Git Eğitimi

•••

18 Haziran 2017

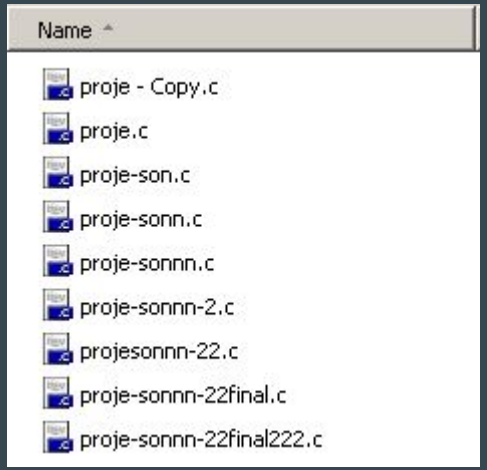
BİÇDA // Tasarım Atölyesi Kadıköy

Sürüm / Versiyon Kontrol Sistemleri (VCS) Nedir?

- bir ya da daha fazla dosya üzerinde yapılan değişiklikleri kaydeden,
- daha sonra belirli bir sürüme geri dönebilmenizi sağlayan bir sistemdir.

Neden VCS?

- Birlikte Uyumlu Çalışma
- (Düzgün) Sürüm Saklama
- Eski Sürüm Kurtarma
- Dosyaların Hayat Hikayesi
- Yedekleme



https://www.git-tower.com/learn/git/ebook/en/command-line/basics/why-use-version-control

DİKKAT

Versiyon Kontrol Sistemleri sadece yazılım amaçlı kullanılmıyor.

https://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding/

Sürüm Kontrol Sistemlerinin Gelişimi

Birinci Generasyon

- RCS
- SCSS

İşlem: Bir seferde tek dosya

Concurrency*: Locks

Networking: Yok

İkinci Generasyon

- CVS
- SourceSafe
- SVN
- TFS

İşlem: Çoklu dosya

Concurrency : Mer -> Com

Networking : Merkezi

Üçüncü Generasyon

- Bazaar
- Git
- Mercurial

İşlem : Değişim Setleri

Concurrency : Com -> Mer

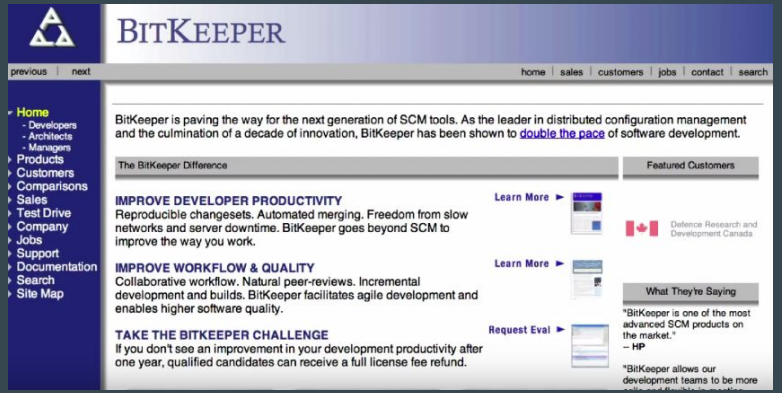
Networking : Dağıtık

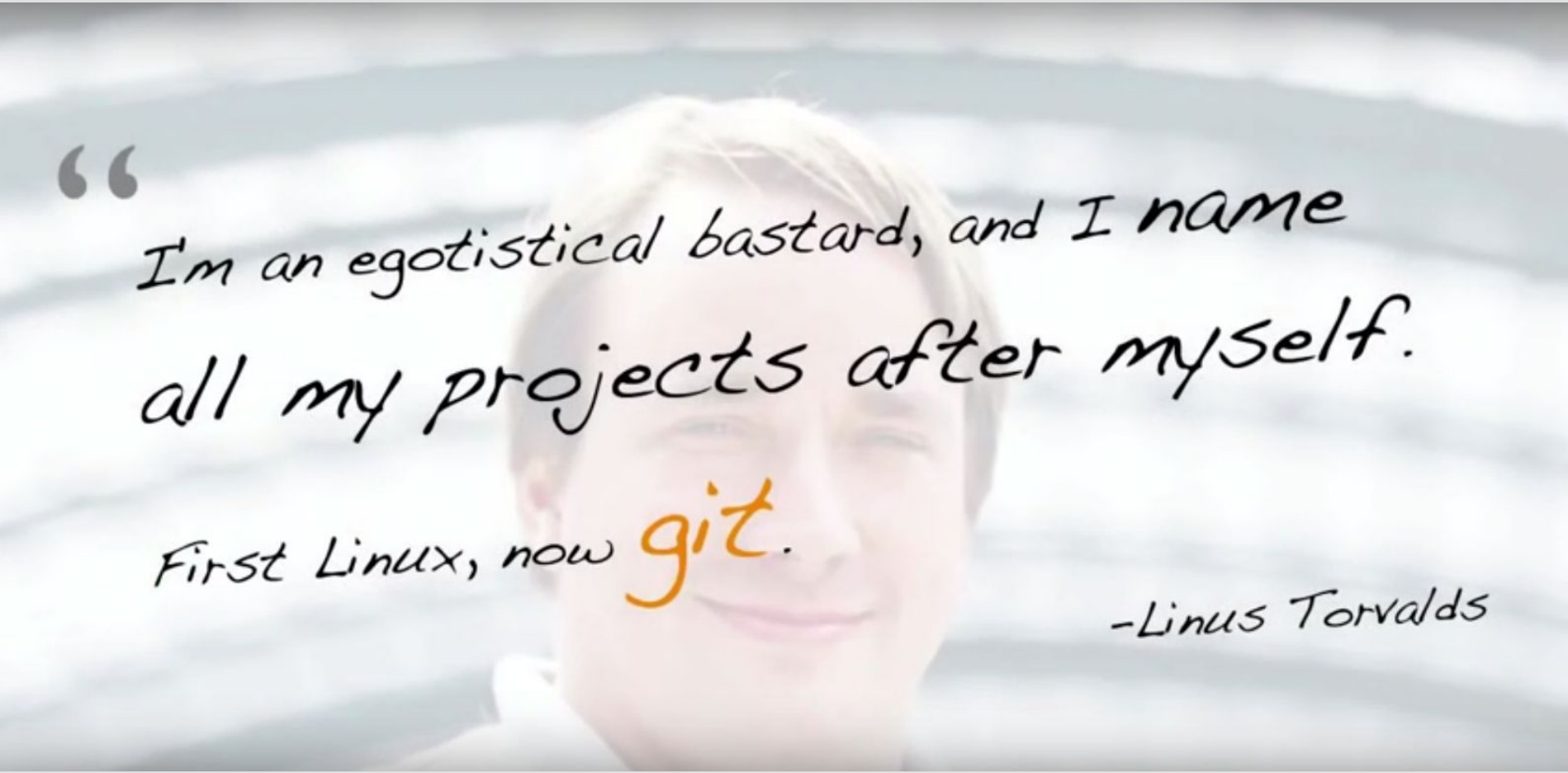
concurrency = aynı anda erişim

Git Nasıl Ortaya Çıktı?

Linux Kernel’inin Gelişimi

- BitKeeper isminde bir VCS ile yapılıyordu.
- Ticari bir yazılım.
- Kapalı kaynak kod.
- Dağıtık mimari.
- Lisansı Linux Kerneli geliştiricileriyle paylaşıldı.
- Ücretsiz geliştirme lisansı iptal edildi.
- https://lwn.net/Articles/130746/
- http://www.infoworld.com/article/2670360/operating-systems/linus-torvalds--bitkeeper-blunder.html




“
I'm an egotistical bastard, and I name
all my projects after myself.
First Linux, now git.
-Linus Torvalds

Neden Git?

- Performans*
- Güvenlik (SHA1 Hash)
- Esneklik
- Açık Kaynak

Git Depolamayı Nasıl Yapar?

- ~~Changeset~~
- ~~Difference~~
- Snapshots

Canlı Örnek #1

Git kurulumu & repository* oluşturma

*repository = kod havuzu

Git Kurulumu

Windows İçin:

- https://git-for-windows.github.io/

Linux ve Mac için:

Dağıtımlarda hazır kurulu geliyor.

Eğer kurulu değilse:

https://git-scm.com/

Terminal açıp **git --version** dediğinizde sonuç geliyorsa doğru yerdeyiz.

İlk Adımlar

Terminal uygulamasını çalıştırın:

- git config --global user.name nbgucer
- git config --global user.email gucer@itu.edu.tr

- git config --global core.eol native **Win**
- git config --global core.autocrlf true **Win**
- git config --global core.editor nano **Linux/Mac**
- git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe" **Win**
- git config --global color.ui auto

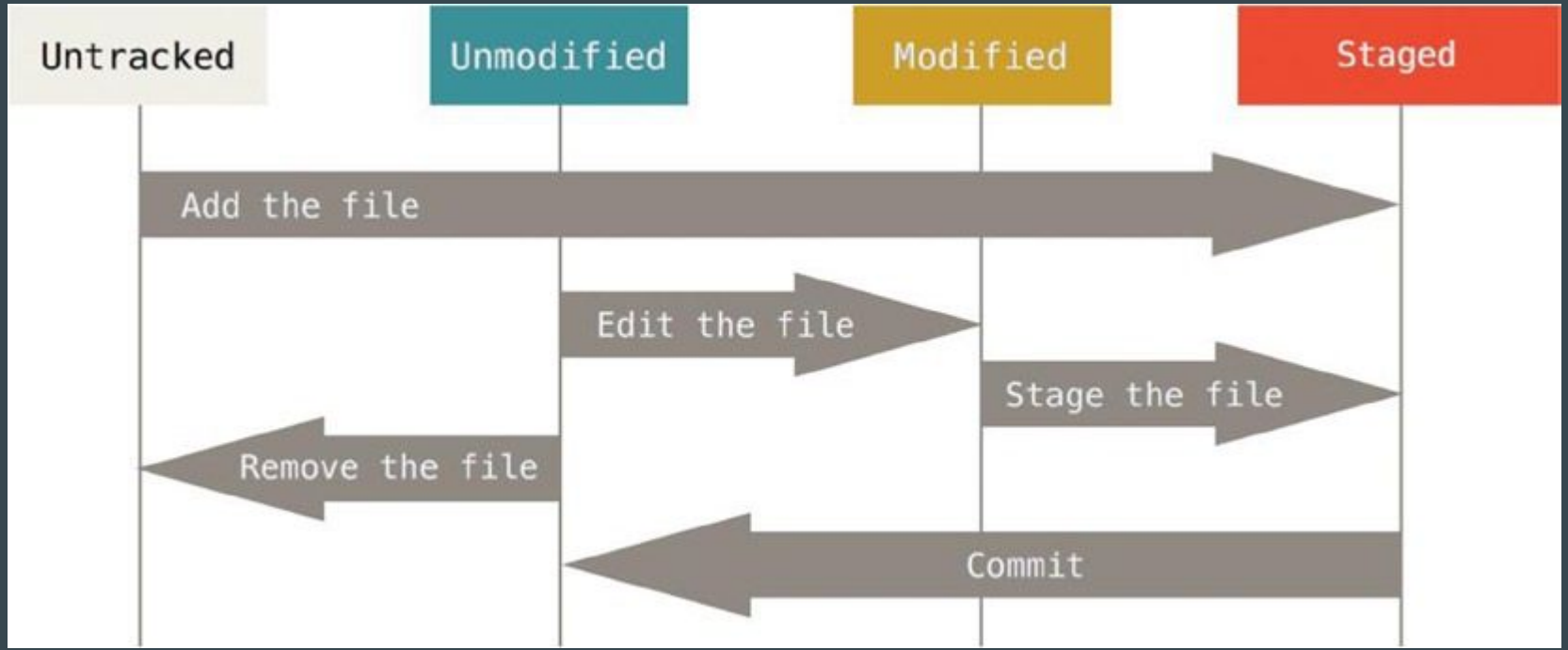
Repository Oluşturma

- Kullanıcınızın okuma/yazma yetkisi olduğu bir klasöre:
- **git init** komutu kullanarak, bulunduğunuz klasörü Git VCS'sine ekleyebilirsiniz.
- Yaptığınız tüm değişiklikler, **.git** isminde gizli bir klasör altında tutulacaktır.
- Çalışmanızdan Git'i kaldırmak mı istiyorsunuz? .git klasörünü silmeniz yeterli.

Clone Oluşturma

- git clone [url] {[klasör_adı]}

Git Dosya Aşamaları



Değişiklikler Yapmak

- git status
- git diff
- git diff --staged
- git add [dosya]
- git reset [dosya]
- git commit -m “[Açıklayıcı mesaj]”

Dosya Adı Yapılandırılması (Refactoring)

- git rm [dosya]
- git rm --cached [dosya]
- git mv [orjinal-dosya] [değişen-dosya]

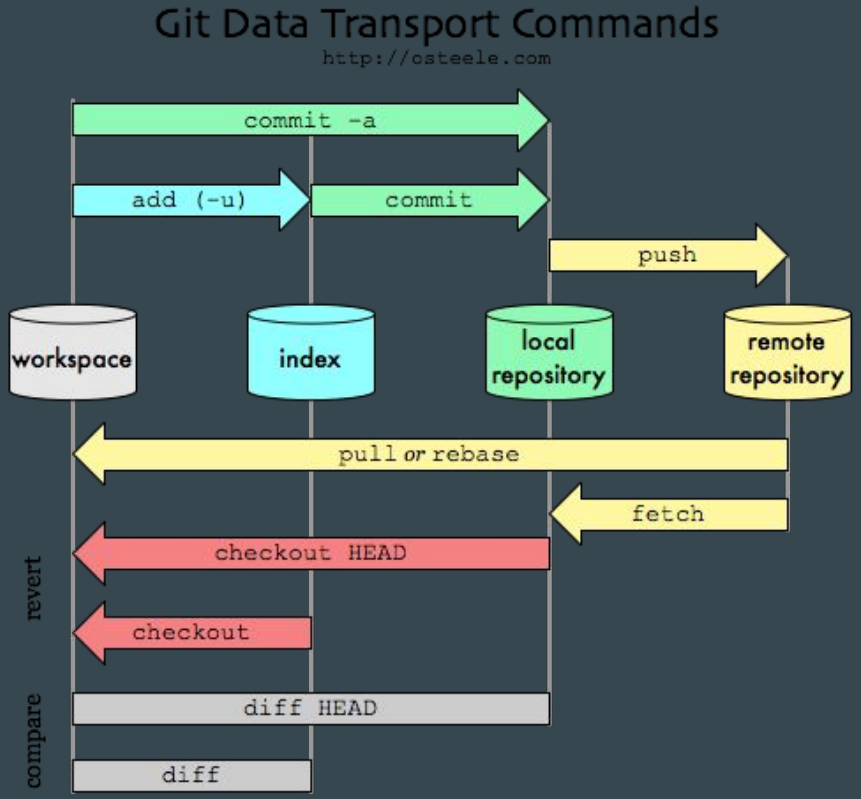
İstenmeyen Dosyaları Çıkartmak

.gitignore dosyası kullanarak kazara build amaçlı kullanılan binary ya da log dosyalarını git takip sisteminden hariç bırakabilirsiniz.

- *.log
- build/
- temp-*

git ls-files --other --ignored --exclude-standard komutu; proje boyunca ignore edilen bütün dosyaları gösterir.

Git Data Aktarım Komutları Özeti



Canlı Örnek #2

Branching

Git’te Branching

- git branch --all
- git branch [dal-adı]
- git checkout [dal-adı]
- git merge [mergelenecek-dal-adı]
- git branch -d [dal-adı]

Bir Yazılım Projesine Özellik Eklemek

- Projeden fork alınır. (branch oluşturulur)
- Özellikler tamamlanır.
- Master branch’e geçilir.
- Özellik Merge Edilir

Conflict / Uyuşmazlık Çözümü

Cannot Merge

You're going to pull some changes, but oops, you're not up to date:

```
git fetch origin git pull origin master From ssh://gitosis@example.com:22/projectname
* branch master -> FETCH_HEAD Updating a030c3a..ee25213 error: Entry 'filename.c' not
uptodate. Cannot merge.
```

Cannot Merge

So you get up-to-date and try again, but have a conflict:

```
git add filename.c
git commit -m "made some wild and crazy changes"
git pull origin master

From ssh://gitosis@example.com:22/projectname * branch master -> FETCH_HEAD
Auto-merging filename.c CONFLICT (content): Merge conflict in filename.c Automatic
merge failed; fix conflicts and then commit the result.
```

Çözümleme

So you decide to take a look at the changes:

```
git mergetool
```

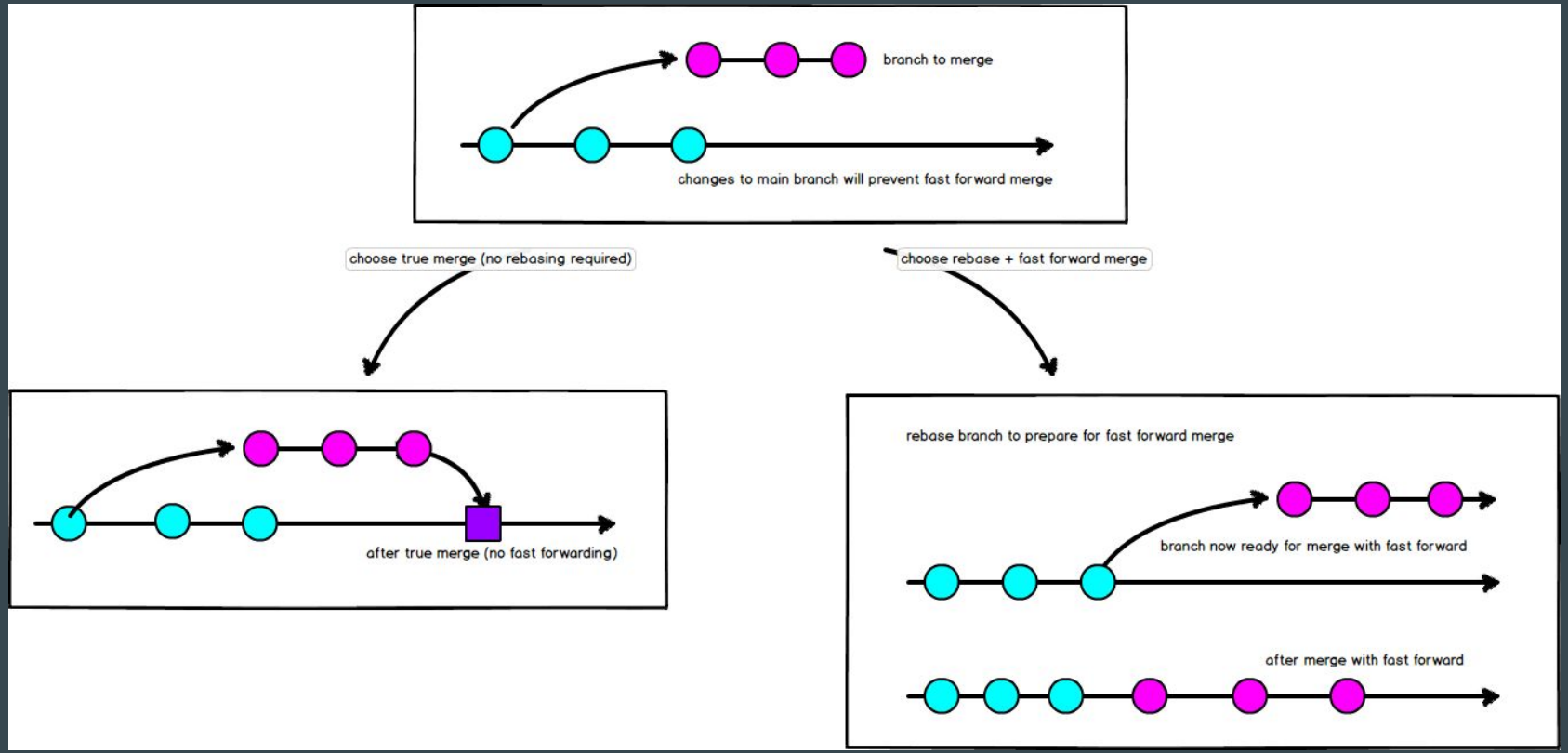
Oh me, oh my, upstream changed some things, but just to use my changes...no...their changes...

```
git checkout --ours filename.c
git checkout --theirs filename.c
git add filename.c git commit -m "using theirs"
```

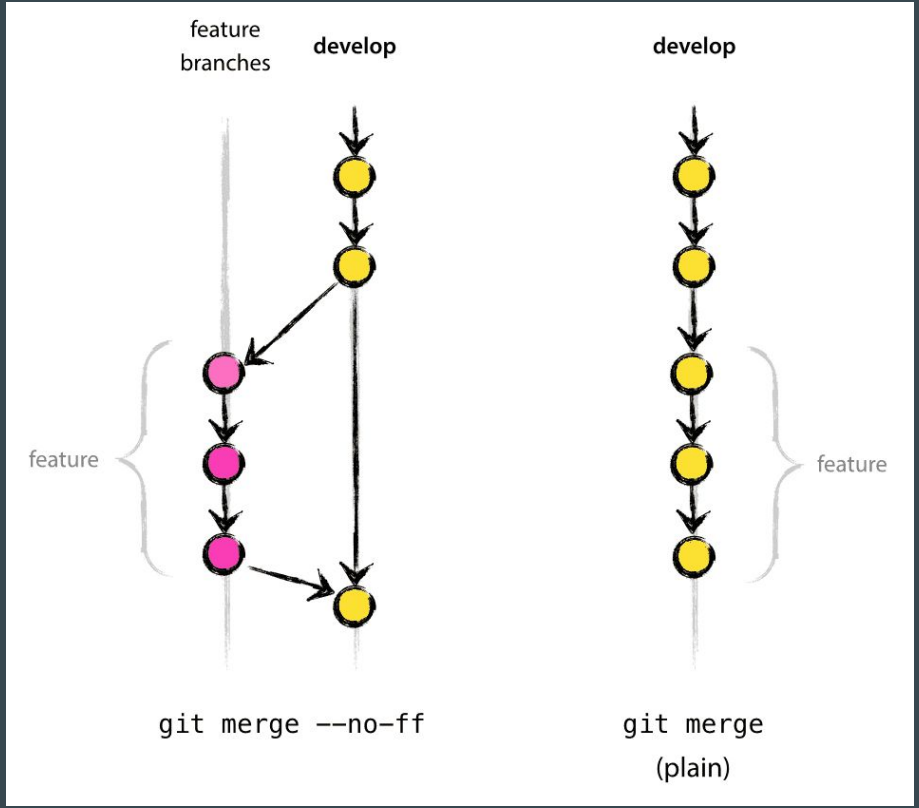
And then we try a final time

```
git pull origin master
From ssh://gitosis@example.com:22/projectname * branch master -> FETCH_HEAD Already up-to-date.
```

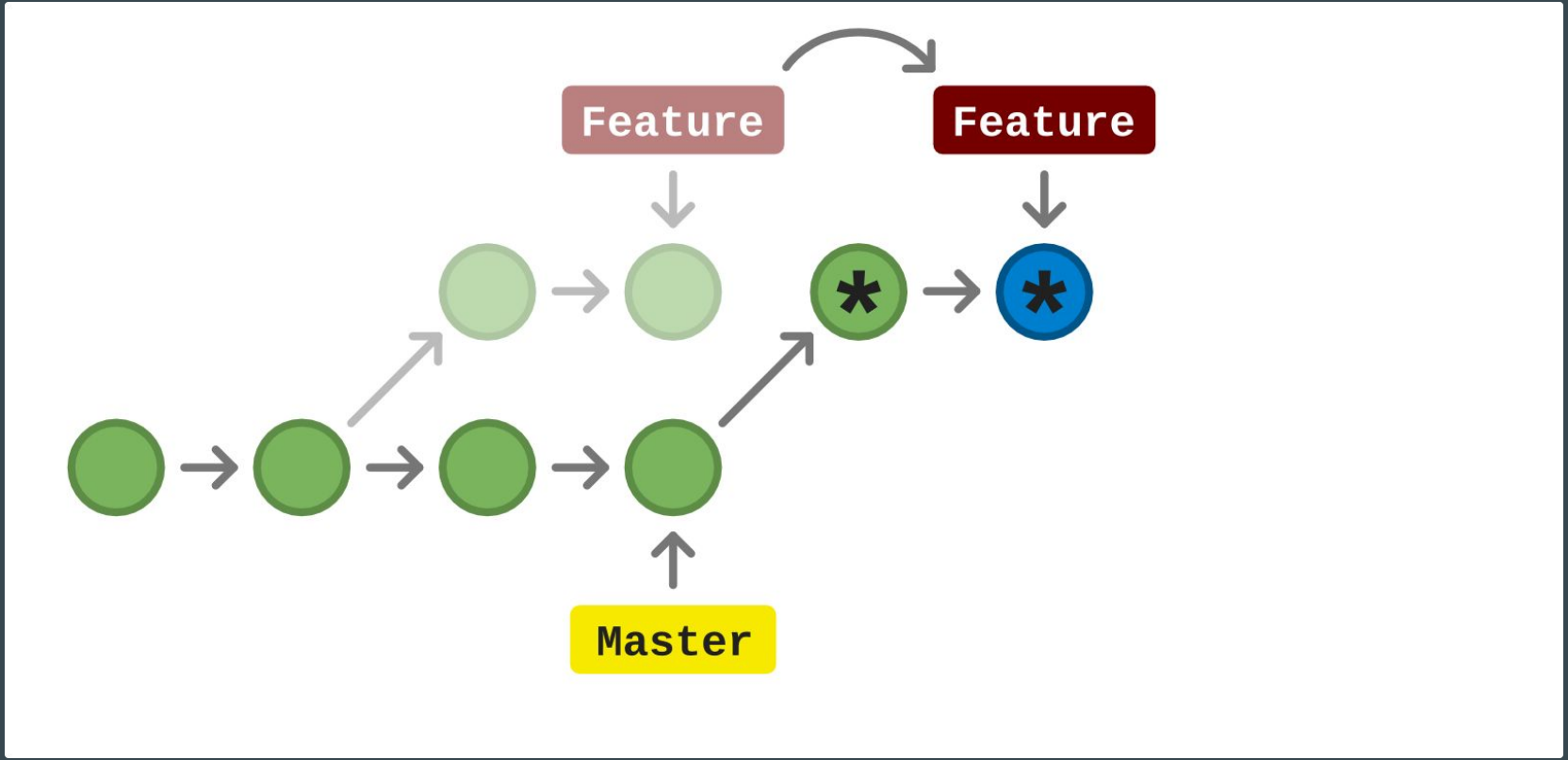
Merge


branch to merge
changes to main branch will prevent fast forward merge
choose true merge (no rebasing required)
choose rebase + fast forward merge
after true merge (no fast forwarding)
rebase branch to prepare for fast forward merge
branch now ready for merge with fast forward
after merge with fast forward

Merge


feature
branches
develop
develop
feature
feature
git merge --no-ff
git merge
(plain)

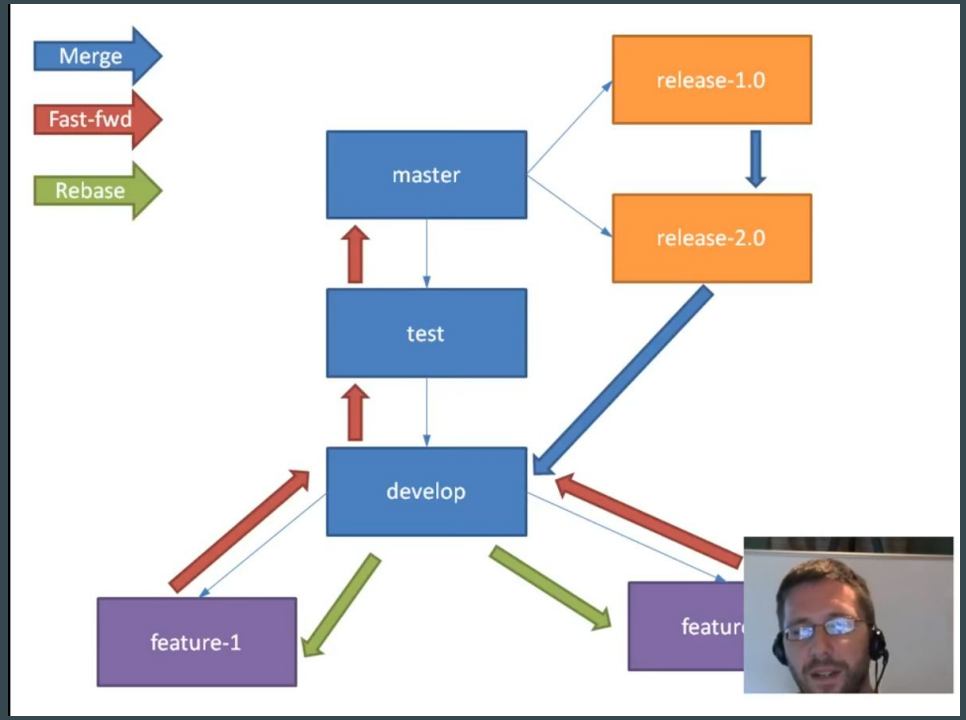
Rebase



Git Branchlama ve Merge'leme Teknikleri

- Full Cycle Strategy including Maintaining Multiple Releases
- Çok sayıda sürüm destekli tam döngü stratejisi

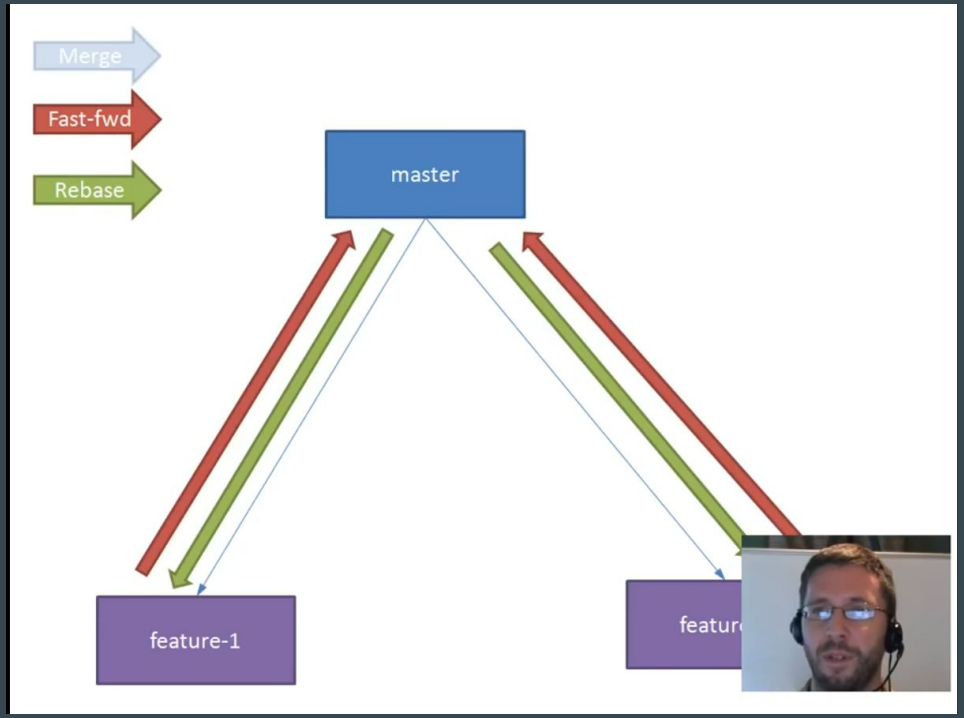
Kaynak: https://www.youtube.com/watch?v=to6tIdy5rNc



Git Branchlama ve Merge'leme Teknikleri

- Very Simple Small Team Strategy
- Basit ufak takım stratejisi

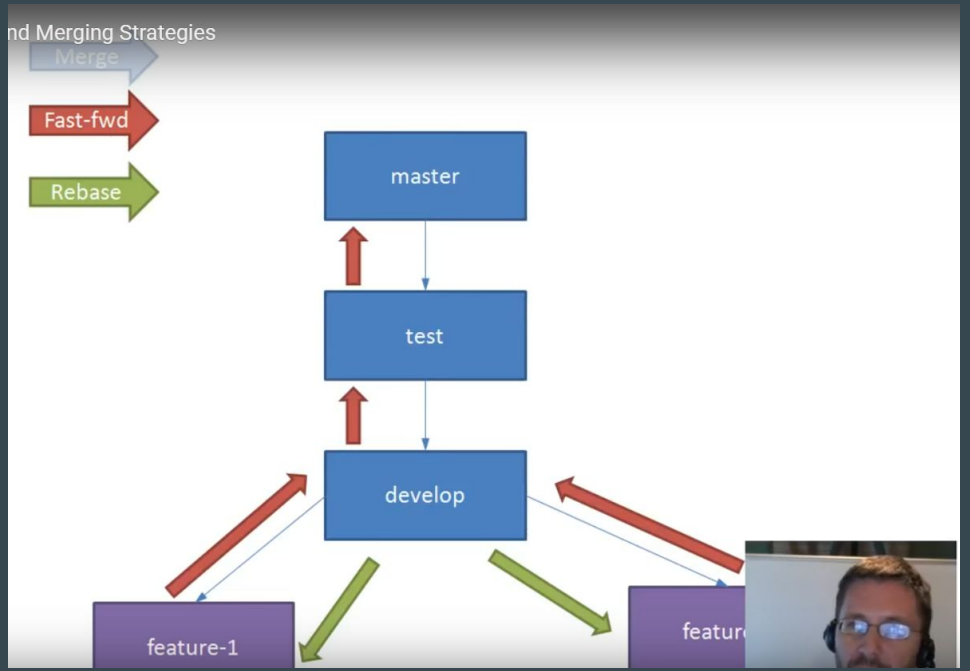
Kaynak: https://www.youtube.com/watch?v=to6tIdy5rNc



Git Branchlama ve Merge'leme Teknikleri

- Full Cycle Strategy With Only One Live Release
- Tek Canlı Sürümlü Tam Döngü Stratejisi

Kaynak: https://www.youtube.com/watch?v=to6tIdy5rNc



Canlı Örnek #3

GitHub üzerinden toplu alıştırma

1) Toplu contribution / pull request

https://github.com/nbgucer/git-egitim

1) Benim hesabımdan çatallayın (Fork)
2) Kendi repo'nuzdan clone'layın.
3) Yerel makinenizde değişiklik yapın.
4) Makinenizde değişikliği kaydedin.
5) Kendi Repo'nuza gönderin.
6) Benim Hesabımda Forklanan projeye değişikliğiniz ile ilgili 'Pull Request' gönderin.

GitHub Özellikleri

- Pull Request
- Code Review
- Web Interface / Desktop Interface
- Markdown Dokümantasyon

Diğer: https://github.com/features

Canlı Örnek #4

Tecrübe paylaşımı

Kaynaklar

Daha Fazla Git

- http://learngitbranching.js.org/
- https://www.codeschool.com/learn/git
- https://git-scm.com/book/tr/v1 (Kısmen Türkçe)
- https://www.atlassian.com/git/tutorials/why-git
- https://guides.github.com
- https://services.github.com/on-demand/
- https://help.github.com

Private Git Alanı İstiyorum

- https://about.gitlab.com/downloads/
- https://github.com/gitbucket/gitbucket
- https://www.linux.com/learn/how-run-your-own-git-server
- www.bitbucket.org

Kaynaklar

Açık Kaynak Yazılıma Katkıda Bulunmaya Nasıl Başlarım?

- https://github.com/MunGell/awesome-for-beginners
- https://opensource.com/life/16/1/6-beginner-open-source
- http://up-for-grabs.net/#/
- https://opensource.com/article/17/3/impact-github-software-career
- https://opensource.com/article/17/1/how-join-technical-community
- http://www.firsttimersonly.com

Teşekkürler