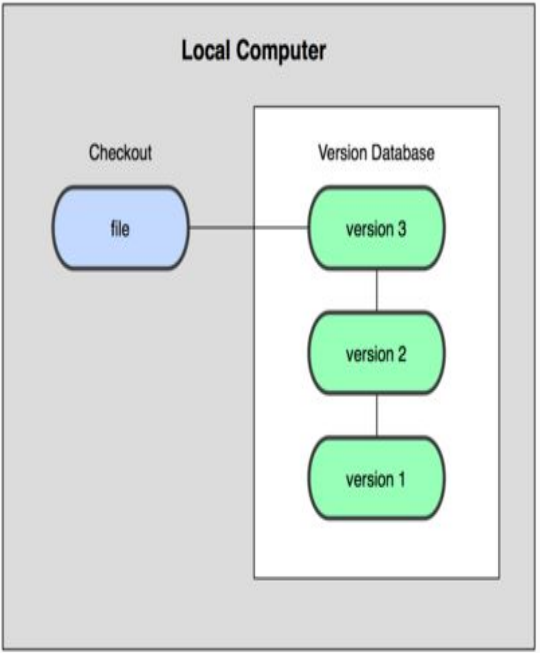
Git Nedir?

Git bir *sürüm/versiyon kontrol sistemi*dir. Yazdığımız projeleri, bilgisayarımızda ya da harici disklerde tehlike altında değilde internet üzerinde tutmamızı ve yönetmemizi sağlayan bir sistemdir.

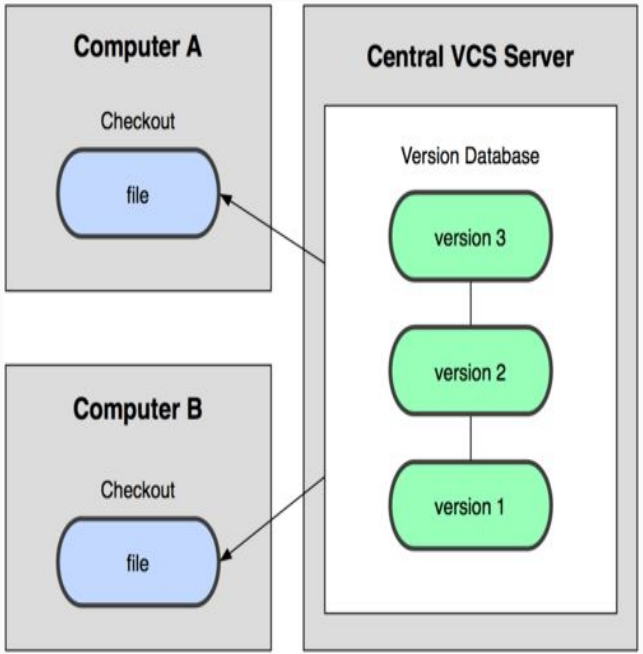
Neden git kullanmalıyız ?

- Dağıtık Yapı
- Versiyonlama
- Önceki versiyonlara dönebilme
- Ekip çalışması

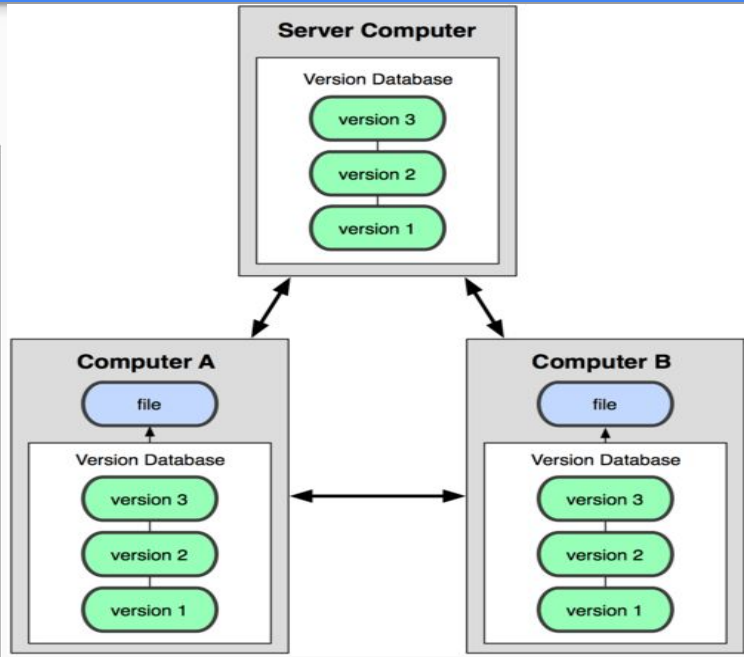
Git Kullanmaya Başlamak



Yerel Sürüm Kontrol Sistemleri



Merkezi Sürüm Kontrol Sistemleri



Dağıtık Sürüm Kontrol Sistemleri

Birden fazla geliştirici ile yapmakta olduğumuz projede, yaptığımız değişiklikleri, hangi dosyada ne değişiklik yaptığımızı kaydetmek, bunu geçmişe yönelik olarak kontrol etmek ve bu kontrol ettiklerimizi tek bir orta noktada toplamak istiyoruz.Git burada devreye giriyor.Git versiyonlamaları yapmamız için bilgisayarımızda yerel depo oluşturuyor.Kendi proje kayıtlarımızı öncelikle kendi bilgisayarımızda depo halinde tutuyoruz.Yaptığımız değişiklikleri stage (sahne) e alıyoruz.Sonra bunu yeni versiyon olarak kendi depomuza gönderiyoruz.Bilgisayarımızdaki depoya uzak bağlantı olarak merkezi depoyu ekliyoruz ve bu depo üzerinden merkezi depo ile konuşuyoruz.

git init

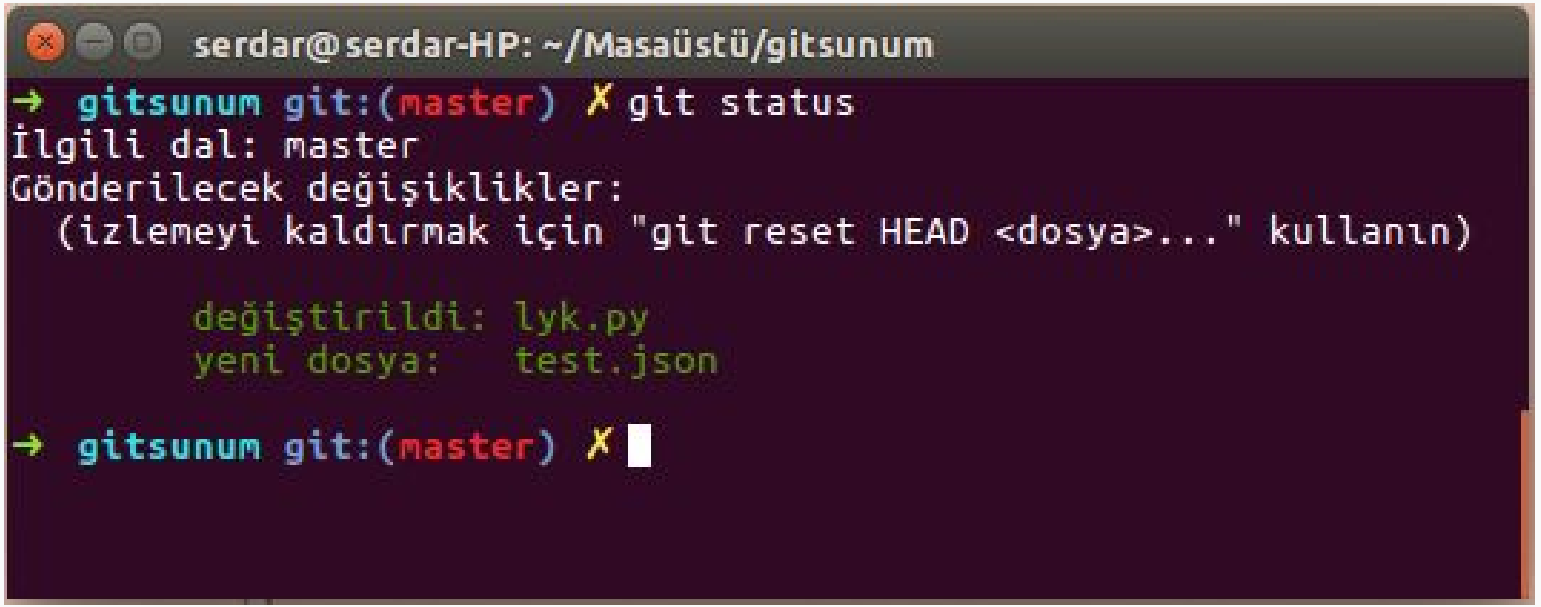
git init komutu bulunduğumuz dizinde boş bir git deposu oluşturur.

/.git/ uzantılı bir depo oluşur bu boş depoyu görüntülemek için
ls -a komutu kullanılır.

git status

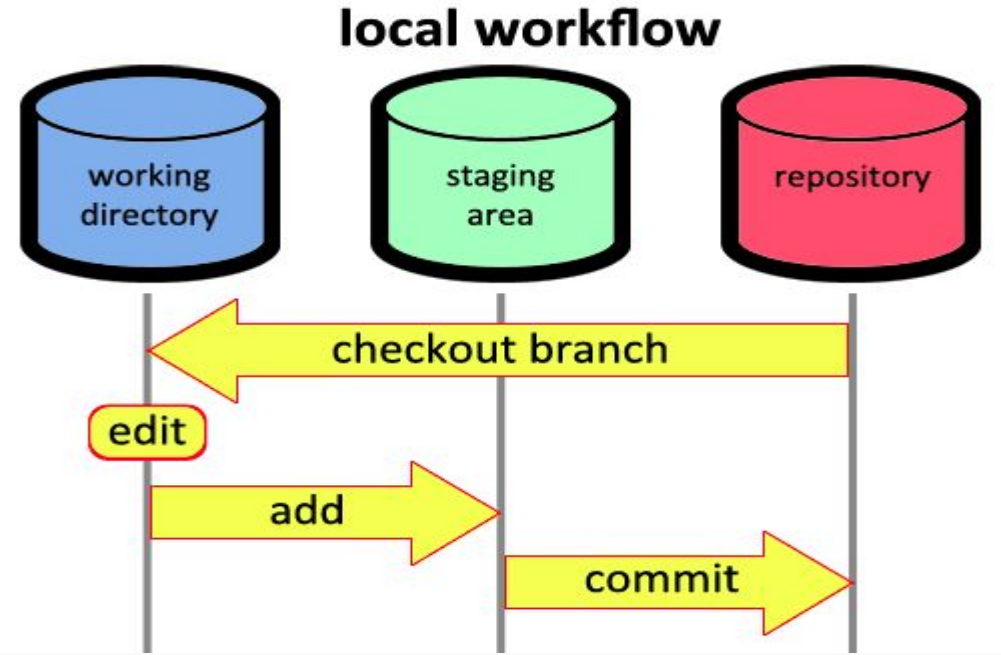
git status komutu bize çalışma dizinimizin durumunu gösterir. Örnek olarak yeni bir dosya eklediysek, bir dosyada değişiklik yaptıysak ya da bir dosyayı sildiysek bunları yerel depomuza yollamadan önce görmemizi sağlar.

Örnek git status komutu



git add

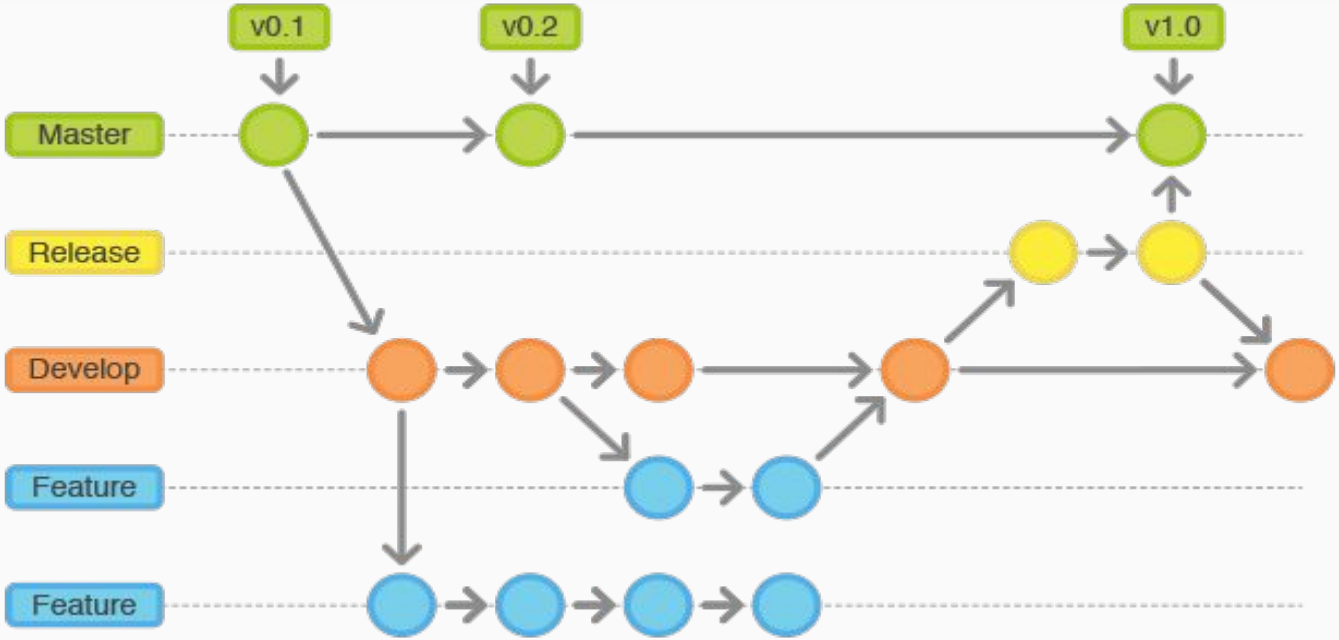
git add deneme.txt (dizin içerisindeki stage e alınmak istenilen dosya) işaret edilen dosyayı sahneye(stage) alır.Depoya işlenmek için hazır hale getirir.



git commit -m “yapılan değişiklik”

git commit komutu yaptığımız değişikliklerin yerel depomuza gönderilmesini ve yeni bir versiyon oluşturmamızı sağlar. git commit’ten sonra kullandığımız **-m** ve sonrası ise yaptığımız değişiklikleri tırnak içine yazarak, daha sonra ne gibi değişiklikler yaptığımızı kolay bir şekilde görmemizi sağlar.

Dallanma nedir? (Branch)



Branch üzerinde çalıştığınız kaynak kodun farklı versiyonlarına sahip olmamızı sağlar.

Örnek vermek gerekirse, her git deposu ön tanımlı olarak master branch'ı ile gelir. Eğer isterseniz geliştirme adlı ekstra bir branch açabilir, kod üzerinde değişiklikleri burada yapabilir ve son olarak stabil olduğundan emin olduğunuz değişiklikleri master branch'e aktarabilirsiniz.

Yeni bir branch oluşturma.

Yeni bir branch oluşturmak için aşağıdaki komutu kullanabiliriz.

- **git branch yeni_branch**

Başka bir branch’e geçmek.

Başka bir branch’e geçmek için aşağıdaki komutu kullanıyoruz.

- **git checkout branch_adi**

Sunucu Ekleme(Github,gitlab,bitbucket vs.)

Uzaktaki bir yazılım havuzunu kısa bir ad vererek eklemek için
git remote add [kisa_ad] [url] komutunu çalıştırırız:

```
$ git remote
origin
$ git remote add pb git://github.com/paulboone/ticgit.git
$ git remote -v
origin  git://github.com/schacon/ticgit.git
pb  git://github.com/paulboone/ticgit.git
```

Dosyaları merkezi depoya yollama.

Dosyalarınızı yerel depoya ekledikten sonra aşağıdaki komut ile github, gitlab gibi merkezi bir depoya yollayabiliriz.

- **git push origin master**

Uzak uç birimden dosya çekme

Sunucudan diğer takım arkadaşlarımız tarafından eklenen bilgileri çekmek ve bu değişiklikleri bizim o anki projemiz ile otomatik merge etmek istiyorsak **git pull** komutunu kullanmanız gerekmektedir. Bir projeyi sıfırdan çekmek için ise **git clone [url]** şeklinde bir komut kullanmamız daha uygun olacaktır.

git pull komutu, değişiklikleri al *(fetch)* ve birleştir *(merge)* yapacaktır.

Pull/Merge request nedir ?

Açık kaynak bir projeye katkı sağlamak için öncelikle github, gitlab gibi sitelerden projenin kaynak kodlarını kendi hesabımıza kopyalamamız gerekir. Kopyaladıktan sonra yaptığımız bütün commitler kendi hesabımızdaki projede geçerlidir. Eğer işimiz bitti ve kodumuz doğru bir şekilde çalışıyor ise bizim kodumuzu ana projeyle birleştirmemiz gerekiyor fakat ana proje bize ait değil, dolayısıyla kodu doğrudan birleştiremeyiz. Bu durumda ana projenin sahibine yazdığımız kodu kendi koduyla birleştirmesi için bir istekte bulunmamız gerekiyor. Bu işleme **pull request** denir.

Commit edilen bir değişikliği geri almak

Hatalı bir düzenleme yaptığımızda veya geliştirdiğimiz bir özelliğin artık gerekli olmadığına karar verildiğimizde yaptığınız değişikliği geri almanız gerekecektir.

git revert komutu commit ettiğimiz herhangi bir değişikliği geri almak için kullanılır. Bu komut ile commit işlemimizin kendisi veya bilgileri silinmez sadece commit işleminizdeki değişiklik geri alınır. Örneğin eklediğimiz bir satırı kaldırmak isterseniz **git revert** komutu ile bunu yapabilirsiniz. Aslında git revert komutu değişikliğinizi geri almak için otomatik olarak yeni bir commit oluşturur ve geri alma işlemi bu commit sayesinde değişiklik tarihçesinde görünür hale gelir.