Git-Github

Version Control Tanım

- Dosyanın ya da dosyaların değişikliklerini kaydeden ve ileri zamanda bu kayıtlara (versiyonlara) geri dönmenize yarayan sistemlere Versiyon Kontrol sistemi denir.

- Bu kaydedilecek dosyalar kod olabileceği gibi herhangi farklı bir dosya da olabilir.

Neden önemsemeliyiz?

- Örneğin bir web ya da grafik tasarımcı olduğunuzu düşünün. Ve projenizin (resim, layout vb.) her versiyonunu tutmak istiyorsunuz. (Tutmalısınız da …)
- İşte bu tür seneryolarda VCS kullanmak çok akıllıca olacaktır.
- **Daha önce dosyalarınızın kaybolmaması için aldığınız önlemlerden örnek verin.**

Git’in Kısa Tarihi

- Açık kaynak projesi olan Linux kernelinin kapsamı oldukça büyük.
- Linux kernelinin ilk yıllarında yazılımındaki değişikler arşivlenmiş dosyalar olarak aktarılıyordu.
- Sonradan (2002) BitKeeper denen DVCS sistemi kullanmaya başladılar.
- Ancak 2005’te BitKeeper ücretsiz olan kullanım hakkını kaldırıp ücretli bir servise döndürdüğünden Linux geliştirme topluluğu (özellikle de Linux’ü geliştiren Linus Torvalds) kendi araçlarını BitKeeper’ı kullanırken öğrendikleri bilgilerle geliştirmeye başladılar.

Git'in Hedefleri

- Hız
- Basit Tasarım
- Lineer olmayan geliştirmeler için güçlü destek (binlerce branch)
- Tamamen dağıtık
- Linux kerneli gibi büyük projelerle etkili bir şekilde başa çıkabilecek (hız ve veri büyüklüğü)

Git Nedir?

- Git, verileri minyatür bir dosya sisteminin bir dizi **anlık görüntüsü (snapshots)** gibi düşünür.
- Gitle birlikte her dosyalarınızı «commit»lediğinizde ya da durumunu kaydettiğinizde Git sizin o zamandaki tüm dosyalarınızın bir fotoğrafını çeker ve kaydeder.
- Ve verimli olmak adına bir dosya üstünde değişiklik yapmadıysanız Git o dosyayı yeniden kaydetmez. Önceki durumdaki dosyayı referans almak yeterli olacaktır.

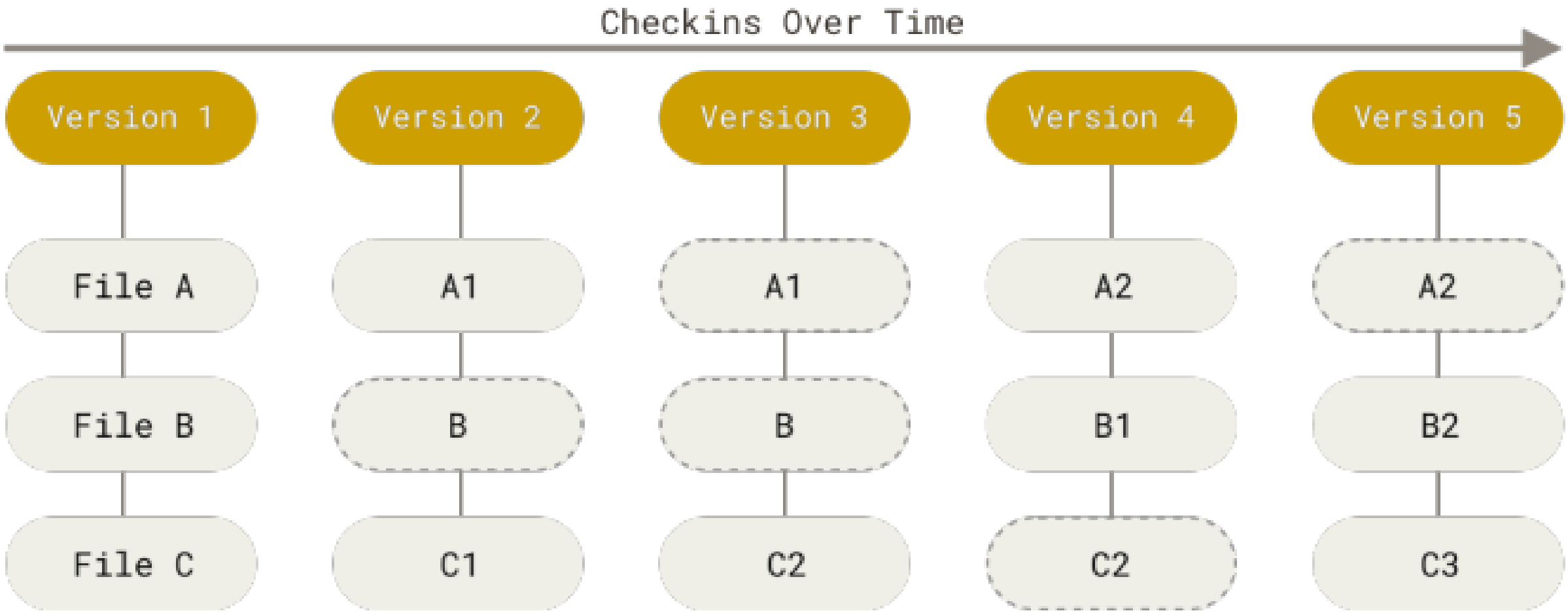


Figure 5. Storing data as snapshots of the project over time

İşlemlerin Lokalde Yapılması

- Git, çoğu işlemde sadece yerel dosyalara ve kaynaklara ihtiyacı vardır. Yani ağ üzerindeki başka bir bilgisayardan bilgi alışverişi yapmaya gerek yoktur.
- Bu da çoğu işlemlerin çok hızlı gerçekleşebilmesine olanak sağlar.

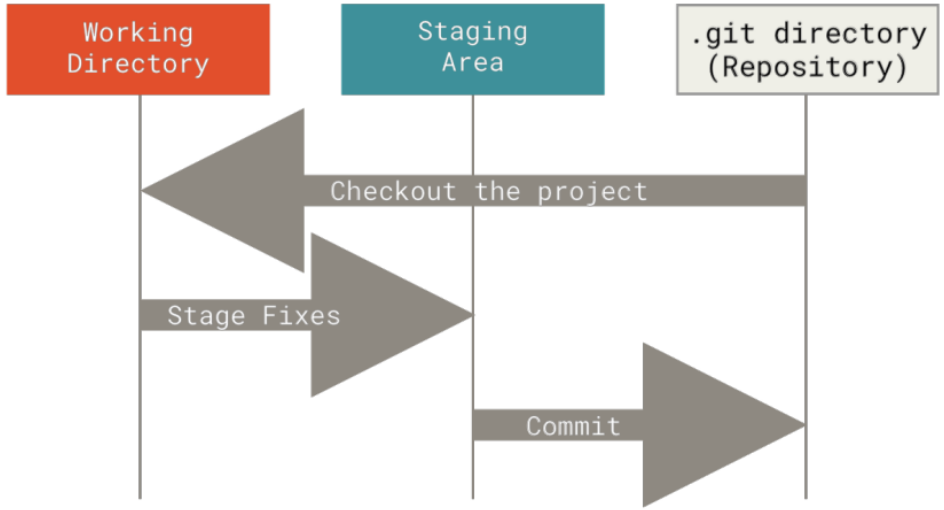
Bütünlük

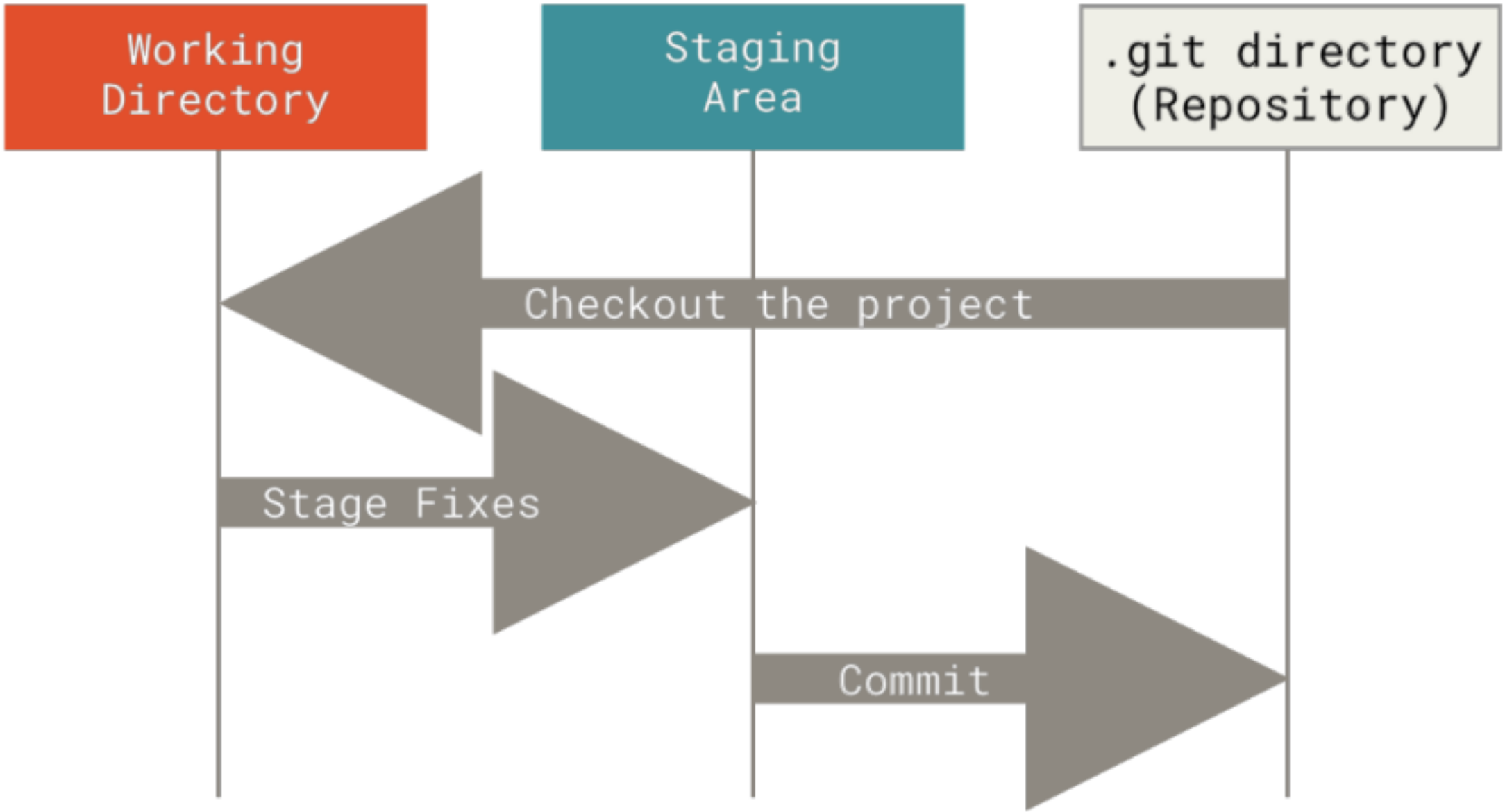
- Git’teki her şeyin depolanmadan önce sağlaması yapılır. Bu da Git bilmeden bir dosyanın ya da dizinin içeriğinin değiştirilmesinin imkansız olduğu anlamına gelir.
- Git bu «Checksum» işlemini SHA-1 hash algoritmasıyla yapar ve şuna benzer:

  24b9da6552252987aa493b52f8696cd6d3b00373
- Git veritabanında her dosyanın içeriğinin hash değerini tutar.
- Git’te veriyi geri getirilemeyecek şekilde silmek zordur.

3 Durum

- **Modified (Değiştirilmiş):** Dosya değiştirilmiş fakat veri tabanına henüz kaydedilmemiş
- **Staged:** Şuanki versiyonda değiştirilmiş dosya sonraki anlık görüntüye kaydedilmek üzere işaretlenmiş
- **Commited:** Veri başarılı ve güvenli şekilde yerel veri tabanında depolandı




Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

Command Line

- Git'i kullanmanın bir sürü yolu olsa da Git'in tüm kapasitesine ancak terminal üzerindeki komutlarla ulaşabiliyoruz.
- Grafik arayüz yazılımları kolaylık olması açısından kullanılabilir fakat her işimizi grafiksel olarak çözümeyiz.
- Grafiksel olarak da IDE'ler, Editörler ya da çeşitli farklı yazılımlar, Git'in kısmi komutlarını çalıştırarak kullanıcıya kolaylık sağlayabilir.

Kurulum-GNU/Linux

Debian:

$ sudo apt install git-all

Fedora-CentOS:

$ sudo dnf install git-all

Kurulum-macOS

Kurulum-Windows

https://git-scm.com/download/win

Config-GNU/Linux

- Git'in nasıl çalışacağına ilişkin değişkenleri tutan araç: *git config*
- Bu değişkenler linux'ta :

1. [path]/etc/gitconfig dosyası: Tüm kullanıcılar için geçerli değişkenleri tutar
2. ~/.gitconfig OR ~/.config/git/config dosyası: Bir kullanıcı için özelleştirilmiş değişkenlerdir.
3. .git/config dosyası: Spesifik repo için geçerli değişkenleri tutar.

Her level bir önceki levelin üstüne yazar.

Config-Windows

Config Dosyalarını bulmak ve ne içerdiklerini listelemek için:

```
$ git config --list --show-origin
```

Dosya Konumları

C:\Users\[USERNAME]

C:\Program Files\Git\etc\gitconfig

Sizin Kimliğiniz

- Git kurduğumuzda ilk yapmamız gereken bir kimlik oluşturmaktır. Bu önemlidir çünkü her Git commiti bu bilgiyi kullanır.

```
$git config --global user.name "John Doe"
$git config --global user.email johndoe@example.com
```

--global seçeneğini kullandığımızdan bunu çalıştığımız her makine için bir kez yapmamız yeterli olacaktır.

Editör Ayarlama

- Git bizden bir mesaj girmemizi istediğinde (commit yapacağımız sırada vb.) text editörü açacaktır. Bu editörün ne olacağını seçebiliriz.

Linux:

- `git config --global core.editor «emacs»/»nano»/»vim»…`

Windows:

- `git config --global core.editor "code --wait"`
- `git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"`

Branch İsmi Değiştirme

Varsayılan olarak Git, «master» isminde branchlar oluşturacaktır. Bunu değiştirebiliriz: (Github ‘main’ kullanır.)

- `git config --global init.defaultBranch main`

Ayarları Kontrol Etme

Çeşitli parametreler vererek bu kontrolü yapabiliriz. (ya da hepsi için daha önce kullandığımız -- list seçeneğini kullanırız.)

$ git config user.name

John Doe

Yardım Alma

```
$ git help [EYLEM]
$ git [EYLEM] --help
$ man git-[EYLEM]
```

Örnek:

```
$ git help config OR clone OR add …
```

Tüm EYLEM lere sadece `git help` komutu ile ulaşabilirsiniz.

Ayrıca detaylı bir dokümantasyon sayfası yerine kısa bir hatırlatma istiyorsanız:

```
$ git add -h
```

Yardım Alma

Bizlere de sorabilirsiniz ☺

Git Repository (Repo) Oluşturma

1. Herhangi bir versiyon kontrol sistemi arasında yer almayan yerel dosya/dizininizi Git reposuna çevirerek
2. «Clone»layarak

Git reposu oluşturabilirsiniz.

1- Var olan Dizinden Repo Oluşturma

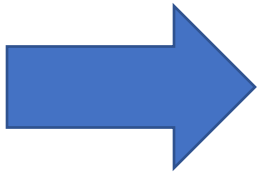
Öncelikle dizinimize ulaşalım:

Windows:

```
$ cd C:/Users/[USERNAME]/benim_projem
```

Linux:

```
$ cd /home/[USERNAME]/benim_projem
$ cd ~/[USERNAME]/benim_projem
```



```
git init
```

macOS:

```
$ cd /Users/[USERNAME]/benim_projem
```

1- Var olan Dizinden Repo Oluşturma (Devam)

- `git init` komutu «.git» adında yeni bir alt dizin oluşturarak repomuz ile ilgili tüm bilgileri burada tutar.
- `ls` (GNU/Linux ya da macOS) ya da `dir /a` (Windows) komutu ile `.git` dizinini görebiliriz.
- Linux / Windows :

  $ echo Hello World > test.txt

  $ git add *.txt

  $ git commit –m ‘Benim ilk commitim!’

2-Var olan bir Repoyu Klonlama

```
$ git clone <URL>
```

Örnek:

```
$ git clone https://github.com/libgit2/libgit2.git
```

bu komut «libgit2» adında terminaldeki bulunduğumuz konuma dizin oluşturur, .git dosyasını içinde başlatır, o repo için tüm verileri çeker ve en son sürümün çalışan bir kopyasını kontrol eder.

Eğer klonladığınız reponun adının sağladığınız url tarafından belirlenmesini istemiyorsanız:

```
$ git clone https://github.com/libgit2/libgit2.git libgit
```

Sayfa 34