git - Jadi

```
$ git init
```

گیت را در دایرکتوری جاری اجرا می‌کند.

```
$ git add .
$ git commit -m 'SOME Messages ...'
```

بعد از ساخت گیت برای ایجاد برنچ master.

```
$ git status
```

اتفاقات و تغییرات را در این دایرکتوری نشان می‌دهد.

```
$ git add FILE_NAME
```

اگر بخواهیم فایل مشخصی را به stage اضافه کنیم.

```
$ git add -A
```

اگر بخواهیم تمام فایل‌های دایرکتوری را به stage اضافه نماییم.

```
$ git commit -m 'SOME Messages ...'
```

فایل‌هایی که بخواهیم به stage اضافه نماییم را باید commit کنیم با یک message.

```
$ git log
```

تمام پیام‌هایی که در هنگام commit کردن وارد کردیم در اینجا نمایش داده می‌شود.

```
$ git diff HEAD
```

آخرین تغییراتی که در فایل‌ها انجام شده است را نسبت به حالت قبلی (قبل از commit کردن) نشان می‌دهد.
(HEAD همان commit آخر ما هست.)

```
$ git diff --staged
```

آخرین تغییرات فایل‌های روی stage را نشان می‌دهد.

```
$ git reset FILE_NAME
```

اگر بخواهیم فایل خاصی را از stage پاک کنیم که commit نشود.

```
$ git checkout -- FILE_NAME
```

اگر بخواهیم آخرین تغییرات فایلی خاص را به ذخیره قبلی (آخرین commit) بازگردانیم.

```
$ git branch
```

نشان می‌دهد چند شاخه در گیت جاری داریم.

```
$ git branch NAME_OF_BRANCH
```

ایجاد شاخه جدید در گیت جاری.

```
$ git checkout NAME_OF_BRANCH
```

جابجایی بین شاخه‌ها.

```
$ git merge NAME_OF_BRANCH
```

در صورتی که بخواهیم تغییرات انجام شده در برنچ مشخصی را با گیت مستر ادغام نماییم.

```
$ git rm FILE_NAME
```

برای پاک کردن فایل مشخصی هم از گیت و هم از فایل سیستم.

```
$ git branch -d NAME_OF_BRANCH
```

حذف شاخه مشخصی در گیت جاری.

```
$ git clone URL_ADDRESS
```

برای اضافه کردن یک دایرکتوری گیت مثلا از روی گیت هاب بر روی کامپیوتر شما.

```
$ git push origin master
```

توجه: origin همان جایی که پروژه گیت را در اینترنت داریم.

تغییرات master را بر روی origin می‌نویسد. (push می‌کند)

نیازی به اتصال با نام کاربری و رمز عبور دارد.

```
$ git pull origin master
```

تغییرات origin را بر روی master می‌نویسد. (pull می‌کند)

نیازی به اتصال با نام کاربری و رمز عبور ندارد.

```
$ git (pull or push) -u origin master
```

تغییرات origin را بر روی master می‌نویسد (pull می‌کند) یا تغییرات master را بر روی origin می‌نویسد. (push می‌کند)
دفعه بعد دیگر نیازی نیست origin و master را بنویسیم. فقط کافیست تایپ کنیم:

```
$ git (pull or push)
```

```
$ git remote add origin URL_ADDRESS
```

یک آدرس اینترنتی را به عنوان برنچ ریموت اضافه می‌کند و نام آن را origin می‌گذارد.

```
$ git show COMMIT_NUMBER
```

نمایش تغییرات بر اساس شماره commit که می‌گوید که در شماره commit مورد نظر چه اتفاقی افتاده است.

```
$ git tag
```

تگ‌های زده شده تا کنون را نشان می‌دهد.

```
$ git tag -a SOME_TEXT COMMIT_NUMBER -m 'SOME_MESSAGES...'
```

a- مخفف annotate (یه نوشته‌ای بذارم کنارش!) و با یک پیام متناسب (با توجه به شماره commit درج شده).

```
$ git tag -a SOME_TEXT -m 'SOME_MESSAGES...'
```

a- مخفف annotate (یه نوشته‌ای بذارم کنارش!) و با یک پیام متناسب (آخرین commit زده شده).

```
$ git tag -l "SOME_TEXT"
```

l- لیست تمام تگ‌هایی که در داخل دابل کوتیشن درخواست می‌کنیم را به ما می‌دهد. مثال:

```
$ git tag -l "V*"
```

لیست تمام تگ‌هایی که با V شروع می‌شود را به ما می‌دهد.

```
$ git show SOME_TAG
```

نام ایجاد کننده، تاریخ ایجاد، تغییرات نسبت به commitهای قبلی و ... بر اساس نام تگ را نمایش می‌دهد.

```
$ git push origin SOME_TAG
```

تگ مشخصی را بر روی origin می‌نویسد. (push می‌کند)

```
$ git push origin --tags
```

تمام تگ‌ها را بر روی origin می‌نویسد. (push می‌کند)

```
$ git checkout SOME_TAG
```

جابجایی بین تگ‌های مشخص.

```
$ git checkout -b <new-branch-name>
```

برای اینکه بتوان تغییراتی که روی تگ‌های مشخص (مثلا ورژن) قابل اعمال باشد باید شاخه (branch) جدیدی برای این کار ساخته شود.

```
$ gpg --list-keys
```

لیست کلیدهای موجود در سیستم را نشان می‌دهد.

```
$ gpg --gen-key
```

برای تولید کلید عمومی و خصوصی جدید.

```
$ git config --global user.name
```

نمایش نام کاربری عمومی.

```
$ git config --global user.email
```

نمایش ایمیل کاربر عمومی.

```
$ git config --global user.signingkey
```

نمایش کلید امضا عمومی.

```
$ gpg --list-secret-keys --keyid-format LONG
```

نمایش کلید ایجاد شده.

```
$ git config --global user.signingkey 26EB79AE89CD20D0
```

اعمال کلید ایجاد شده توسط gpg.

```
$ git tag -s SOME_TEXT -m 'SOME_MESSAGES...'
```

s- مخفف sign (امضاش کن!) و با یک پیام متناسب (آخرین commit زده شده).

```
$ git show SOME_TAG_SIGNED
```

نام ایجاد کننده، تاریخ ایجاد، امضای ایجاد کننده، تغییرات نسبت به commitهای قبلی و ... بر اساس نام تگ را نمایش می‌دهد.

```
$ git tag -v SOME_TAG_SIGNED
```

برای اطمینان از امضای تگی مشخص از این دستور استفاده می‌شود.

```
$ git commit -S -m 'SOME Messages ...'
```

با اضافه کردن S- این commit امضا می‌شود.

```
$ git help COMMAND
```

برای دیدن راهنمای هر دستور به صورت بالا عمل نمایید.

```
$ git blame NAME_OF_FILE -L{NUMBER_OF_LINE}
```

برای مشاهده تغییرات و مسئول ایجاد (نفری که تغییرات را انجام داده است) آن در یک فایل (و به طور در دقیق‌تر با سوئیچ L- در یک خط مشخص یا در چندین خط) از دستور بالا استفاده می‌شود. به طور مثال:

```
$ git blame NAME_OF_FILE -L8
```

مشاهده تغییرات و مسئول ایجاد آن در فایلی مشخص و در خط 8

```
$ git blame NAME_OF_FILE -L8,10
```

مشاهده تغییرات و مسئول ایجاد آن در فایلی مشخص و در خط 8 تا 10

```
$ git bisect start
binary search commit
```

به منظور خطایابی در کد زده شده از این دستور استفاده می‌شود. در ابتدای امر باید در بالاترین سطح (دایرکتوری) باشید. با اعمال دستور فوق می‌توانید عملیات را شروع نمایید.

```
$ git bisect bad COMMIT_NUMBER
```

با دستور فوق اعلام می‌کنید که وضعیت در اینجایی که هستید بد است (البته اگر در ادامه دستور فوق commit number ای هم زده شود نشان می‌دهد که وضعیت در آن commit بد است)

```
$ git bisect good COMMIT_NUMBER
```

در ادامه‌ی دستور فوق، گیت برای شما در commit های قبلی جستجو کرده و با مقایسه تغییرات به شما پیشنهاد می‌دهد که مثلاً در commit های قبلی فلان commit خوب است؟ که اگر با آن موافق بودید (یعنی باگ ایجاد شده برطرف می‌شد!) می‌توانید از دستور بالا استفاده فرمایید.