Git Version Control Sheet

Terminal Kullanımı

ls (list): mevcut konumdaki verileri listeler
pwd (print working directory): çalışılan yeri gösterir
cd (change directory) … **:** yer değiştirir
cd .. : mevcut konumdan çıkıp bir önceki konuma gider
clear: terminali temizler
mkdir (make directory) : klasör oluşturma
touch _ : dosya oluşturma
rm (remove): dosya silme
rm -rf: klasör silme
git : yardım dokümantasyonu
git –version: versiyon listeleme

Kullanıcı Adı ve E-Mail Kaydetme

git config –global user.name "name" : kullanıcı adı kaydetme
git config –global user.email "email" : email kaydetme
git config user.name : kullanıcı adını görüntüleme
git config user.email : emaili görüntüleme

.gitignore: gizli olması istenilen dosyalar staging ile repo'da olsun istenilmez. ".gitignore" klasörü oluşturulur. gösterilmesi istenmeyen dosyalar .gitignore içerisine atılır.

Git Ekleme Aşamaları

branch: dal
commit: aşamalar
git status: git'in güncel durumunu gösterir.
git init: klasörü git'e bağlar.
git add . : çalışma klasörünü staging index'e atar.
git commit: local repo'ya işlemleri atar. Ayarlanmış text editörde commit mesajı yazılmasını talep eder.
git commit -m" ": terminalde doğrudan commit yapılır. Commit mesajı " " içerisine yazılır.
git log: log'ları gösterir.
<! – 2 kez init kodu yazma –!>
<! – ilk önce git status çalıştır mutlaka –!>

ls -la: gizli işlemleri gösterir.
rm -rf.git : git'i silme

Branch İşlemleri

HEAD: git'te nerede olduğunu gösterir. Genelde son commit'i gösterir fakat yer değişikliği de mümkün olduğundan farklı konumlarda da olabilir.

git branch: güncel branch'leri gösterir.
git branch branchName : yeni branch oluşturur.
git switch branchName : branch'ler arasında yer değiştirme.

q enter: satırdan ve end sınırından çıkmak için kullanılır.
üst/alt ok tuşları ile önceden yazılmış kodlar arasında dolaşılır.

git merge branchName: merge işlemi branch'leri birleştirmede kullanılır. Önce eklenecek branch'e geçilir. Ardından merge komutu ile eklenmesi istenilen branch adı yazılır.

Fast Forwarding: commitler master üzerinden yapılır.
Merge Conflict: aynı dosya içerisinde değiştirme olduğunda, farklı dosyaları silme veya değiştirme olduğunda ortaya çıkabilir.
çözüm: git add . + commit yapılır.

STASH

Stash: git'te silinen bilgi tamamen silinmez. Stash'e aktarılır. Böylece silinen bilgi geri yüklenmek istendiğinde erişilebilir.
git stash list: stash'teki verileri gösterir.
git stash pop: stash'ten bilgileri geri alır/çıkartır.
git stash apply stashname: sadece seçilen stash geri alınır.
git stash clear: stash'leri siler.
git stash apply: stash'te tek bir veri varsa stash'te pop işlemi uygulamadan stash'teki bilgiyi geri alır.

Detached Head: head ile master farklı konumlarda olduğunda ortaya çıkan durumdur.
çözüm: git switch master
yeni branch oluştur
Silme ve Geri Dönme
git reset: geri dönerken silme işlemidir. Sadece commitler silinir.
git reset –hard: hem içerikler hem commitler silinir.
revert: commitler silinmez, log'da tutulur.değişiklikler üzerinden aynı branch'te yeni commitler ile devam edilir.
git restore: merge ile geri alma.
git checkout commitName: eski commit'e dönülür.
reset'te commitler silinirken revert'te commitler silinmez.

git diff commitName_commitName: branch, commit, dosyalar, klasörler arasındaki farklılıklar gösterilir.
HEAD'de de yazılıp bakılabilir.
syntax hatası olursa boşluk yerine iki nokta kullanılır.

Rebasing

Rebasing: Master ile branch arasında merge yapılır. Merge sonrası branch'te "merge commit"ler oluşur. "rebase" ile merge commitler ortadan kalkar ve master'daki commitlerin önüne branch'teki commitler eklenir.

Commitler hizalanır. Commit sırası yani kronoloji bozulur, aslında git'teki tarihi değiştiriyoruz.

Log temizlemek ve tarihi tekrar yazmak için kullanılır.

<!-- Dikkat edilmesi gereken şey rebase yapılırken dosyaların paylaşımının yapılmamış olması lazım. Conflictlere veya başka problemlere sebep olabilir **--!>**

Remote İşlemler

-u : upstream.

git push	
git push origin master	git'e pushlama işlemi
git push -u origin master	

git remote: origin'de miyiz diye kontrol eder.
pull request (pr): repo sahibinden merge'leme yaptırmak ,değişikleri uygulamak, için izin istenir. Kontrol sağlar.

git branch -r: remote branch'i gösterir.
git fetch: git'teki değişiklikleri alıp getirir. Commit işlemi değildir. Sadece değişiklikleri gösterir. Bilgileri dahil etmek istiyorsak ek olarak merge işleminin yapılması gerekir.

git pull: değişiklikleri entegre eder. fetch'ten farkı sadece göstermez. **<!-- git pull = git fetch + merge --!>**
git pull origin master
! remote branch ile local branch arasında dolaşmak için switch yerine checkout kullanılır.

git clone urlName: bir GitHub repo'sunu otomatik olarak indirir.
fork: Farklı bir GitHub repo'sunu kendi GitHub repolarımıza ekleriz. Eklenen repo'nun asıl sahibi ile bağı kalmaz.

Clone ve Fork Farkı

Clone işleminde indirilen repo'nun hala repo sahibi ile bağı vardır. Ortak çalışma denilebilir. Değişikliklerin uygulanması

için repo sahibinden izin alınır. Fork’da ise alınan repo’nun bağı kalmaz.