

# Temel Git terimleri ve komutları

Git, yazılım geliştirme süreçlerinde kullanılan, hız odaklı, dağıtık çalışan bir sürüm kontrol ve kaynak kod yönetim sistemidir.

*Genel olarak baktığımızda Git'i bilmenin bir programlama dili bilmek kadar hatta daha fazla önemli olduğunu söyleyebiliriz. Bu yazıda Git içerisinde temel terimlere ve komutlara özet şeklinde değineceğiz.*

**Repository** Proje dosyalarını uzak bir sunucuda depolar. Genel kullanımda "Repo" olarak da kısaltılır.

**Branch** Projenin bir çok bölümünü derli toplu şekilde tutulmasını sağlar. Branchin avantajı ana branchteki(master) geliştirme yapısını etkilemeden projeyi diğer oluşturduğumuz branchler üzerinden yönetebiliriz.

**Master** Reponun ana branchidir. Git üzerinde hiç bir işlem yapmazsak değişiklikler master branchi üzerinden devam eder.

**Commit** Proje dosyalarınızda belli bir değişiklik yaptığınızda o değişikliğin anlık görüntüsünün alınıp kaydedilmesine denir.

**Checkout** Mevcut branch üzerindeki commit'lerin üzerinde geçiş yapabilmemizi sağlar.

**Fork** Repository'nin bir kopyasının alınmasıdır. Temel anlamda forkun kullanım amacı alınan bu kopya üzerinde değişiklikler yaptıktan sonra projenin ana reposuna gönderilerek projenin gelişmesine katkı sağlamaktır.

**Pull Request (PR):** Fork edilen proje üzerinde değişiklikler yaptıktan sonra gerçek repository'e gönderilerek o projenin sahibi olan geliştiricinin değerlendirmesine sunmaktır. Eğer PR kabul edilirse ana repository üzerinde, fork ettiğiniz proje üzerinde değişiklikler işlenir.

**Merge** Branch üzerinde yaptığımız değişiklikleri master branch'i üzerinde birleştirme işlemidir.

*Şimdi ise komutlara göz atalım.*

## **git config**

Kullanıcı adı, mail gibi ayarların belirleneceği komuttur. Örnek  
`git config --global user.email sam@google.com`

## **git init**

Bu komut ile proje dizininizde GIT dizinini oluşturur. Bu dizinde projenizin repo adresi, projenin akışı, bilgileri gibi veriler bulunur.

## **git add**

Verilen parametrelere göre o dosyaları dizine ekler ve commit'lemeye hazır hale gelirler. "Git add ." komutu mevcut dizindeki tüm dosyaları dizine ekler. "Git add dosya.txt" komutu mevcut dizindeki "dosya.txt" dosyasını dizine ekler.

```
git add dosya.txtgit add .
```

## **git rm**

git add komutunun tersi olarak belirttiğiniz dosya veya dosyaları çalışma dizininden siler.

```
git rm dosya.txt
```

## **git commit**

git commit -m "ilk commit" komutu çalıştırdığımızda "ilk commit" başlığıyla o anki çalışma dizinindeki dosyaları .git içindeki özel bir bölüme(head) ekler.

```
git commit -m "ilk commit"
```

## **git status**

Proje dosyalarının o anki durumu hakkında bilgi verir. Durumu değiştirilmiş dosyaları gösterir.

## **git remote**

"git remote -v" komutu ile projeye bağlanan uzak sunucuları listeler. Mevcut projeyi uzak sunucuya eklemek için ise "git remote add" komutunu çalıştırabiliriz.

```
git remote add origin https://github.com/username/project.git
```

Yukarıdaki komuttaki Github üzerinde belirttiğimiz repository'nin projeye eklemesini sağlarız.

## **git push**

Commit'lediğimiz dosyaları uzak sunucudaki repository'e gönderir.

```
git push origin master
```

## **git pull**

Uzak sunucudaki proje dosyaları üzerindeki bir değişiklik veya ekleme varsa onları bizim localimizdeki proje dosyaları ile birleştirir.

Diğer komutlara ise aşağıdaki linkten ulaşabilirsiniz.

---

İkinci bir kaynak

# **Git Komutları ve Kullanımı**

Günümüzde ortak projeler geliştirmek, açık kaynaklı kişisel kod geliştirmek için bir çok kişi git tercih ediyor. Bu kaynak kontrol aracı ile geliştirilen projeler host edilmesi için GitHub, GitLab platformlarında oldukça popüler olarak kullanılmaktadır.


2 Sebepden dolayı kaynak kontrol sistemine ihtiyaç duyarız;


- Kaynak kodunuzun değişimlerini takip etmek ve bunu yönetebilmek

- Birlikte kod geliřtirmek.

Git bu ihtiyaları karřılamak hazır bir takım fonksiyonlara sahip. Ařağıda jrebel hazırlamıř olduėu grselde bu fonksiyonları grebilirsiniz.

## 1. Git Komutları

**Git** Cheat Sheet



### Create a Repository

From scratch -- Create a new local repository  
`$ git init [project name]`

Download from an existing repository  
`$ git clone my_url`

### Observe your Repository

List new or modified files not yet committed  
`$ git status`

Show the changes to files not yet staged  
`$ git diff`

Show the changes to staged files  
`$ git diff --cached`

Show all staged and unstaged file changes  
`$ git diff HEAD`

Show the changes between two commit ids  
`$ git diff commit1 commit2`

List the change dates and authors for a file  
`$ git blame [file]`

Show the file changes for a commit id and/or file  
`$ git show [commit]:[file]`

Show full change history  
`$ git log`

Show change history for file/directory including diffs  
`$ git log -p [file/directory]`

### Working with Branches

List all local branches  
`$ git branch`

List all branches, local and remote  
`$ git branch -av`

Switch to a branch, my\_branch, and update working directory  
`$ git checkout my_branch`

Create a new branch called new\_branch  
`$ git branch new_branch`

Delete the branch called my\_branch  
`$ git branch -d my_branch`

Merge branch\_a into branch\_b  
`$ git checkout branch_b`  
`$ git merge branch_a`

Tag the current commit  
`$ git tag my_tag`

### Make a change

Stages the file, ready for commit  
`$ git add [file]`

Stage all changed files, ready for commit  
`$ git add .`

Commit all staged files to versioned history  
`$ git commit -m "commit message"`

Commit all your tracked files to versioned history  
`$ git commit -am "commit message"`

Unstages file, keeping the file changes  
`$ git reset [file]`

Revert everything to the last commit  
`$ git reset --hard`

### Synchronize

Get the latest changes from origin (no merge)  
`$ git fetch`

Fetch the latest changes from origin and merge  
`$ git pull`

Fetch the latest changes from origin and rebase  
`$ git pull --rebase`

Push local changes to the origin  
`$ git push`

### Finally!

When in doubt, use git help  
`$ git command --help`

Or visit <https://training.github.com/> for official GitHub training.

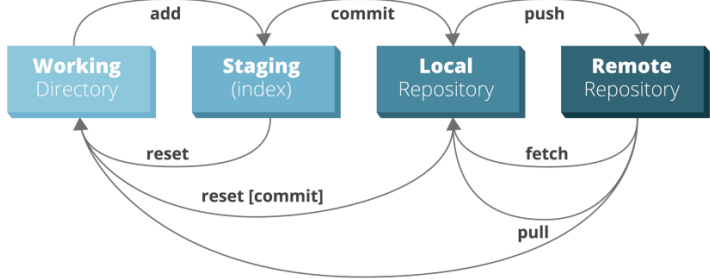


image from <https://www.jrebel.com/blog/git-cheat-sheet>

Bir klasr oluřturup ierisinde  
**git init**

fonksiyonu alıřtırdığımızda **.git** altında ufak bir dosya ynetim sistemi oluřturduėunu grebilirsiniz.

Bu klasörün içerisinde dosyalarınızın state(versiyonlarını) tutacaktır.

- Bu dosyalar arasındaki farklar,
- **commit** sırasındaki kesitler

hepsi bu dosyalar içerisinde tutulmaktadır.

```
onurs-MacBook-Pro:gitless ondayibasi$ ls -al
total 8
drwxr-xr-x  4 ondayibasi  staff  136 Feb 14 11:16 .
drwxr-xr-x 39 ondayibasi  staff 1326 Feb 14 11:04 ..
drwxr-xr-x 13 ondayibasi  staff  442 Feb 14 11:16 .git
-rw-r--r--  1 ondayibasi  staff  578 Feb 12 20:41 .gitignore
onurs-MacBook-Pro:gitless ondayibasi$ cd .git
onurs-MacBook-Pro:.git ondayibasi$ ls -al
total 40
drwxr-xr-x 13 ondayibasi  staff  442 Feb 14 11:16 .
drwxr-xr-x  4 ondayibasi  staff  136 Feb 14 11:16 ..
-rw-r--r--  1 ondayibasi  staff   23 Feb 12 20:41 HEAD
drwxr-xr-x  2 ondayibasi  staff   68 Feb 12 20:41 branches
-rw-r--r--  1 ondayibasi  staff  312 Feb 12 20:41 config
-rw-r--r--  1 ondayibasi  staff   73 Feb 12 20:41 description
drwxr-xr-x 12 ondayibasi  staff  408 Feb 12 20:41 hooks
-rw-r--r--  1 ondayibasi  staff  206 Feb 14 11:07 index
drwxr-xr-x  3 ondayibasi  staff  102 Feb 12 20:41 info
drwxr-xr-x  4 ondayibasi  staff  136 Feb 12 20:41 logs
drwxr-xr-x 24 ondayibasi  staff  816 Feb 14 09:49 objects
-rw-r--r--  1 ondayibasi  staff  107 Feb 12 20:41 packed-refs
drwxr-xr-x  5 ondayibasi  staff  170 Feb 12 20:41 refs
onurs-MacBook-Pro:.git ondayibasi$ █
```

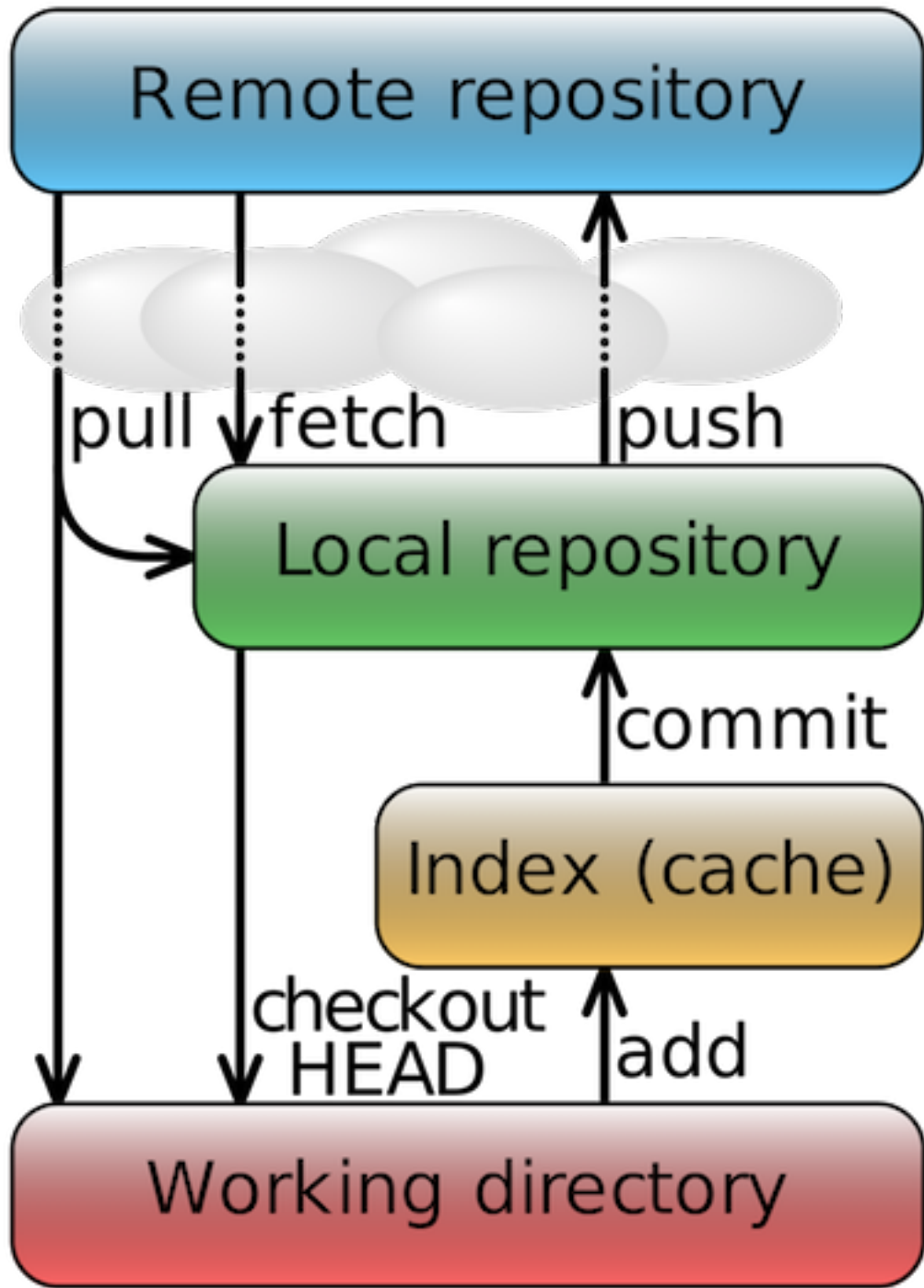
klasörde yer alan git dosyası ve .git içerisinde neler bulunur

```
HEAD
branches
config
description
hooks
index
logs
objects
packed-refs
refs
```

**.gitignore** dosyası git dosya sistemine katılmasını istediğiniz binary vb dosyalar için kullanılır. Örnek istenmeyen dosya türlerini bu [linkinden](#) erişebilirsiniz.

## 2. Git Veri Akışı

Aşağıdaki resimden git veri akışının nasıl gerçekleştirildiğini görebilirsiniz. Aşağıdaki komutlarla bunun nasıl çalıştığına bakalım.



### 3. Untracked Files (İzleri Oluşturulmamış Dosyalar)

a1.txt ve a2.txt dosyalarını çalıştığınız klasöre eklediğimizde bunları henüz izleri oluşturulmamıştır.

```
git status
```

fonksiyonu ile izi oluşmamış(untracked) dosyaları görebilirsiniz.

```
onurs-MacBook-Pro:gitlesson odayibasi$ ls
onurs-MacBook-Pro:gitlesson odayibasi$ touch a1.txt
onurs-MacBook-Pro:gitlesson odayibasi$ touch a2.txt
onurs-MacBook-Pro:gitlesson odayibasi$ ls
a1.txt  a2.txt
onurs-MacBook-Pro:gitlesson odayibasi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
    a1.txt
    a2.txt
```

```
nothing added to commit but untracked files present (use "git add" to track)
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

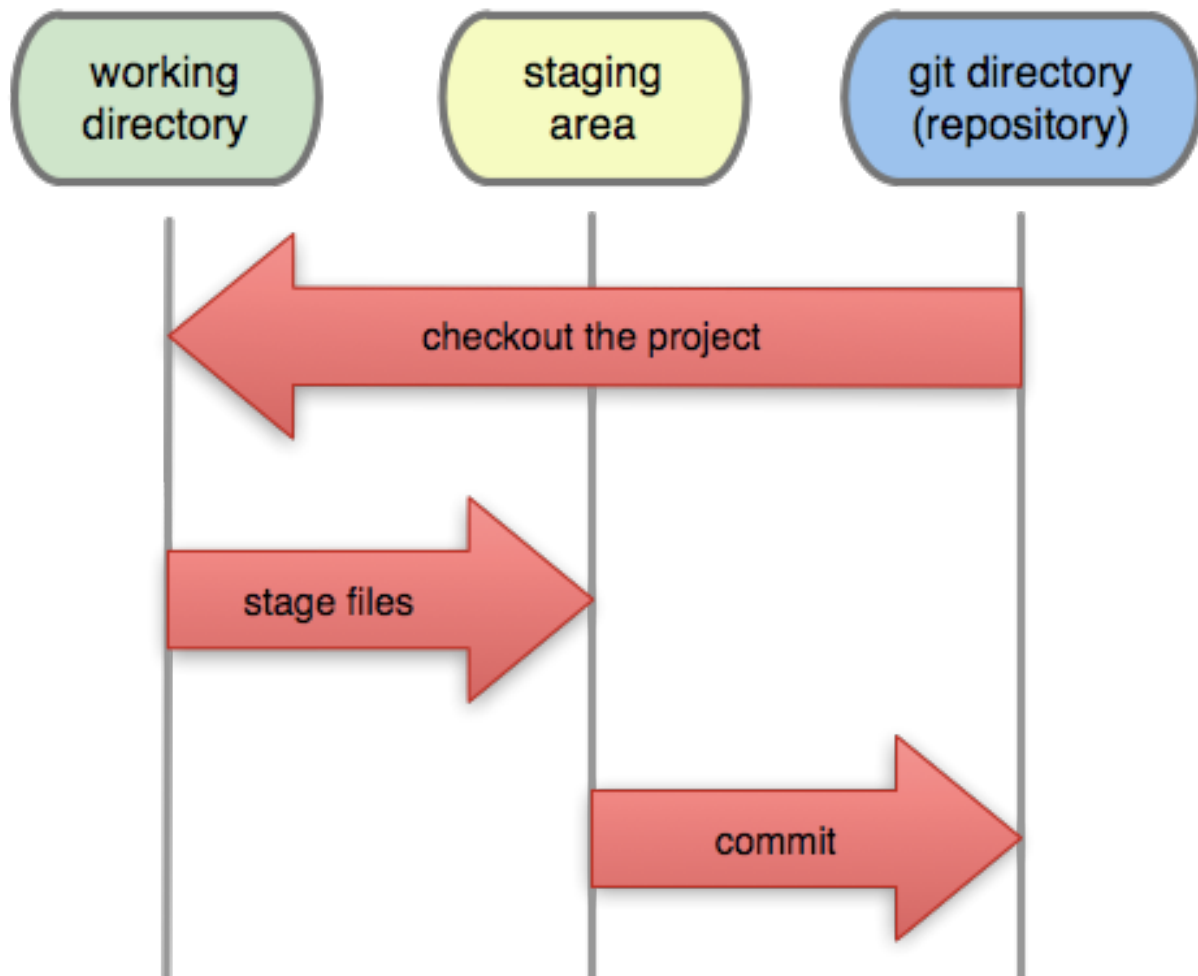
## 4. Staging Taşıma

```
git add
```

fonksiyonu ile a1.txt , a2.txt dosyasını **Staging Alanına** geçirmiş oluruz.



# Local Operations



## Git Local Operations

```
onurs-MacBook-Pro:gitlesson odayibasi$ git add a2.txt
onurs-MacBook-Pro:gitlesson odayibasi$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
new file:   a1.txt
new file:   a2.txt
```

```
onurs-MacBook-Pro:gitlesson odayibasi$ █
git add nasıl çalışır ?
```

İstediğiniz dosyayı **Staging Alanından** çıkarmak için  
**git reset HEAD <filename>**

komutunu kullanmanız yeterlidir.

## 5. Local (Kendi Çalışma Ortamınızaki) Dosyaları Değiştirmek

a1.txt dosyasında değişiklik yaptım

```
git status
```

dosyanın local tarafta modified olduğunu görebilirim.

```
git diff
```

ile de baktığımda dosyada hangi alanların değiştiğini görebilirim

```
git checkout -- a1.txt
```

ile yaptığınız değişikliği geriye alabilirsiniz.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

```
modified:   a1.txt
```

```
onurs-MacBook-Pro:gitlesson odayibasi$ git diff
```

```
diff --git a/a1.txt b/a1.txt
```

```
index 05544d8..5b381b6 100644
```

```
--- a/a1.txt
```

```
+++ b/a1.txt
```

```
@@ -1,1 @@
```

```
-a12sdsdfsd
```

```
+a12sdsdasdasdfsd
```

```
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

Dosyalar üzerinde Çalışmak

## 6. Commit (Save to Local Repo)

```
git commit -m 'msg'
```

komutu ile dosyalarınızı Local Repo'nuzda kaydedin..

```
onurs-MacBook-Pro:gitlesson odayibasi$ git blame a1.txt
```

```
00000000 (Not Committed Yet 2017-02-15 14:53:13 +0300 1) a12sdsdfsd
```

```
onurs-MacBook-Pro:gitlesson odayibasi$ git commit -m 'a1 a2 files added'
```

```
[master 6755f20] a1 a2 files added
```

```
2 files changed, 2 insertions(+)
```

```
create mode 100644 a1.txt
```

local repoya kayıt

## 7. Push (Save to Remote Repo)

```
git push origin master
```

komutu ile uzaktaki repo üzerine kayıt yapabilirsiniz.

```
[onurs-MacBook-Pro:gitlesson odayibasi$ git push origin master
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 600 bytes | 0 bytes/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/odayibasi/gitlesson.git
b0e5b0a..6755f20 master -> master
```

## 8. Pull (Get From Remote Repo)

```
git pull
```

Push yapmadan bir pull yapın çünkü başkaları Remote Repo'da değişiklikler yapmış olabilir önce değişiklikleri alıp sonrasında kendi değişikliklerinizi yapmanız gerekir.

```
[onurs-MacBook-Pro:gitlesson odayibasi$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/odayibasi/gitlesson
6755f20..21c32d6 master    -> origin/master
Updating 6755f20..21c32d6
Fast-forward
 README.md | 2 ++
 README.txt | 2 ++
 2 files changed, 4 insertions(+)
 create mode 100644 README.md
 create mode 100644 README.txt
onurs-MacBook-Pro:gitlesson odayibasi$
```

## 9. Conflict/Merge

Gerçek dünyada siz bir dosyada bir güncelleme yaparken başka birisi o dosyayı silmiş, ismini değiştirmiş olabilir. Bunun için bu farklılıkları çözmemiz gerekir. Oluşabilecek durumlar.

**1nci tip Sorun:** İki kişinin aynı dosyada değişiklik yapması .

**Çözüm:** Bendeki kaynağı kullan, Ondaki kaynağı kullan,

İkimizdekini mantıklı bir şekilde gözle birleştir.

**git pull**

komutunu çalıştırdığınızda CONFLICT oluşur. Dosyada

<<<<<HEAD olarak gösterilen kısımlar Local değişiklik ===

Remote'dan gelen değişiklikleri görebilirsiniz.

```
onurs-MacBook-Pro:gitlesson odayibasi$ git pull
Auto-merging a1.txt
CONFLICT (content): Merge conflict in a1.txt
Automatic merge failed; fix conflicts and then commit the result.
onurs-MacBook-Pro:gitlesson odayibasi$ cat a1.txt
<<<<<< HEAD
LocalDeğişiklik
sdsfssa12sdsdfs
=====
Remote Degisiklik
a12sdsdfs
>>>>>> c83626901e56ecc2b1c60949301a2c19bc836e0f
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

**2nci tip Sorun:** Birisinin çalıştığı dosyaları ,diğer kişinin silmesi

**Çözüm:** Bendeki kaynağı kullan, Ondaki kaynağı kullan..

Değişikliği yapıp direk push etmeye çalıştıgımda  
**(fetch first)**

uyarısını aldım.

```
onurs-MacBook-Pro:gitlesson odayibasi$ git push origin master
To https://github.com/odayibasi/gitlesson.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/odayibasi/gitlesson.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

2nci tip sorunda **AutoFix** işleminin yapılamadığını görebilirsiniz.

Bunun için kendiniz dosyayı local repo'dan çıkarıp tekrar commit yapmanız gerekir.

```
onurs-MacBook-Pro:gitlesson odayibasi$ git pull
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), done.
From https://github.com/odayibasi/gitlesson
  2e0e4ad..0881dfc master    -> origin/master
CONFLICT (modify/delete): a1.txt deleted in 0881dfc0f1e25303e071f0c9c6914c0f80ed8334 and modified in HEAD. Version HEAD of a1.txt left in tree.
Automatic merge failed; fix conflicts and then commit the result.
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

## 10. Değişikliklerin Tarihçesini ve Detayını Gör git logs

komutu ile değişikliklerin tarihçesini görebilirsiniz..

```
onurs-MacBook-Pro:gitlesson odayibasi$ git log
commit f3303553e0a2e87652c0797ca66a3c8b5b2b944c
Author: Onur DAYIBAŞI <onur.dayibasi@cs.com.tr>
Date:   Wed Feb 15 16:07:05 2017 +0300
```

aaa

```
commit 0af54e9dd855873a8adffb08760080f10a5e3599
Author: Onur DAYIBAŞI <onur.dayibasi@cs.com.tr>
Date:   Wed Feb 15 16:04:55 2017 +0300
```

a1.txt

```
commit 4fddfc15c83bf9d6840db43972105768bc944ed5
Merge: 847b780 37e7e38
Author: Onur DAYIBAŞI <onur.dayibasi@cs.com.tr>
Date:   Wed Feb 15 16:04:33 2017 +0300
```

a1.txt

```
commit 847b780bbdd17c2757bbe605ce46505c18881ecf
Author: Onur DAYIBAŞI <onur.dayibasi@cs.com.tr>
Date:   Wed Feb 15 16:03:12 2017 +0300
```

a1.txt

```
commit 37e7e384b6407f3624e641abe8369533a17f8ca4
Author: odayibasi <odayibasi@gmail.com>
Date:   Wed Feb 15 15:02:46 2017 +0200
```

Delete a1.txt

```
commit 2b77140ef53e0cf2f6ad2b9ba7d415898882f43f
Author: Onur DAYIBAŞI <onur.dayibasi@cs.com.tr>
Date:   Wed Feb 15 16:02:43 2017 +0300
```

a1.txt

## git shows

komutu ile ilgili comit içerisinde yapılan değişiklikleri görebilirsiniz..

```
onurs-MacBook-Pro:gitlesson odayibasi$ git show f3303553e0a2e87652c0797ca66a3c8b5b2b944c:a1.txt
sdsadasfsdff
```

## 11. Git Branch Oluşturma

Genelde projelerde master, **development**, **bugfix**, **release**, **poc** amaçlı dallar(branch) oluşturulur. Bu dalları oluşturmak git ile oldukça basittir .

## git branch

komutları ile branch lerinizi kolay bir şekilde yönetebilirsiniz..

```
onurs-MacBook-Pro:gitlesson odayibasi$ git branch bugfix
onurs-MacBook-Pro:gitlesson odayibasi$ git branch
  bugfix
* master
onurs-MacBook-Pro:gitlesson odayibasi$ git checkout bugfix
Switched to branch 'bugfix'
onurs-MacBook-Pro:gitlesson odayibasi$ git branch
* bugfix
  master
onurs-MacBook-Pro:gitlesson odayibasi$ git push origin bugfix
Username for 'https://github.com/odayibasi/gitlesson.git': odayibasi
Password for 'https://odayibasi@github.com/odayibasi/gitlesson.git':
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/odayibasi/gitlesson.git
 * [new branch]      bugfix -> bugfix
onurs-MacBook-Pro:gitlesson odayibasi$ git branch -d bugfix
error: Cannot delete branch 'bugfix' checked out at '/Users/odayibasi/Documents/github/gitlesson'
onurs-MacBook-Pro:gitlesson odayibasi$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
onurs-MacBook-Pro:gitlesson odayibasi$ git branch -d bugfix
Deleted branch bugfix (was d607d50).
onurs-MacBook-Pro:gitlesson odayibasi$ git branch
* master
onurs-MacBook-Pro:gitlesson odayibasi$ █
```

## Undoing Local Changes That Have Not Been Committed

If you have made changes that you don't like, and they **have not been committed** yet, do as follows:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run **git status** and you should see the affected file listed.
3. Run the following command, replacing **filename.html** with your file path (which you could copy and paste from the git status command):
  - **git checkout filename.html**
4. That file has now been reverted to the way it was at the previous commit (before your changes).

## Undoing (geri alma) a Specific Commit (That Has Been Pushed)

If you have one specific commit you want to undo, you can revert it as follows:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run **git status** and make sure you have a clean working tree.
3. Each commit has a unique hash (which looks something like **2f5451f**). You need to find the hash for the commit you want to undo. Here are two places you can see the hash for commits:
  - In the commit history on the GitHub or Bitbucket or website.
  - In your terminal (Terminal, Git Bash, or Windows Command Prompt) run the command **git log --oneline**
4. Once you know the hash for the commit you want to undo, run the following command (replacing **2f5451f** with your commit's hash):
  - **git revert 2f5451f --no-edit**
  - **NOTE:** The **--no-edit** option prevents git from asking you to enter in a commit message. If you don't add that option, you'll end up in the VIM text editor. To exit VIM, press **:** to enter command mode, then **q** for quit, and finally hit **Return** (Mac) or **Enter** (Windows).
5. This will make a new commit that is the opposite of the existing commit, reverting the file(s) to their previous state as if it was never changed.
6. If working with a remote repo, you can now push those changes:

- **git push**

## **Undoing Your Last Commit (That Has Not Been Pushed)**

If you made a mistake on your last commit and have not pushed yet, you can undo it. For example, maybe you added some files and made a commit, and then immediately realized you forgot something. You can undo the commit, and then make a new (correct) commit. This will keep your history cleaner.

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run this command:
  - **git reset --soft HEAD~**
  - TIP: Add a number to the end to undo multiple commits. For example, to undo the last 2 commits (assuming both have not been pushed) run **git reset --soft HEAD~2**
  - NOTE: **git reset --soft HEAD~** is the same as **git reset --soft HEAD^** which you may see in Git documentation.
3. Your latest commit will now be undone. Your changes remain in place, and the files go back to being staged (e.g. with git add) so you can make any additional changes or add any missing files. You can then make a new commit.

## **Undoing Local Changes That Have Been Committed (But Not Pushed)**

If you have made local commits that you don't like, and they **have not been pushed** yet you can reset things back to a



previous good commit. It will be as if the bad commits never happened. Here's how:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
  2. Run **git status** and make sure you have a clean working tree.
  3. Each commit has a unique hash (which looks something like **2f5451f**). You need to find the hash for the last good commit (the one you want to revert back to). Here are two places you can see the hash for commits:
    - In the commit history on the GitHub or Bitbucket or website.
    - In your terminal (Terminal, Git Bash, or Windows Command Prompt) run the command **git log --oneline**
  4. Once you know the hash for the last good commit (the one you want to revert back to), run the following command (replacing **2f5451f** with your commit's hash):
    - **git reset 2f5451f**
    - **git reset --hard 2f5451f**
    - **NOTE:** If you do **git reset** the commits will be removed, but the changes will appear as uncommitted, giving you access to the code. This is the safest option, because maybe you wanted some of that code and you can now make changes and new commits that are good. Often though you'll want to undo the commits and through away the code, which is what **git reset --hard** does.
-

# Uygulamali

```
mehmetaltuntas@MBP-von-Mehmet Eu3TestNGSelenium % git log --oneline
c97d4c2 (HEAD -> branch3) Changes are committed
e124969 (origin/new_work, origin/master, new_work, master) new changes added
e71de4d new changes added
4cfa8ad testNg UPDATED
mehmetaltuntas@MBP-von-Mehmet Eu3TestNGSelenium % git status
On branch branch3
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .idea/workspace.xml
        modified:   src/test/java/com/cybertek/tests/day14_extent_reports/ExtentDemoTest.java
mehmetaltuntas@MBP-von-Mehmet Eu3TestNGSelenium %
```