Git Nedir?

* Projeleri internet üzerinde tutmayı ve yönetmeyi sağlayan bir sistemdir.
* Versiyon yönetme kolaylığı sağlar. İstenildiğinde projenin son haline veya herhangi bir gün ki halini görebilirsiniz.
* Birden fazla kişinin eş zamanlı olarak aynı proje üzerinde çalışmasını, kod alışverişi yapabilmeyi sağlar.
* Git offline olarak kullanılabilir. Her bir kullanıcının bilgisayarında projenin bir kopyası mevcut olur. Ve bu sayede offline olarak da proje geliştirebilirsiniz.
* Hızlı çalışır ve az yer kaplar.

Başlıca Git Komutları

1. **‘git config’**

Kullanıcı adı, mail, dosya formatları gibi genel yapılandırma ayarlarını yapabileceğiniz komuttur.

Etc:

$ git config --global user.name "Kod5"

$ git config --global user.email [yazar@kod5.org](mailto:yazar@kod5.org)

1. **‘git init’**

Bu komut, bulunduğunuz dizini bir Git repository’si haline getirir ve .git adında bir dizin oluşturur. Daha sonra bu repository’i uzaktaki bir sunucuya gönderebilirsiniz.

Etc:

Kök dizininde MyProject adında bir proje oluşturalım. Daha sonra bu dizinde bir Git repository’si oluşturalım. Aşağıdaki komutları sırasıyla girdiğinizde konsola bir açıklama yazacaktır. “Belirtilen dizinde boş bir git repository’si oluşturuldu ” şeklinde.

mkdir MyProject

cd MyProject

git init

Initialized empty Git repository in /Users/YAZAR/MyProject/.git/

1. **‘git clone’**

Bu komut localde ya da uzak bir sunucuda yer alan repository’nin çalışan bir kopyasını bulunduğunuz dizine kopyalar.

Etc:

git clone /path/to/repository

git clone username@host:/path/to/repository

1. **‘git add**

**Belirli bir dosyayı ya da tüm projeyi çalışma dizinine (index) ekler. Yani commit’ lenmeye hazır hale getirir.**

**Etc:**

git add Readme.txt

git add \*

İlk satır ile, ‘Readme.txt‘ isimli dosya; ikinci satır ile de proje içerisindeki tüm dosyalar çalışma dizinine eklenmiş oluyor.

1. **‘git commit’**

**Çalışma dizinine eklenen dosyaları head denilen .git klasörü içindeki kısıma ekler. Commit işlemi yerelde gerçekleşen bir işlemdir. Uzak sunucudaki repository’ de bir değişiklik yapmaz.**

**Etc:**

git commit -m "Projede X işlemi yapıldı."

1. **‘git push’**

**Commit’ lenmiş değişikliklerin uzak sunucudaki repository’ e gönderilmesini sağlar.**

**Etc:**

git push origin master

1. **‘git status’**

Repository’niz in o anki durumu hakkında bilgi verir. Değişiklik yapılmış veya yeni eklenmiş fakat add ya da commit işlemi uygulanmamış dosyalar varsa bunları liste halinde size gösterir.

Etc:

git status

1. **‘git pull’**

**Uzak sunucudaki repository’ de değişikliğe uğramış ya da yeni eklenmiş dosyalar varsa bunları indirir ve yereldeki repository ile birleştirir.**

1. **‘branch’**

**Branch dal anlamına gelir. Bir proje üzerinde aynı anda birden fazla kişinin çalışmasına olanak verir. Proje için bir bir dal oluşturur ve o dal da yapılan değişiklikler o dal seçiliyken görülür. Bu Proje versiyonları için de geçerlidir.**

**Etc:**

**git branch yerelimizdeki branchleri ve hangi branchte olduğumuzu gösterir.**

**Git branch aliveli aliveli adında bir branch oluşturur.**

1. **‘checkout’**

**Bu komut brancler arasında geçiş yapmayı sağlar. Yani branch komutu ile bir dal oluşturup o dal üzerinde projede çeşitli değişiklikler yapıldığını varsayalım. Eğer projede checkout komutu ile değişikliklerin yapıldığı dala geçilmezse varsayılan dal da bu değişiklikler görünmeyecektir.**

1. **‘diff’**

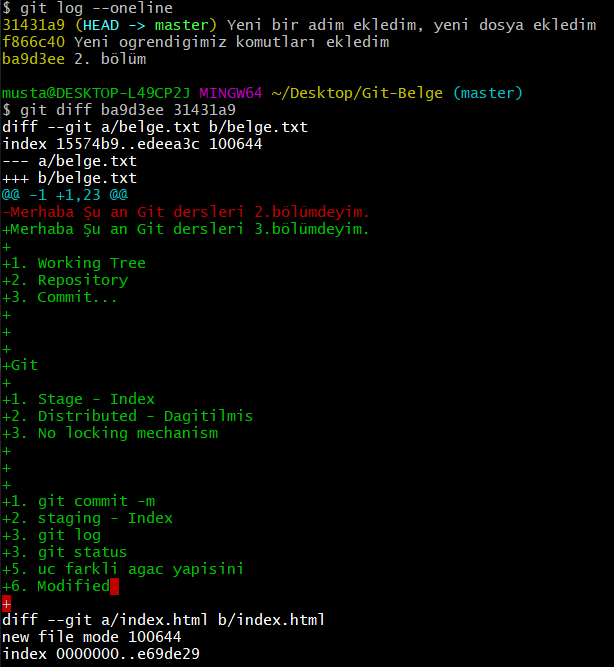
Birleştirme işlemine geçmeden önce iki dal arasındaki farkları nasıl inceleyebileceğimizi öğrenelim. Bunun için git diff komutunu kullanabiliriz.

Etc:

git diff belgede yapılan değişiklikleri gösterir. Ama değişikliklerden sonra git add komutu verilmişse git diff takip edilen dosyalarda değişiklik görünmeyeceğinden herhangi bir değişiklik görülemez.

git diff –cached komutu ile repository ile stage edilen değişikleri karşılaştırıyor . Böylelikle git add komutu yapılmış olsa dahi değişiklikleri görebiliyoruz.

İki commiti karşılaştırmak için ;



1. **‘merge’**

Git’de dallar arasındaki commit’leri almanın birden fazla yolu var. Bu yollardan en temeli bir dalı, diğer dala birleştirme yani merge işlemidir.

Etc:

git merge chapter2

***Branch Silmek***

* git branch –d <branchadi>

Bilgisayar üzerinden branchi siler.

* Git push origin –-delete <branchadi>

Site üzerinden branchi siler.

***commit Silmek***

* git log –oneline yapılarak comit hashi öğrenilir.
* Git reset –-hard <hash>
* Girdiğimiz hash li committen sonraki tüm commitleri siler.
* Uzak depodaki commiti de silebilmek için ardından git push origin +master yazıyoruz.

Github Nedir?

Github, sürüm kontrol sistemi olarak Git kullanan yazılım projeleri için bir depolama servisidir.

Gitlab Nedir?

GitLab, Github’ın kullanıcılara sağladığı işlevlerin tamamını sunan bir Git servisidir. Açık kaynak (open source) projelerinizi bu servis üzerinde ücretsiz bir şekilde oluşturabilir ve yönetebilirsiniz. Gitlab daha çok firmalarda tercih ediliyor çünkü  Ücretsiz sürümünde kendi sunucularınızı kurarak sadece kurum içi kullanıcıların erişebileceği Gitlab servisi hizmeti bulunmaktadır.

Mustafa Pıçakçı

# Kaynakça

<https://www.mobilhanem.com/git-egitimi/>

<http://rogerdudler.github.io/git-guide/index.tr.html>

<https://medium.com/@halilozel1903/git-github-ve-gitlab-nedir-e51a87e8e4e7>