BATCH : BATCH 48

LESSON : GIT & GITHUB

DATE : 19.03.2022

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

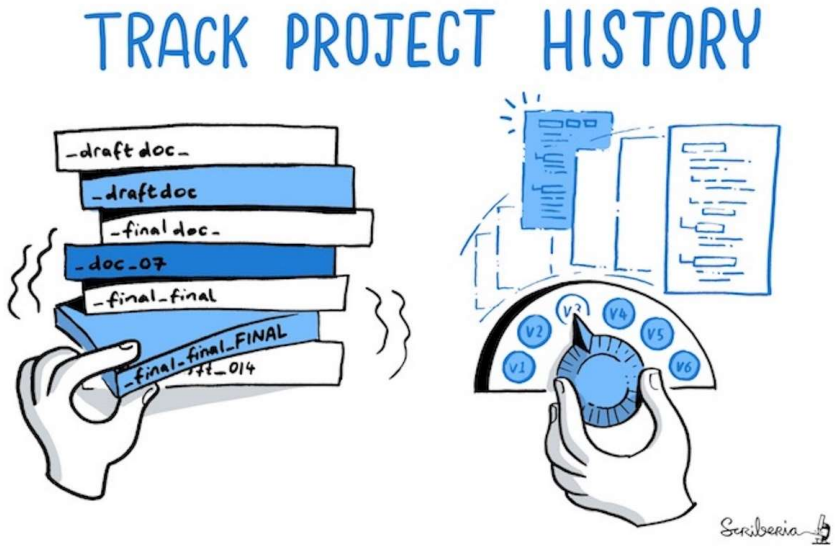
Versiyon Kontrol Sistemi

- VKS, bir uygulamada belli değişikliklerden sonra, o ana kadar ortaya çıkan ürün ile iligli bir versiyon oluşturulması, yeni değişikliklerin ayrı bir versiyona konulması işlemidir.

Version Control System

VKS nedir

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



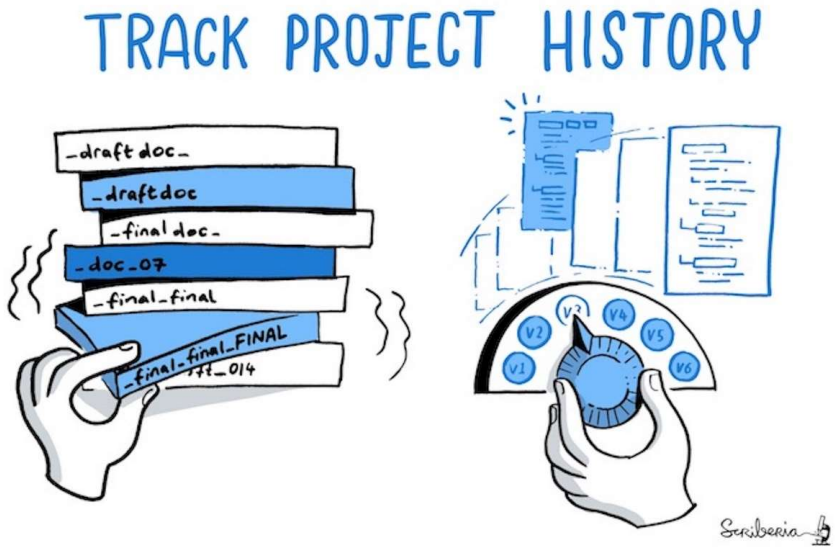
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

VKS nedir

Versiyon Kontrol Sistemi,

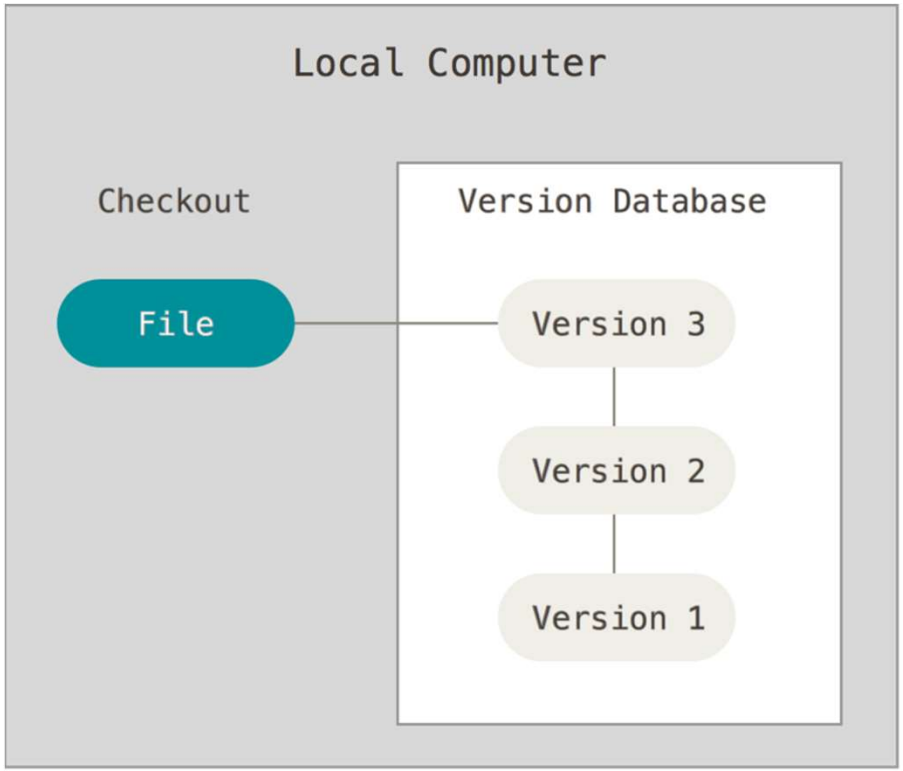
- seçili dosyaların bir önceki versiyona döndürülmesi,
- projenin tamamının bir önceki versiyona döndürülmesi, zaman içerisinde yapılan değişikliklerin karşılaştırılması,
- probleme neden olabilecek değişikliklerin en son kimin tarafından yapıldığı gibi bir çok işlemin gerçekleştirilebilmesini sağlar.

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

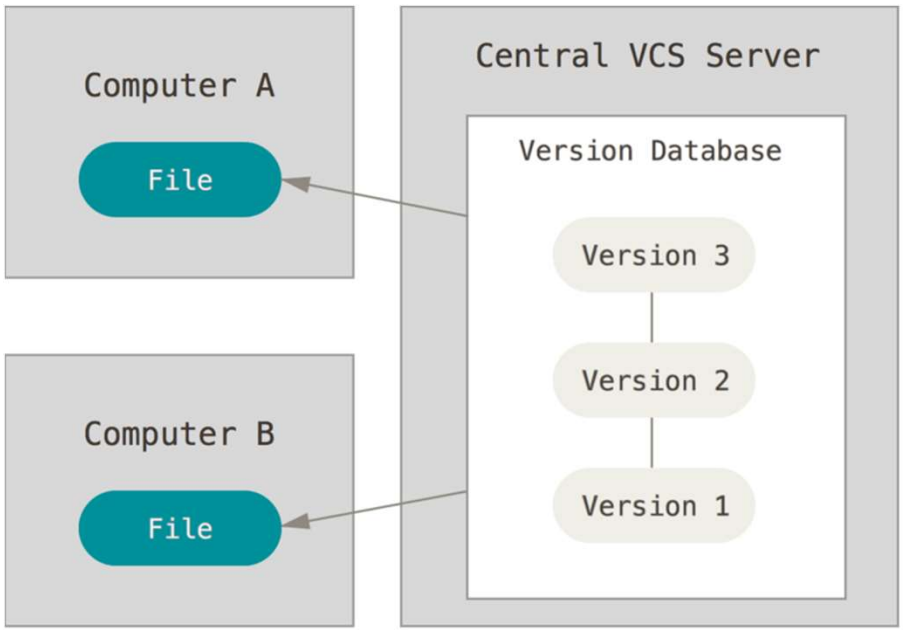
Yerel VKS nedir



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

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***.

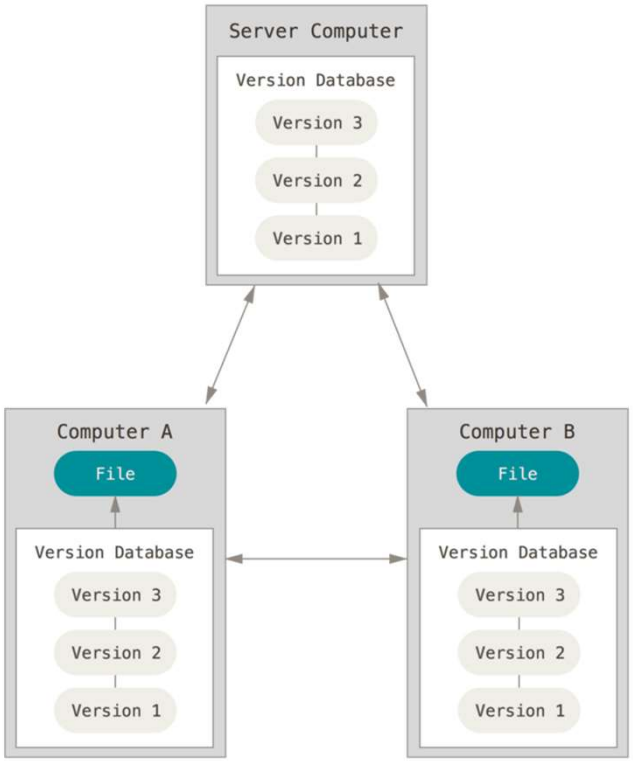
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Merkezi VKS nedir

- 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**.
- CVS Merkezi VKS'ye bir ornektir.

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

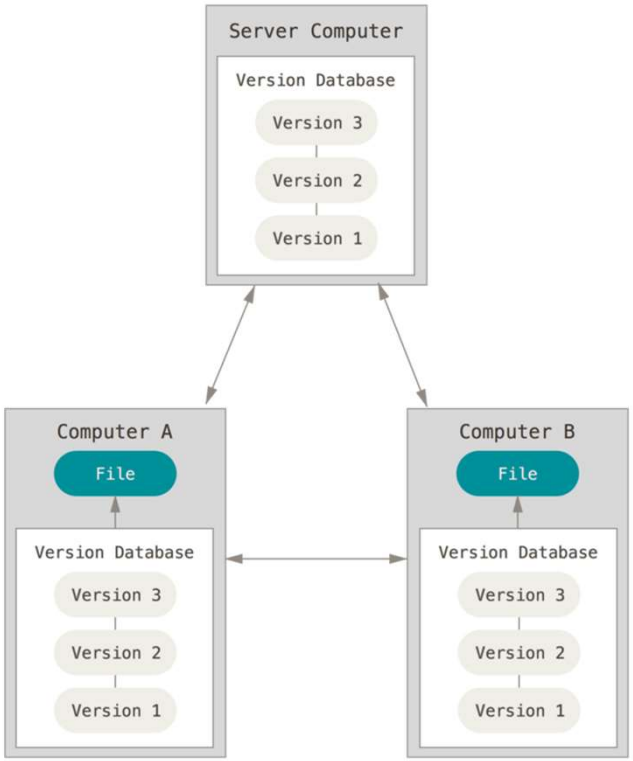
Dağıtık(Distributed) VKS nedir



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 **Daginik VKS' de hem merkezi bir sunucu bulunmaktadır, hem de client'larda da aynı yapının bir kopyası bulunmaktadır.**

Dağıtık(Distributed) VKS nedir



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

- 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 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.

- 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

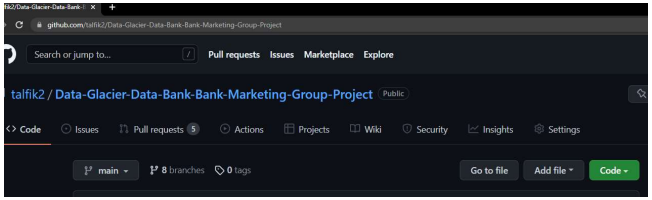
- Versiyon kontrol ve birlikte çalışma altyapısını ayrı tutmak istediğimiz her bir bağımsız yapıya **repository** denir. Genellikle her proje için ayrı bir repository tanımlanır.

https://github.com/talfik2/Data-Glacier-Data-Bank-Bank-Marketing-Group-Project

GitHub Kullanimlari

- 1. Console Yoluyla

- 2. Websitesi yoluyla



- 3. GitHub Desktop Yoluyla

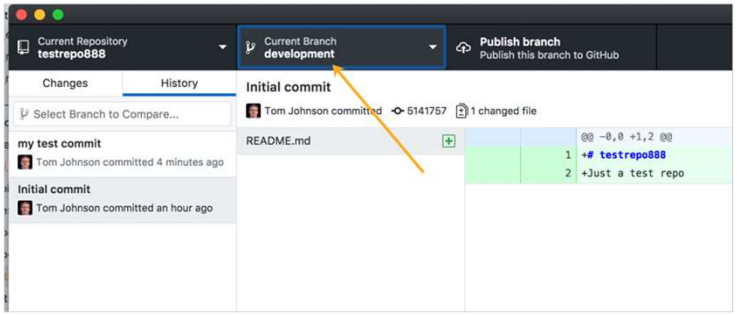


Image Credits: https://idratherbewriting.com/learnapidoc/pubapis_github_desktop_client.html

Genel Kavramlar

REPOSITORY

Working Space

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

Staging Area

Versiyon oluşturulacak(commit edilecek) olan dosya veya klasörlerin geçici olarak toplandığı yerdir. Versiyon oluşturulduktan(commit edildikten) 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 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

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.

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