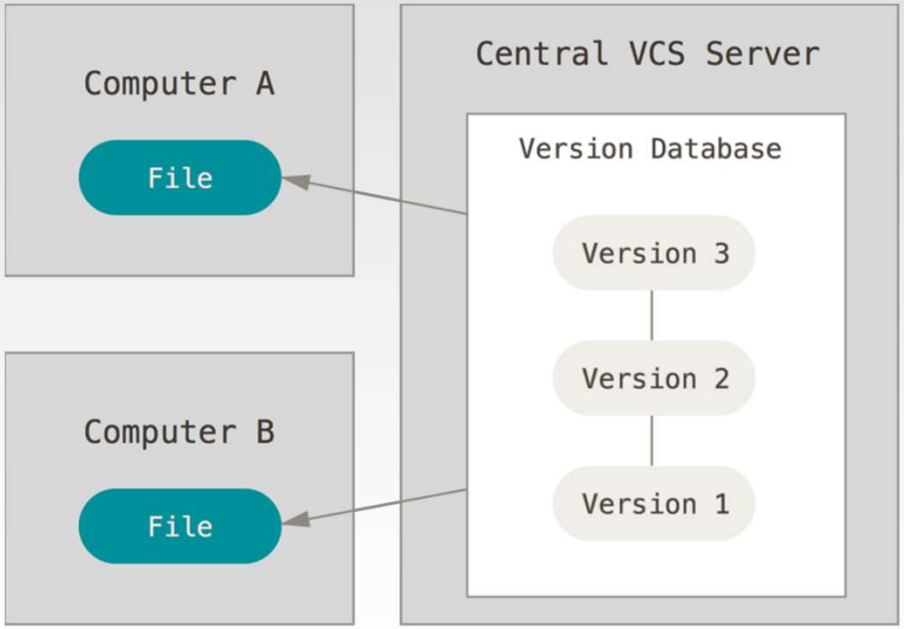
Yerel VKS nedir



- YVKS, versiyon kontrol sisteminin lokal bilgisayarda tutulduğu sistemlerdir.
- Bu sistemde geliştirici kendi lokal bilgisayarında uygulama ile ilgili versiyon sistemi kullanabilir **ancak farklı developer'lar ile çalışmak isterse YVKS sistemi bunun için bir çözüm üretemez**.

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

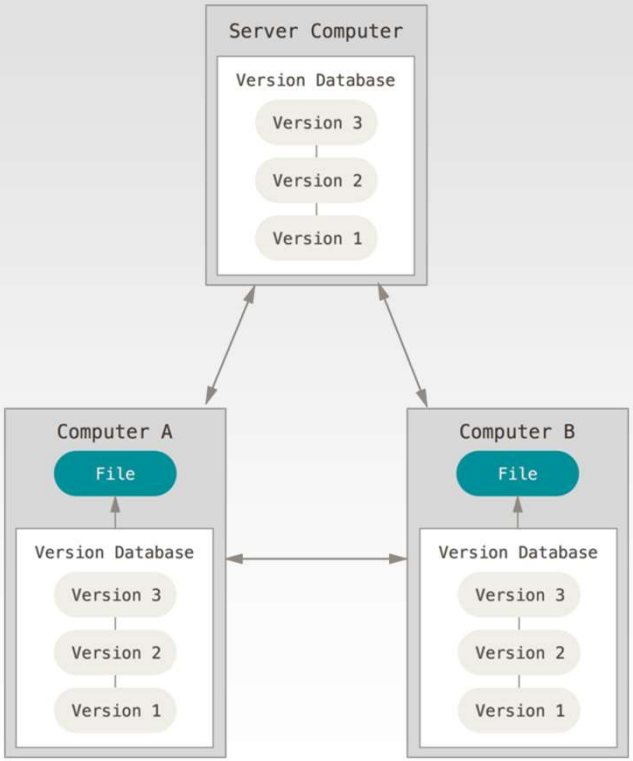
Merkezi VKS nedir



- Bu sistemde versiyonların depolanması ve kontrolü uzaktaki bir sunucu üzerinden yapılmaktadır. ***Lokal cihazlarda herhangi bir depolama ve kontrol yapılmaz***.
- Bu sistemin en büyük **sorunu o sunucuda bir sorun oluştuğu andan itibaren hiç kimse iş yapamaz veya üzerinde çalışmakta oldukları herhangi bir şeye sürüm değişikliklerini kaydedemezler**.

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

Dağıtık(Distributed) VKS nedir



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

- İşte tam da burada devreye Dağıtık(Distributed) Versiyon Kontrol Sistemleri (DVKS) giriyor. Bir **DVKS' de hem merkezi bir sunucu bulunmaktadır, hem de client'larda da aynı yapının bir kopyası bulunmaktadır.**

- Dolayısıyla eğer bir sunucu devre dışı kalırsa, client'larda da aynı yapı bulunduğundan sunucu devreye girene kadar her bir geliştirici lokalde çalışabilirken, sunucu devreye alındığında client'lar tarafından sunucu rahatlıkla güncelleyebilir. **Her client, en nihayetinde tüm verilerin tam bir yedeğidir** aslında.

Kurulum ve İlk Ayarlar

- Git Kurulum
- Linux İçin:

```
sudo apt update
sudo apt install git -y
```

- Git altyapısını oluşturmak ve git komutlarını kullanabilmek için Git kütüphanesinin kurulması gerekmektedir
  [https://git-scm.com/downloads]

  git –version // git in kurulup kurulmadığını anlamak için

Kurulum ve İlk Ayarlar

- Git configuration

  git config --global color.ui true

  git config –global user.name "Ali Gel"

  git config –global user.email "ali@gel.com"

Terminal de komutların renklendirilmesini sağlar

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

- 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

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

Local repo oluşturma

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

git init

Genel Kavramlar

REPOSITORY

Working Space

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

TS1

TS0

Staging Area

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

TS2

TS3

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.

TS4

Slide 16

TS0 git add burada uygulanır

* git add . (Her şeyi ekleyeyim diyorsan)

* git add spesifikdosya.txt (Spesifik bir dosyayı ekleyeyim diyorsan)
Tevfik SASTIM, 2022-01-15T09:30:58.291

TS1 git init burada uygulanır.
Tevfik SASTIM, 2022-01-15T09:32:12.619

TS2 Şimdi buranın sonunda commit'i yapanın kimliğini belirlemek için git config kullanılabilir:

* git config user.email sarah@example.com

* git config user.name sarah
Tevfik SASTIM, 2022-01-15T09:37:25.334

TS2 0 Not devam ediyor çift tıkla
Tevfik SASTIM, 2022-01-15T09:41:58.663

TS3 Commit burada yapılır, ancak öncesinde son durumu kontrol etmek için git status'u kullanmak evladır.

* git status

* git commit -m "Commit'e ekleyeceğin kısa mesaj"
Tevfik SASTIM, 2022-01-15T09:40:01.692

TS3 0 Not devam ediyor çift tıkla
Tevfik SASTIM, 2022-01-15T09:42:11.802

TS3 1 othing added to commit but untracked files present (use "git add" to track) diyorsa git add yapmalıs
Tevfik SASTIM, 2022-01-15T12:46:50.821

TS4 Her şeyi bitirdik, directorymizde ne var ne yok bilgi sahibi olmak istiyoruz, git log'u kullanırız.

* git log
Tevfik SASTIM, 2022-01-15T10:03:56.825

Local versiyonlar oluşturma

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

git status

Oluşturulan versiyonları görmek için bu komut kullanılır

git log

Working Space

Git add iki farklı şekilde kullanılır.
*1- Belli bir dosyayı **staging area**'ya göndermek için git add komutundan sonra dosyanın ismi yazılır.*
2- Değişiklik yapılan tüm dosyaları stage a göndermek için nokta konulur.

Değişikliklerin Stage' e gönderilmesi

git add

git add *dosya_adi*
git add .

Versiyon oluşturma

git commit

git commit –m "ilk versiyon«
git commit ile belli bir dosyayı **staging area'dan commit store'a yollarız**
Değişiklikleri **local repo**'da saklama işlemine commit diyoruz.

TS0

TS0 Sadece commit'lemek için:

* git commit -m "Kısa mesaj"

Tevfik SASTIM, 2022-01-15T09:55:01.120

Versiyon detaylarını görme

```
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
```

- Bir versiyonda hangi değişikliklerin olduğunu görmek için öncelikle **git log** komutu kullanılarak ilgili **commit in hash kodu öğren**ilir. Ardından aşağıdaki komut kullanılarak detaylara ulaşılır
- git show'dan çıkmak için 'q' ya basılır.
- Bütün çıktıyı tek bir satırda almak istiyorsan: **git log --oneline**

git show *[hash kodun ilk 7 karakteri]*

Versiyon oluşturmak için kodlar

Ana komutlar

```
git init
(Repoda değişiklik yapılır)
git add .
git commit –m "versiyon metni"
```

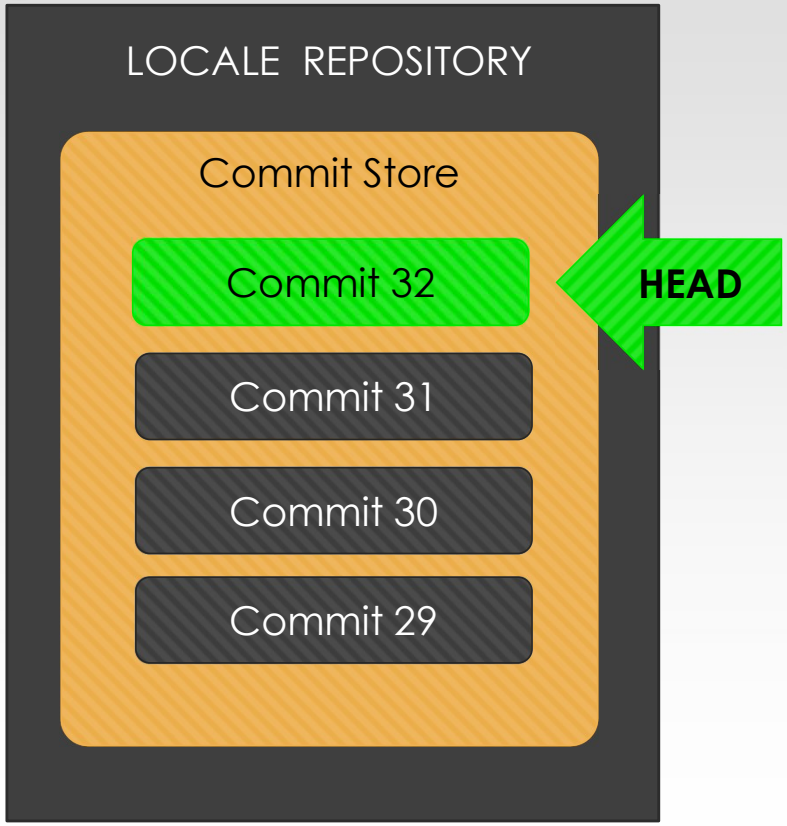
- `git init` → Repo oluşturur. Her projede en başta bir kere kullanılır.
- `git add .` → Dosyaları staging area ya gönderir
- `git commit –m "versiyon metni"` → Versiyon oluşturur

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

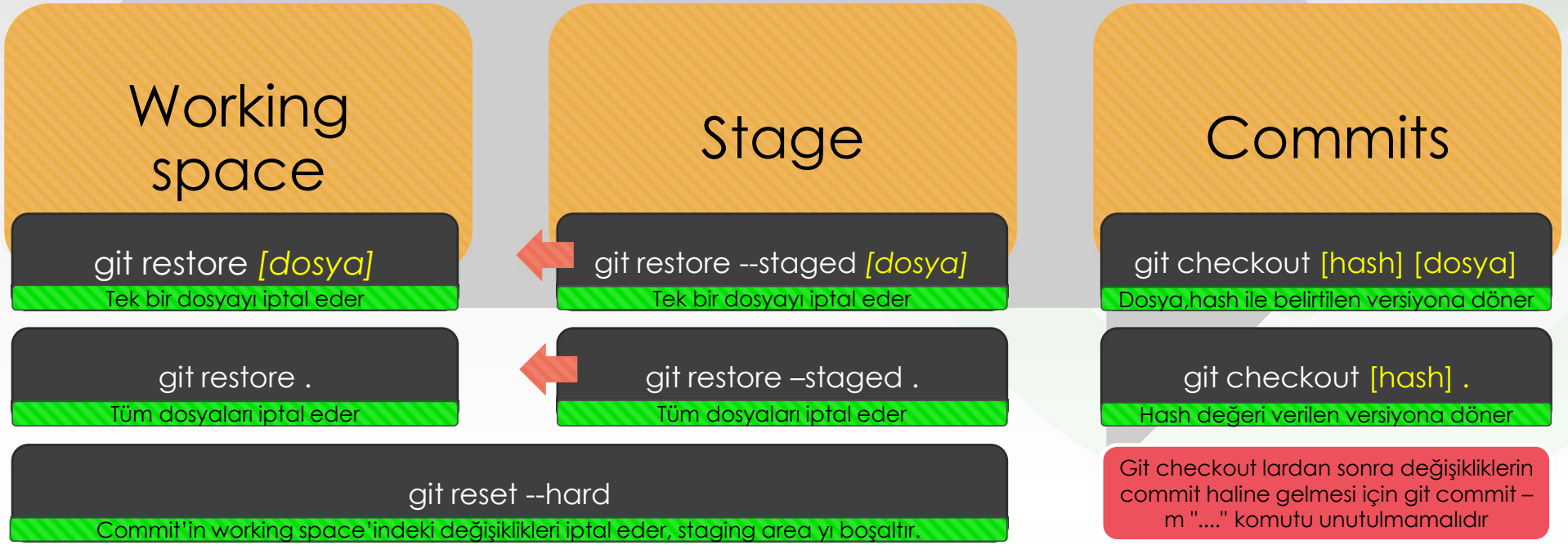


- Bir repo içinde birden fazla commit olabilir. Bunlardan en son alınan commit' e **HEAD** denir.
- Bu HEAD değiştirildiğinde önceki versiyonlara dönüş yapılabilir.

```
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
```

Değişiklikleri iptal etmek

TS0



TS1

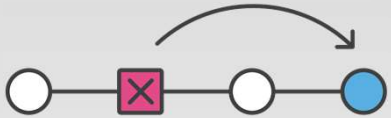
Slide 21

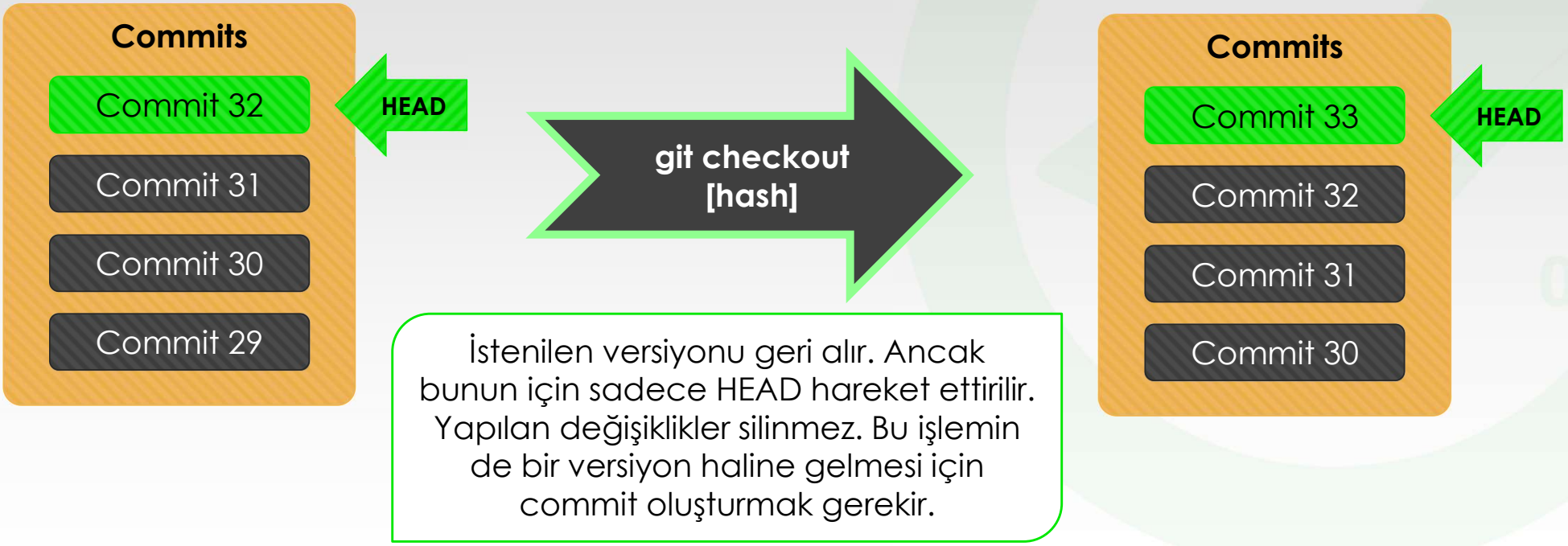
TS0 git revert <hashcode'un ilk 5 hanesi>
* hash code 'unu yazdığın commit'teki değişiklikleri geri alır ancak değişiklikleri git history'inde tutmaya devam eder.
* git revert ile varolan mesajı da güncelleyebilirsin
* İşlemin bitince esc + ctrl + z + z ye basmayı unutma
Tevfik SASTIM, 2022-01-15T14:48:32.704

TS1 Her bir iptal işleminin sonrasında git status deyip iptal edilip edilmediğini onaylayabilirsin
Tevfik SASTIM, 2022-01-15T14:52:18.260

Önceki versiyonlara dönmek

1.Yöntem: CHECKOUT





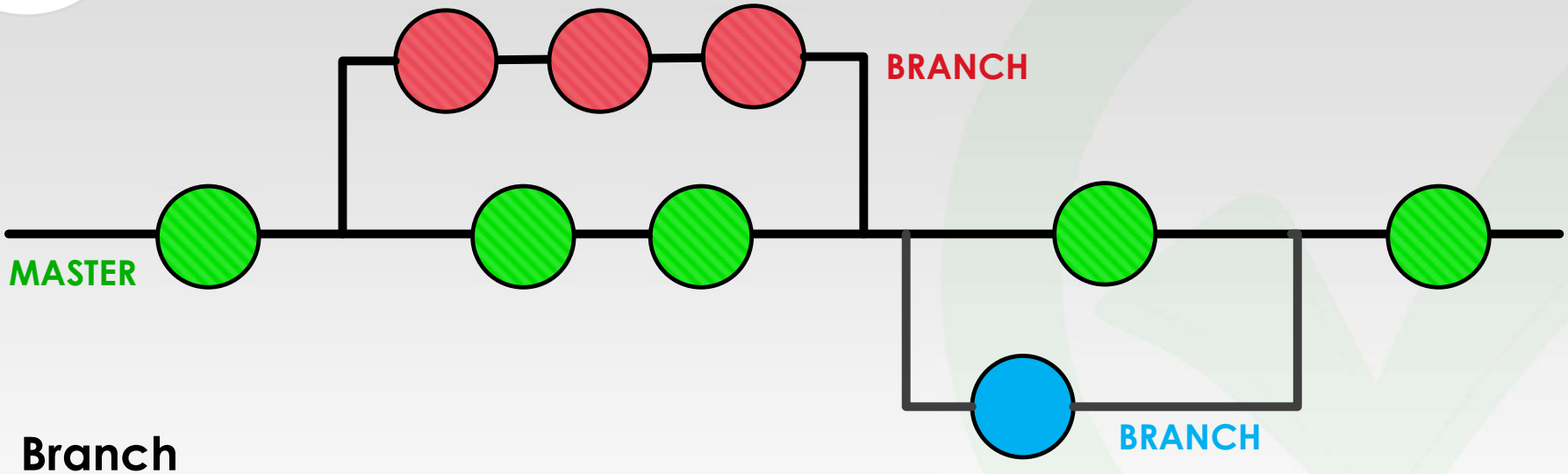
İstenilen versiyonu geri alır. Ancak bunun için sadece HEAD hareket ettirilir. Yapılan değişiklikler silinmez. Bu işlemin de bir versiyon haline gelmesi için commit oluşturmak gerekir.

Önceki versiyonlara dönmek

2.Yöntem: RESET

İstenilen versiyona geri döner, bu versiyondan daha sonra yapılan tüm commit ler ve içerdiği değişiklikler geri alınamayacak şekilde iptal edilir.

Branch (Dal)



Branch

Master branch, projemizin ana yapısıdır. Zaman zaman bu ana yapıyı bozmadan bazı denemeler yapmak ve gerekirse kolaylıkla bu denemeleri iptal etmek ya da master ile birleştirmek için branch ler kullanılır. Branch ler içindeki değişiklikler master dan bağımsız olarak saklanır.

Branch Komutları

git branch *[isim]*

Yeni branch oluşturur

git branch –d *[isim]*

Branch i siler

git branch -m *[isim]*

Branch ismini değiştirir
Hangi branch'taysanız onun ismini değiştirir.

git branch

Mevcut branch leri listeler

git checkout *[isim]*

Branch aktif hale gelir
Çalışmak istediğin branch'ı seçersin

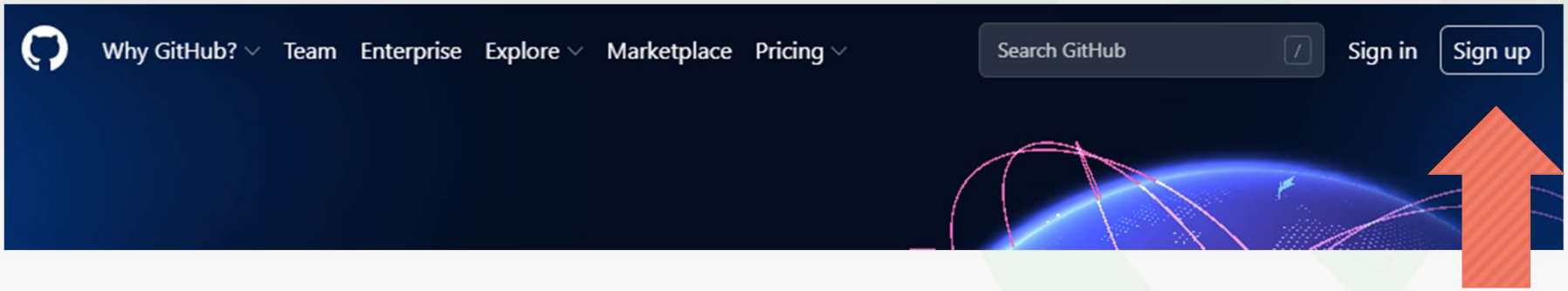
git branch -a

Local ve remote daki bütün branch'leri görmek için kullanırız

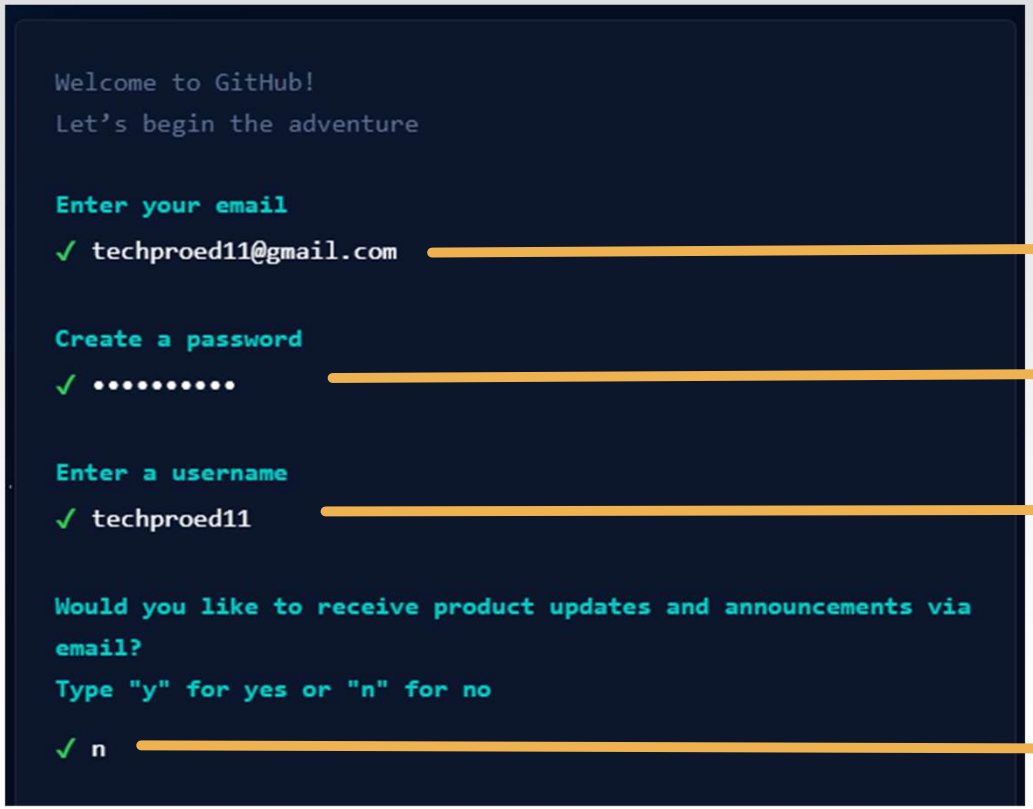

GitHub

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

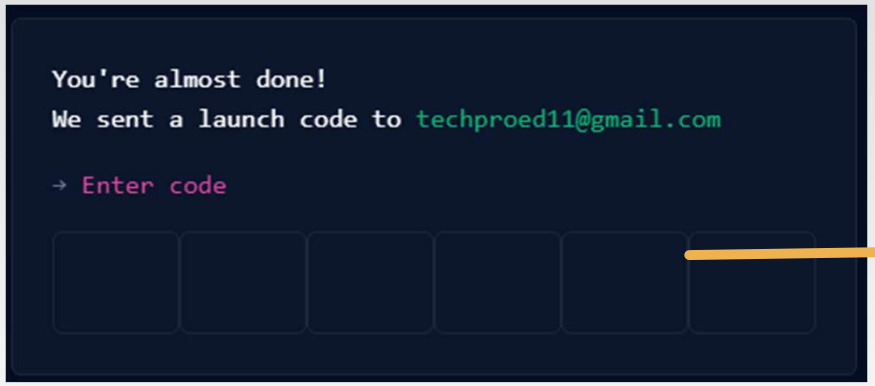
Github hesabı oluşturma

Doğrula butonuna basıyoruz

Doğrulama adımlarını geçiyoruz

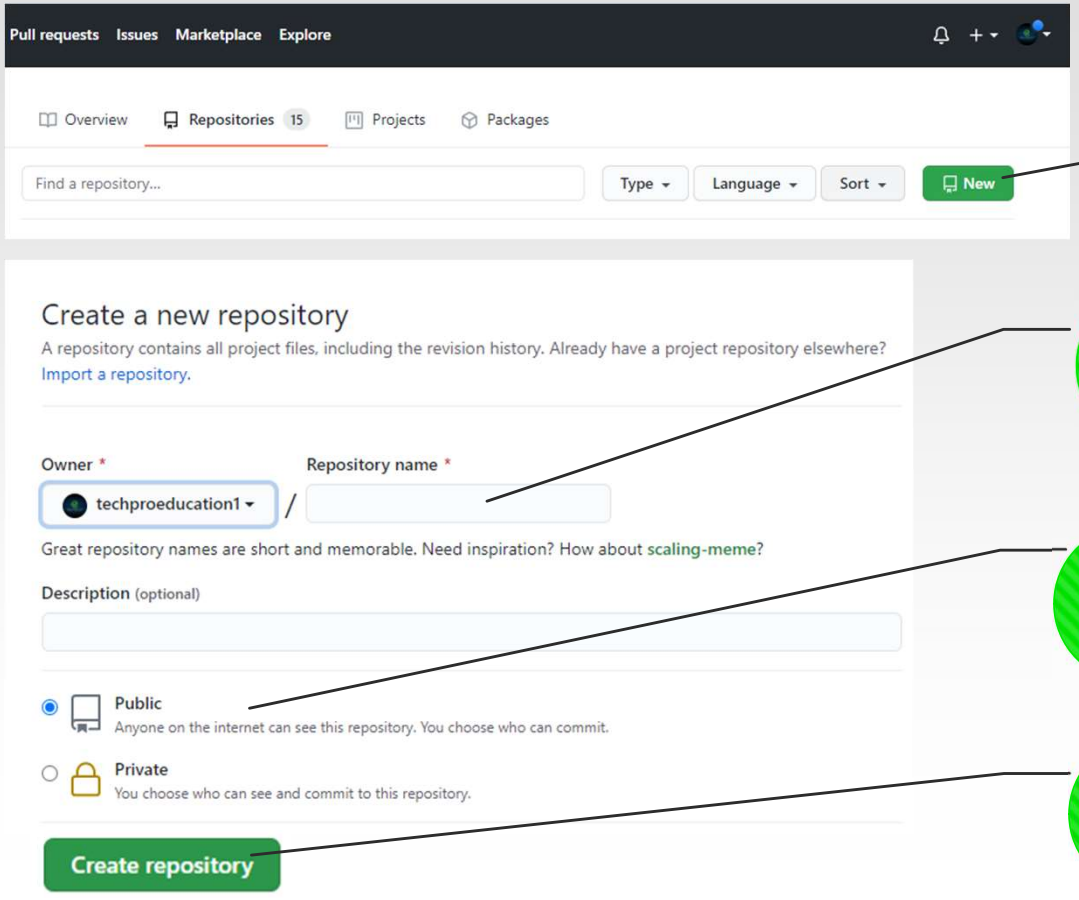
Create Account butonuna basıyoruz

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 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.
Fetch	Github daki en son versiyonun, lokal ile karşılaştırılarak –varsa- değişikliklerin indirilmesi işlemidir.
Merge	İndirilen değişikliklerin lokale uygulanması işlemidir
Pull	Fetch ve Merge işlemini tek başına yapar

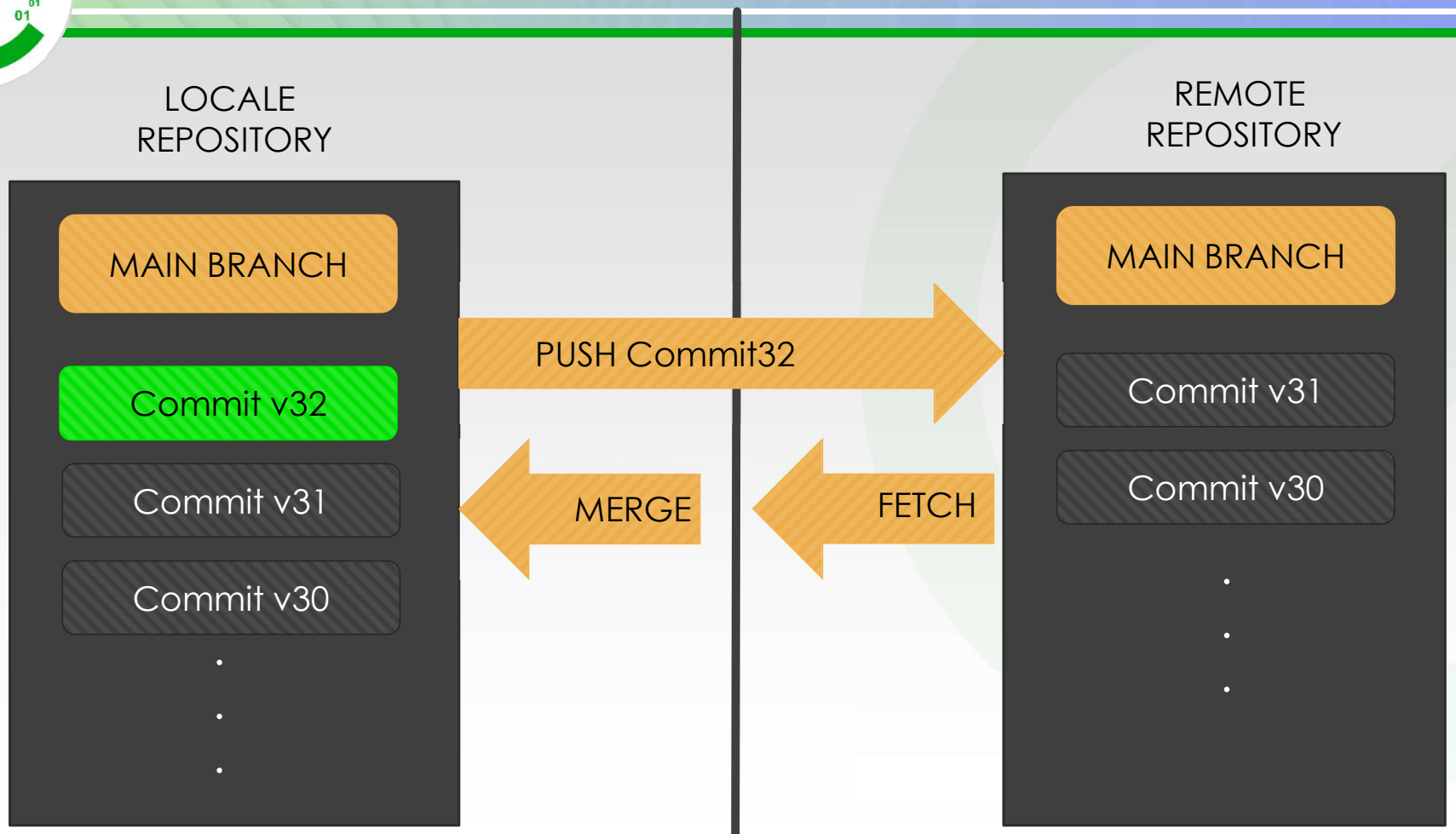
Cloning

Github daki bir reponun lokale indirilerek geliştirilmesi için öncelikle klonlanmalıdır. Bunun için aşağıdaki komut kullanılır. **Yani burada remote'daki repo'yu local'e indiririz.**

git clone *[git url]*

Directory'miz neresiyse remote'daki proje oraya indirilir.

Github Çalışma Prensibi



Github Çalışma Prensibi

Main Branch çalışmanın son halidir. Bu yüzden main'deki belgeler her zaman önceden gözden geçirilmiş ve onaylanmış olmalıdır. Bütün kullanıcılara 'main' e push etme yetkisi vermek best practice değildir.

Local Repo'nun Github'a Yüklenmesi

- Local 'de en az 1 commit oluşturduktan sonra, **her proje için bir kereye mahsus olmak üzere** aşağıdaki komutlar kullanılır. Böylece local repo ile Github ilişkilendirilmiş olur.

1 git remote add origin *[REMOTE_URL]*

Remote url, Github sitesi üzerinden ilgili repo sayfasında code bölümünden elde edilir TS0

2 git push --set-upstream origin master TS1

Local master branch, remote repo ya gönderilmiş olur.

TS2

Git push origin master komutu kullnaıldığında eğer daha önceden Github hesabına login olunmamışsa login ekranı açılabilir veya kullanıcı adı şifre/ssh key istenebilir.
Ssh key şuradan oluşturulabilir: https://github.com/settings/keys
Key oluşturulduktan sonra bir yere kaydedilmelidir!!

Slide 36

TS0 remote repo'yu local repoyla bağlamak için

* git remote add origin
i kullanırız.
Tevfik SASTIM, 2022-01-15T12:30:00.184

TS1 Burada origin local branch'mız master remote branch'imiz
Tevfik SASTIM, 2022-01-15T12:31:30.884

TS2 Zaten sonrasında bizden kullanıcı adı ve şifre ister
Tevfik SASTIM, 2022-01-15T12:33:43.725

Local Repo nun Github a Yüklenmesi

- İlk yüklemeden sonra yapılacak yeni local commit'leri Github a göndermek için aşağıdaki komutun kullanılması yeterlidir.

```
git push
```

* Sonrasında shell'de çıkan url'deki branches seçeneğine tıklanarak push kontrol edilir.

.gitignore

Staging area' ya gitmesini istemediğimiz, yani versiyon kontrol sistemine dahil etmek istemediğimiz dosya ve klasörlerimizi tanımladığımız özel bir dosyadır. Bu dosya .git klasörünün bulunduğu dizinde olmalıdır.

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

.gitignore

Github dan commit çekme

- Github üzerinden bir commit ile local repo güncellenmek istenirse aşağıdaki komutlar kullanılır

git fetch

Değişiklikleri remote'dan local'e indirir

git merge

İndirilen değişiklikleri local repoya uygular

VEYA

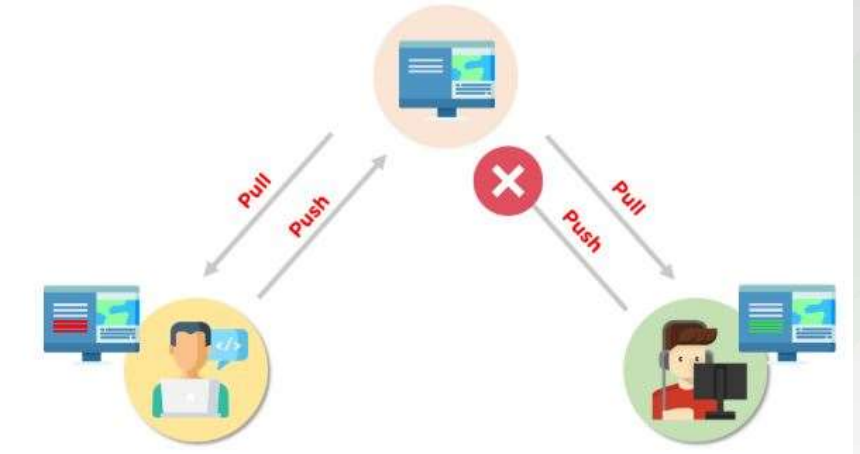
git pull

fetch & merge

Merge Conflict

Birleştirilecek commitlerde, aynı dosyanın aynı satırında birbirinden farklı değişiklikler varsa bu durumda merge işlemi sırasında çakışma oluşur. Buna **merge conflict** denir.

Merge conflict, remote-local birleştirmelerinde veya branch birleştirmelerinde gerçekleşebilir.

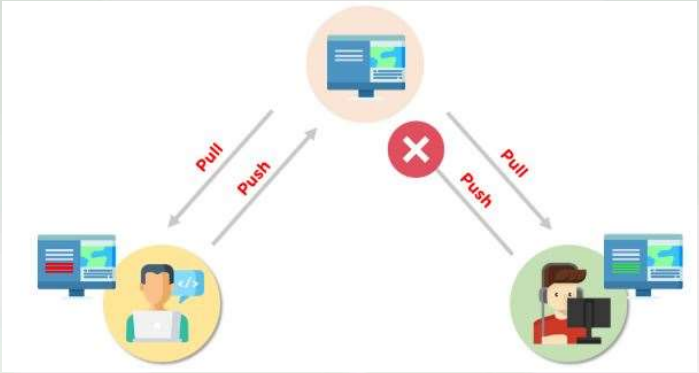



```
$ cat merge.txt
<<<<<<< HEAD
this is some content to mess with
content to append
=======
totally different content to merge later
>>>>>>> new_branch_to_merge_later
```

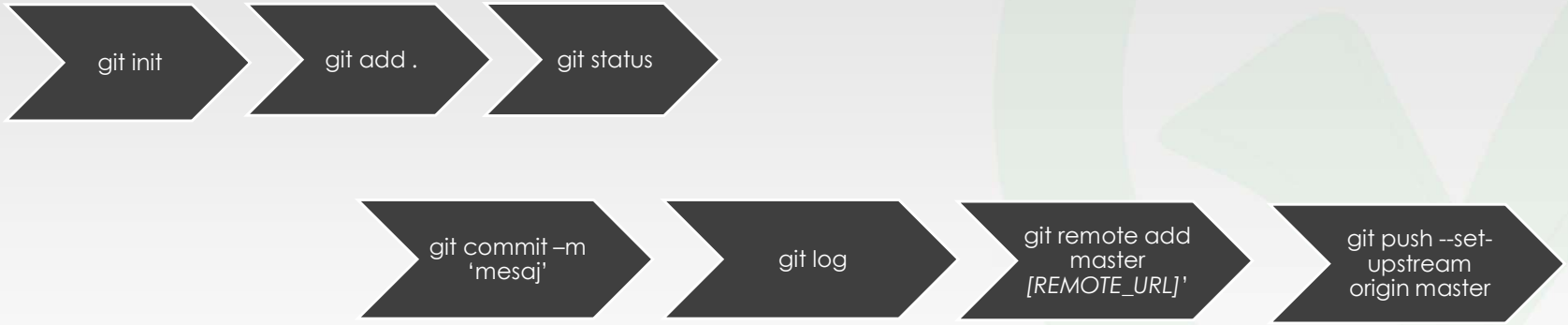
Merge Conflict

Merge conflict olduğunda neler yapılabilir ?

1. Çakışmanın oluştuğu dosya açılır ve manuel olarak çakışma giderilir. Dosya kaydedilir ve ardından tekrar bir commit oluşturulur.
   git commit –m "conflict çözüldü"
2. Ya da birleştirme işleminden vazgeçilebilir
   git merge –abort

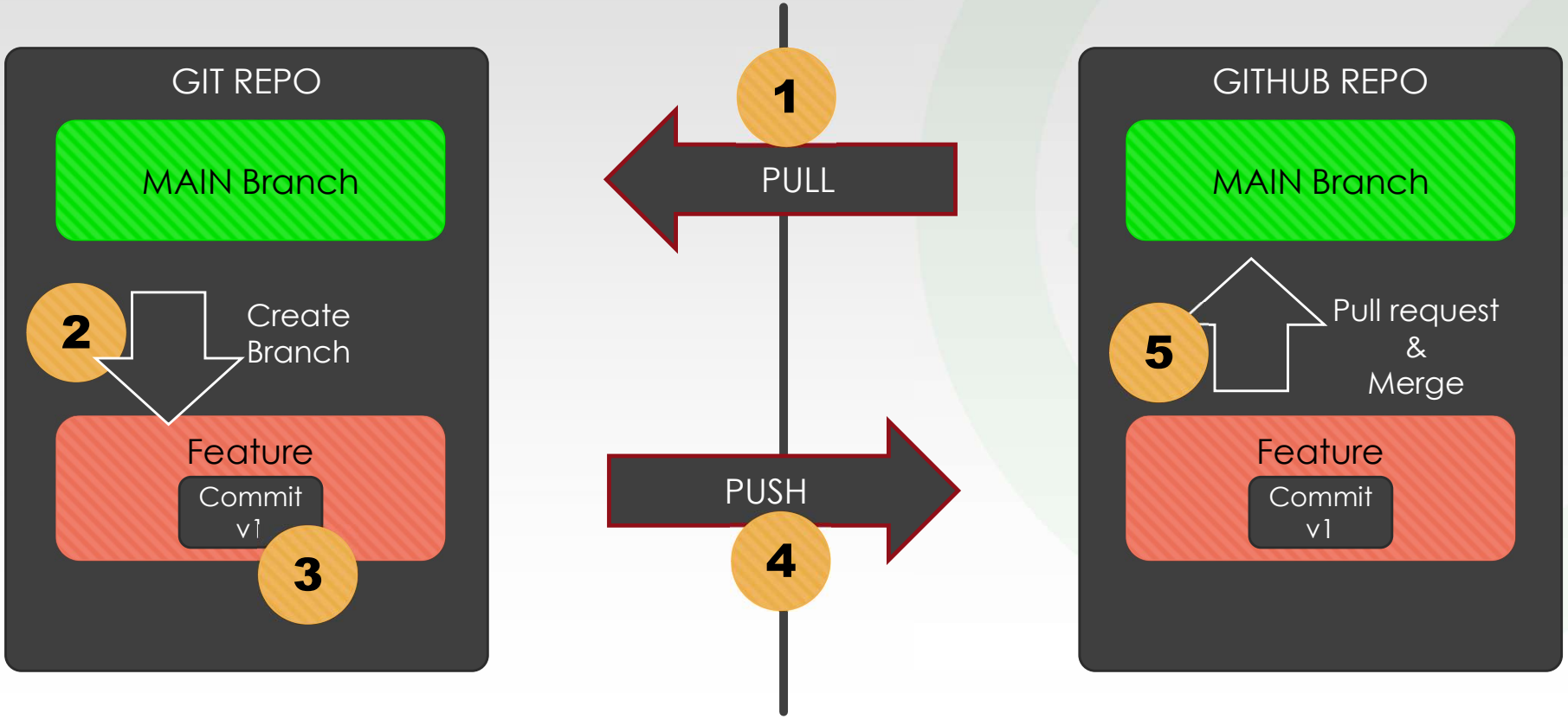


GitHub Lokaldeki Veriyi Remote' a Yüklemek İçin



Git – Github Çalışma döngüsü

BEST PRACTISE



Git – Github Çalışma döngüsü

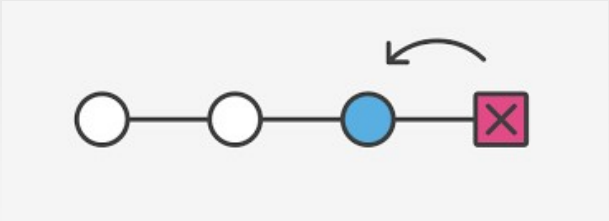
- Bu modelde Local Master da güncelleme yapılmaz. Local Master sürekli remote master ile beslenerek diğer developer ların ne gibi güncellemeler yaptıkları izlenerek local branch üzerinde oluşabilecek conflict ler önlenebilir.

Birden fazla colloborator ile çalışılan repo larda feature branch i push yapmadan önce mutlaka master pull yapılmalı ve olabilecek conflict ler düzeltilmelidir.

Git Reflog

- Git te yapılan bütün işlemlerin çıktısıdır.

  - git reflog



- git reflog kullanılarak geçmiş işlemlerin izi sürülebilir, gerektiği durumlarda bu işlemler modifiye edilebilir.

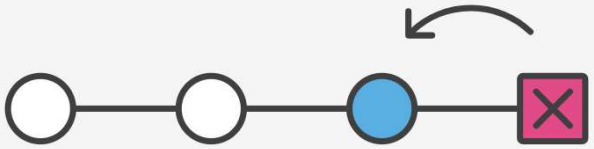
Git Reflog

- Mesela **git reflog + git reset –hard <hashkodun 7 hanesi>** ile hash kodununun 4 hanesi verilen state'e geri dönülebilir.

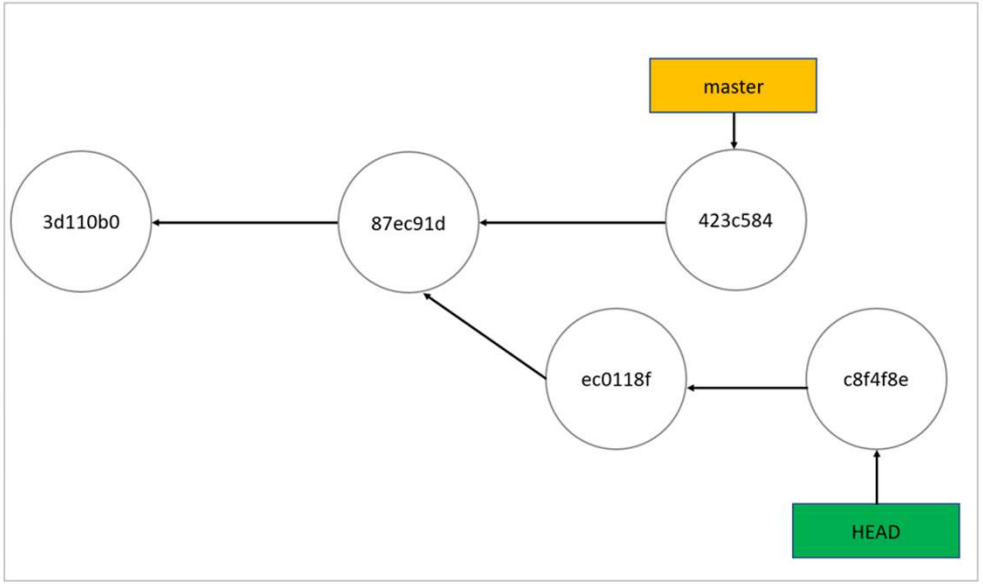
```
bcb8fb1 (deneme1) HEAD@{4}: checkout: moving from deneme1 to master
bcb8fb1 (deneme1) HEAD@{5}: Branch: renamed refs/heads/deneme2 to refs/heads/deneme1
bcb8fb1 (deneme1) HEAD@{7}: checkout: moving from master to deneme2
bcb8fb1 (deneme1) HEAD@{8}: checkout: moving from deneme1 to master
d86e0ad HEAD@{9}: checkout: moving from d86e0addb245baf67d8d03e877a0f08c3a421344 to deneme1

talfi@DESKTOP-CK670UE MINGW64 ~/devops/Git (master)
$ git reset --hard bcb8fb1
HEAD is now at bcb8fb1 Revert "Geri aldık"  reverts commit d86e0addb245baf67d8d03e877a0f08c3a421344.

talfi@DESKTOP-CK670UE MINGW64 ~/devops/Git (master)
$
```

Git Reflog

- **Not!: git log** sadece commit'lerle ilgili bilgileri temel bilgileri verirken **git reflog** repo'nun geçmiş durumuyla ilgili kullanıcıyı daha detaylı bilgilendirir.

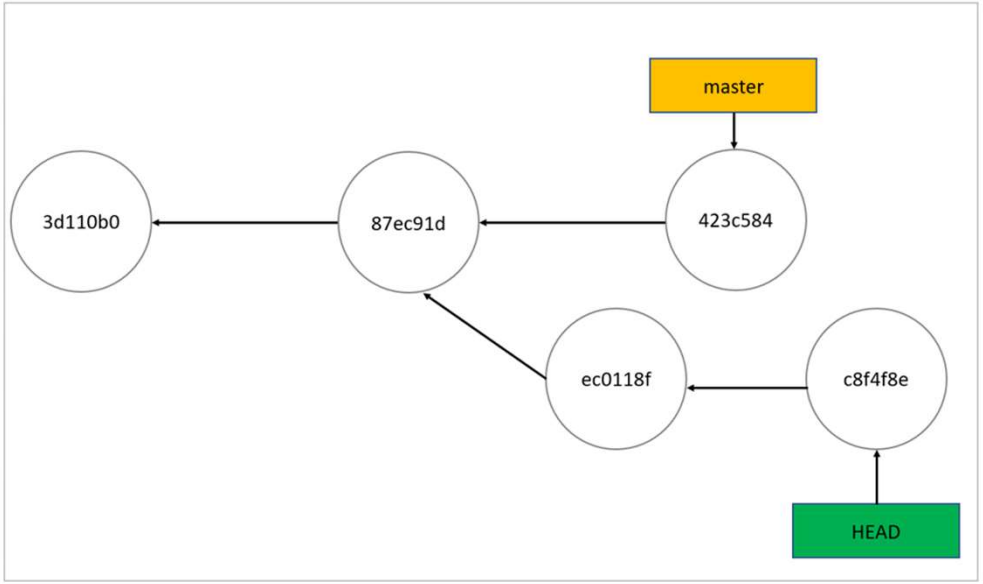


EKLER



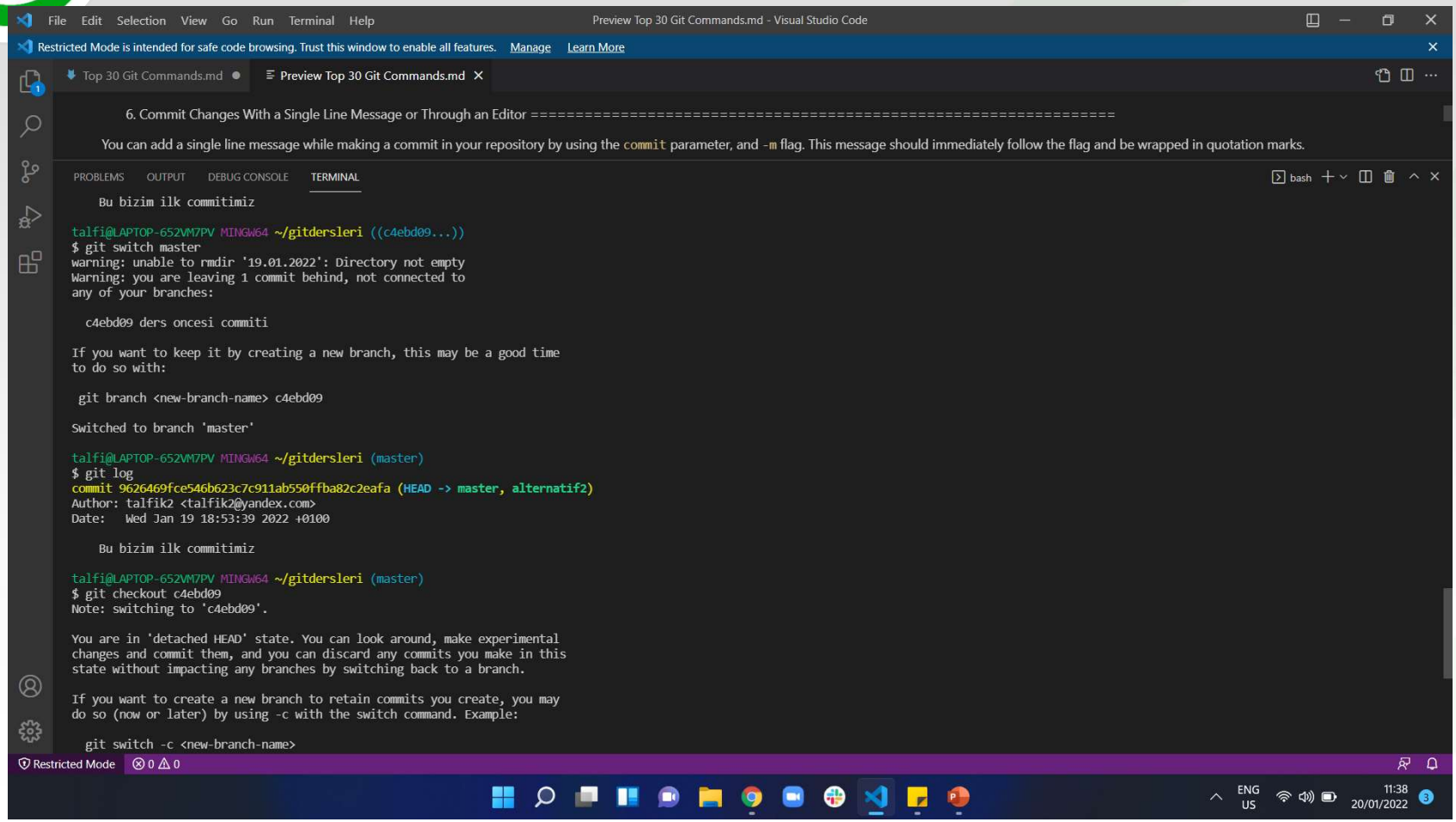
- 87ec91d bizim DETACHED HEAD state'imiz.
- Detached HEAD yazisini aldigimiz butun comitler master'dan değil de 87ec91d nin actigi yeni branch'ten devam eder.

EKLER

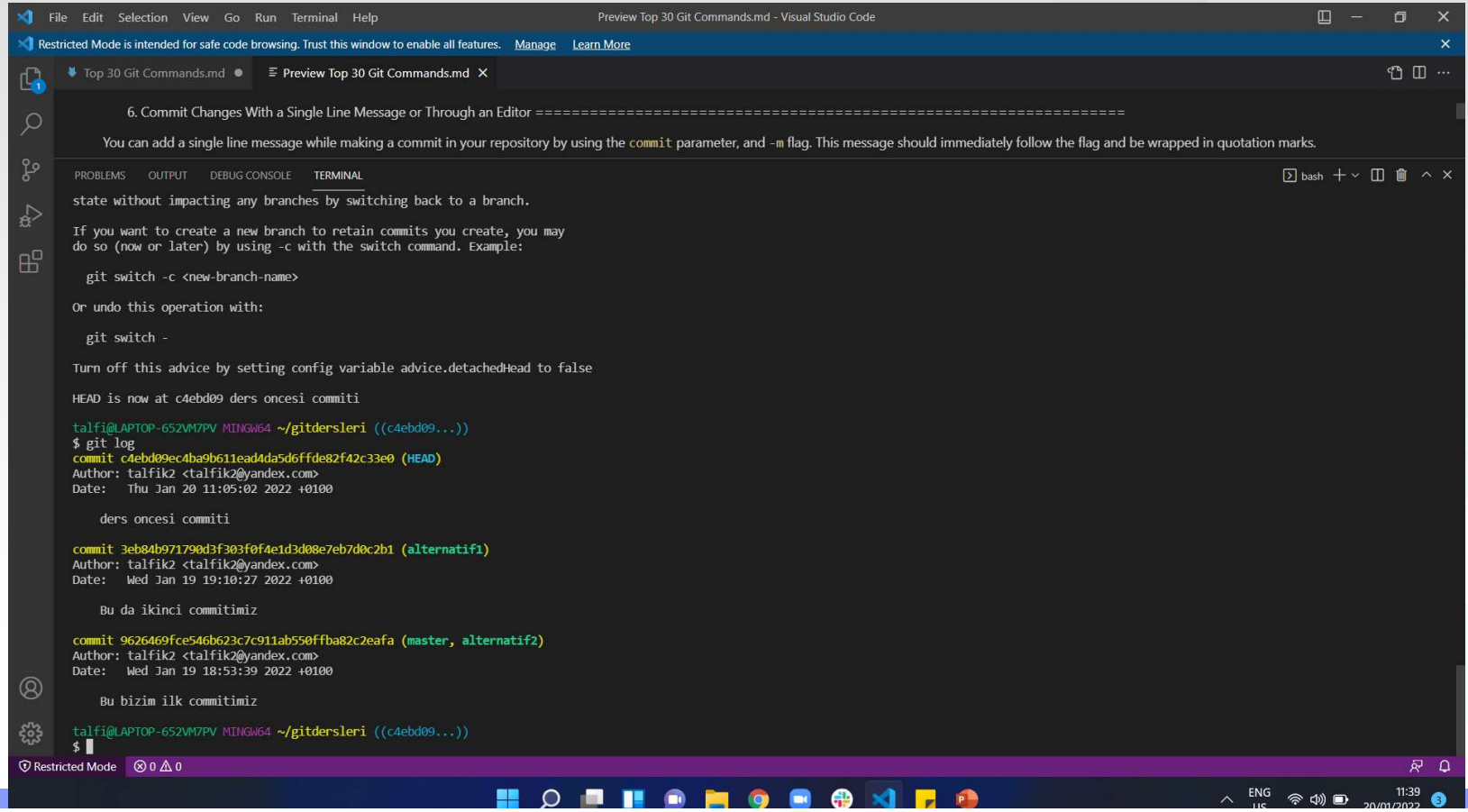


- HEAD'in mantigi Git Repo'muzun anlik konumunu vermesi. Yani başka bir degisle HEAD ile su an neredeyim? Hangi committeyim? Sorularinin cevabini aliyoruz.
- HEAD ve DETACHED HEAD ile ilgili daha detayli bilgi sahibi olmak icin: https://www.cloudbees.com/blog/git-detached-head

EKLER


File Edit Selection View Go Run Terminal Help
Preview Top 30 Git Commands.md - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Top 30 Git Commands.md
Preview Top 30 Git Commands.md
6. Commit Changes With a Single Line Message or Through an Editor ==========
You can add a single line message while making a commit in your repository by using the commit parameter, and -m flag. This message should immediately follow the flag and be wrapped in quotation marks.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
Bu bizim ilk commitimiz
talfi@LAPTOP-652VM7PV MINGW64 ~/gitdersleri ((c4ebd09...))
$ git switch master
warning: unable to rmdir '19.01.2022': Directory not empty
Warning: you are leaving 1 commit behind, not connected to
any of your branches:
c4ebd09 ders oncesi commiti
If you want to keep it by creating a new branch, this may be a good time
to do so with:
git branch <new-branch-name> c4ebd09
Switched to branch 'master'
talfi@LAPTOP-652VM7PV MINGW64 ~/gitdersleri (master)
$ git log
commit 9626469fce546b623c7c911ab550ffba82c2eafa (HEAD -> master, alternatif2)
Author: talfik2 <talfik2@yandex.com>
Date: Wed Jan 19 18:53:39 2022 +0100
Bu bizim ilk commitimiz
talfi@LAPTOP-652VM7PV MINGW64 ~/gitdersleri (master)
$ git checkout c4ebd09
Note: switching to 'c4ebd09'.
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:
git switch -c <new-branch-name>
Restricted Mode
ENG US
11:38 20/01/2022

EKLER


File Edit Selection View Go Run Terminal Help
Preview Top 30 Git Commands.md - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Top 30 Git Commands.md
Preview Top 30 Git Commands.md
6. Commit Changes With a Single Line Message or Through an Editor ==========
You can add a single line message while making a commit in your repository by using the commit parameter, and -m flag. This message should immediately follow the flag and be wrapped in quotation marks.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
state without impacting any branches by switching back to a branch.
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:
git switch -c <new-branch-name>
Or undo this operation with:
git switch -
Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at c4ebd09 ders oncesi commiti
talfi@LAPTOP-652VM7PV MINGW64 ~/gitdersleri ((c4ebd09...))
$ git log
commit c4ebd09ec4ba9b611ead4da5d6ffde82f42c33e0 (HEAD)
Author: talfik2 <talfik2@yandex.com>
Date: Thu Jan 20 11:05:02 2022 +0100
ders oncesi commiti
commit 3eb84b971790d3f303f0f4e1d3d08e7eb7d0c2b1 (alternatif1)
Author: talfik2 <talfik2@yandex.com>
Date: Wed Jan 19 19:10:27 2022 +0100
Bu da ikinci commitimiz
commit 9626469fce546b623c7c911ab550ffba82c2eafa (master, alternatif2)
Author: talfik2 <talfik2@yandex.com>
Date: Wed Jan 19 18:53:39 2022 +0100
Bu bizim ilk commitimiz
talfi@LAPTOP-652VM7PV MINGW64 ~/gitdersleri ((c4ebd09...))
$
Restricted Mode
0 0
ENG
11:39
20/01/2022