Git Nedir?

Git bir versiyon kontrol sistemidir. **Versiyon kontrol sistemi** (Version Control System) bir proje üzerinde yaptığımız değişiklikleri adım adım kaydeden ve isterseniz bunu internet üzerinde depoda (repository) saklamamızı ve yönetmemizi sağlayan bir sistemdir. Ayrıca Git bir dağınık (distributed) versiyon kontrol sistemi olduğu için her bir istemci (client) sadece dosyaların en son anlık görüntüsünü kontrol etmekle kalmaz, aynı zamanda tam geçmişi de dahil olmak üzere depoyu tamamen kopyalar. Her clone gerçekten tüm verilerin tam bir yedeğidir. Böylece, herhangi bir sunucu ölürse ve bu sistemler bu sunucu aracılığıyla iş birliği yapıyorsa, istemci depolarından herhangi biri geri yüklemek için sunucuya kopyalanabilir. Bu nedenle dağınık bir versiyon kontrol sistemi olan Git'i kullanmak projemiz için oldukça güvenlidir.

Ayrıca Git, bir projede çalışırken yaptığınız değişikleri görebilmenizi, eski haline erişebilmenizi sağlar. Bu şekilde projenin her adımına erişebilir ve geliştirme süreçlerini sağlıklı bir şekilde izleyebiliriz. Ayrıca bir projede birden fazla kişi çalışması durumunda da aynı dosya üzerinde yapılan değişikliklerin sorun çıkmadan kontrol edilebilmesine, projelerin güncel versiyonlarına ulaşabilmeye yardımcı olur. Bu tür avantajlar sağlaması nedeniyle birçok yönden fayda sağlar ve hız artırır.

Başlıca Git Komutları ve Gitlab'a Proje Ekleme

Git kullanımına başlamak için ilk olarak config ayarlarının yapılması gerekmektedir.

Terminali açıyoruz ve ardından;

```
git config --global user.name "<ad>" komutunu ve,

git config --global user.email "<email>" komutunu giriyoruz.
```

- Config ayarları bilgisayar üzerinde tek bir kullanıcı varsa global olarak yapılabilir. Farklı projelerinizde farklı config ayarı yapmak isterseniz terminal üzerinden proje dizinine ilerledikten sonra global komutu olmadan da config ayarlarını yapabilirsiniz.

İlk olarak terminali açıyoruz ve projemizin olduğu dizine gidiyoruz;

```
cd  proje_dizini
```

İlk Komut : git init

```
#make directory a git repository

$ git init
```

Bu komut bir dizini boş bir Git deposuna dönüştürür. Bu, bir depo (repository) oluşturmanın ilk adımıdır. Git init'i çalıştırdıktan sonra, dosya/dizin eklemek ve işlemek mümkündür.

İkinci Komut : git add

Bir dosya bir depoya kaydedilmeden önce, dosyanın Git hazırlama alanına eklenmesi gerekir. Bu komut sayesinde çalışma dizinindeki dosyalar, Git için hazırlama alanına (stage area) eklenir. Belirli veya tüm unstaged dizinleri/ dosyaları **git add** komutu ile hazırlama alanına (stage area) ekleriz. Bu komutu kullanmanın birkaç farklı yolu vardır;

```
$ git add <file or directory name>

# To add all files not staged:

$ git add .

# To stage a specific file:

$ git add index.html

# To stage an entire directory:

$ git add css
```

Yukarıda da gördüğümüz gibi '**git add [dosya adı ya da klasör adı]** 'yazılarak stage area yani hazırlama alanına belirtilen dosyalar eklenir.

Ayrıca '**git add .'** komutu ile unstage alanındaki (working directory) **tüm** değişiklikleri hazırlama alanına ekleyebiliriz.

Üçüncü Komut : git commit

Sırada yaptığımız değişiklikleri commitlemek var. Bu komut, stage area yani hazırlama alanına aldığımız değişikliklerimizi yerel depoya (local repository) yüklemek için yapılması gereken işlemdir. Bu komut ile dosyalarda yapılan değişiklikleri yerel bir depoya kaydederiz.

Bu kaydı gerçekleştirirken commit'e mesaj eklemek oldukça faydalı bir uygulama olacaktır. Bu mesajları ekleyerek belirli bir değişikliği bulmak veya üzerinden zaman geçtikten sonra yapılan değişiklikleri anlamak oldukça kolay ve hızlı olur. Ayrıca kolayca erişebilmek ve görüntüleyebilmek için her commit'in benzersiz bir kimliği (hash) de vardır.

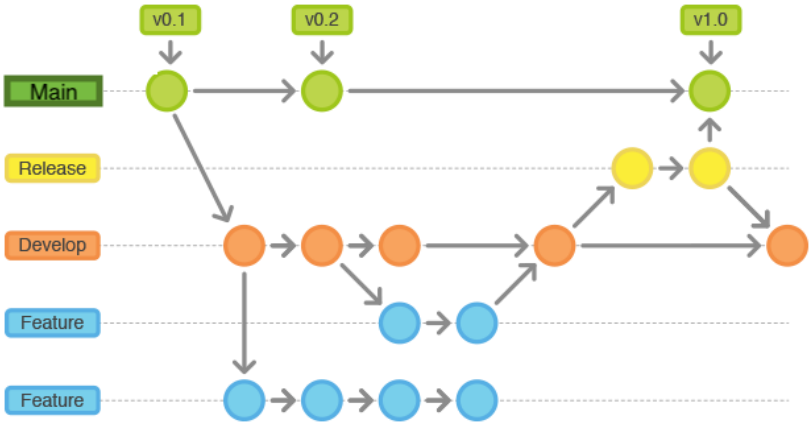
```
# Adding a commit with message

$ git commit -m "Commit message in quotes"
```

Dördüncü Komut : git branch

Branch kavramını Türkçe'ye dallanma olarak çevirebiliriz. Bir proje için ana dal, **master** dalıdır, daha doğrusu dalıydı, eskiden ana branch ismi master olarak anılırken ırkçılığı çağrıştırdığı gerekçesiyle artık **main** olarak isimlendirilmesine karar verildi.

Projemizi ilk açtığımızda karşımıza çıkan bu ana dalda çalışmak pek de doğru değildir. Şu an belki ilk kez bir projeniz var ve ilk kez Git kullanıyorsunuz ama zamanla yayınladığınız, insanlara ulaşan bir projeniz olabilir ve üzerinizde değişiklikler yapmanız ya da yeni özellikler (feature) eklemeniz gerekebilir. Bu gibi durumlarda yeni bir branch açmak ve oradan geliştirmelere devam etmek gerekir.

Bu durumu sizlere bir örnek ile açıklayabilirim. Diyelim ki projemize yeni bir özellik (feature) eklemek istiyoruz ama bu özelliğin projemizin güncel versiyonunu patlatma ihtimali var. Projemizin o anki haline bir şey olmadan geliştirmeye devam etmek ama bir yandan da diğer özelliği geliştirip denemek istiyorsunuz. Bu tarz bir durumla karşı karşıya kaldığınızda yapılacak en güzel şey yeni bir branch (dal) açarak orada geliştirmeyi denemek ardında da onu projenin branch güncel haline merge etmek olacaktır. Bu şekilde projeniz zarar görmeden istediğiniz kadar değişim yapabilirsiniz. Üstelik birden fazla branch açma olanağınız da var.



Main dalından açılan dev dalında geliştirmeler yapılıp en son main dalına birleştiriliyor.

Localde bulunan tüm brachleri görüntülemek için; `git branch` komutu kullanılır

Yeni bir dal oluşturmak için; `git branch <branchname>` komutu kullanılır.

Dallar arasında geçiş yapmak için `git checkout <branchname>` komutu kullanılır.

Dalları birleştirmek için ilk olarak ana dala geçilir ardından main ile hangi branch birleştirilmek isteniyor ise; `git merge <branchname>` komutu kullanılır.

Beşinci Komut : git remote add

Yerel depoya **origin** adıyla uzak depoyu (remote repository) ekleme işlemidir. Bu şekilde yerel depodaki değişikliklerimizi Gitlab'daki uzak depoya aktarabiliriz. Öncelikle Gitlab hesabınızda bir repository oluşturarak işe başlayabilirsiniz. Oluşturduğunuz repository'nin url sini kopyalayın. Ardından;

```
# Add remote repository

$ git remote <command> <remote_name> <remote_URL>

$ git remote add origin https://github.com/kullanıcı-adınız/repository-adınız.git
```

Komutunu terminale yazarak yerel depoda origin adlı bir remote ekleriz. (Burada origin yaygın olarak kullanılan bir isim tabiki onun yerine başka şeyler de yazabilirsiniz.)

Altıncı Komut : git push

Bu komut yerel commitleri uzak depoya gönderebilmemiz içindir. **git push** iki parametre gerektirir: uzak depo remote ismi ve pushlanacak dalın ismi.

```
# Push a specific branch to a remote with named remote

      $ git push origin dev
```

Github veya Gitlab'da oluşturduğunuz repositoryleri localde kullanmak için

`git clone <repositoryurl>` komutu kullanılır.

Organizasyon projelerinde localde değişiklik yapmadan önce repository de değişiklik yapılıp yapılmadığını anlamak için `git pull` komutu kullanılır. Projemizde ekleme-çıkarma işlemi yapmadan önce bu komutu kullanarak dosya dizinine güncel versiyonu çekmeliyiz.

Git hakkında yardımcı kaynaklar:

https://l24.im/zEBR

https://l24.im/1Ya

https://l24.im/2Qi