Git

جهت دانلود git می توانید از سایت زیر اقدام نمایید.

https://git-scm.com

config

جهت اعمال تنظیمات git مورد استفاده قرار می گیرد(از جمله افزودن نام کاربری و ایمیل که به منظور استفاده از git ورود آنها الزامی می باشد). در صورتی که قصد استفاده از github را دارید بهتر است اکانت آن را وارد کنید.

```
Git config --global user.email “jadidi.v@gmail.com”
Git config --global user.name “jadidiv”
```

init

بعد از آنکه Git CLI را روی سیستم‌تان نصب کردید، به دایرکتوری که قصد پیاده‌سازی پروژه گیت در آن را دارید بروید. وقتی که در آن دایرکتوری قرار گرفتید، دستور git init را اجرا نمایید.

حال مخزن گیت شما پیاده‌سازی شد. از این به بعد گیت تمام تغییرات مربوط به این دایرکتوری را ثبت خواهد کرد. اگر به صورت درست تمام دایرکتوری‌های موجود در مخزن‌تان را بررسی نمایید، مشاهده می‌کنید که یک پوشه .git قرار گرفته است که پیکربندی‌های مربوط به گیت شما در آن قرار می‌گیرد.

status

می‌توانید وضعیت مخزن گیت را هر زمان که می‌خواهید با استفاده از دستور gitstatus بررسی کنید.

در واژه‌شناسی مربوط به گیت، فایل‌ها بعد از اینکه ذخیره و آماده کامیت کردن شدند، در مرحله staged قرار می‌گیرد. بعد از اینکه فایل‌ها در یک دیتابیس محلی واقع در پوشه .git قرار گرفتند به وضعیت committed تغییر پیدا می‌کنند. و وقتی که تغییراتی در آن‌ها قرار دادید اما آن‌ها(تغییرات) را هنوز کامیت نکرده‌اید، به وضعیت modified در می‌آیند.

Add

می‌توانید فایل‌ها را به دایرکتوری مربوط به پروژه یا همان مکان Stage با استفاده از دستور زیر وارد کنید:

```
git add index.html style.css images
```

این دستور فایل‌های index.html و style.css و پوشه images را به وضعیت Stage در می‌آورد. اگر می‌خواهید تمام موارد قرار گرفته در پوشه‌ای که در حال کار هستید را به حالت stage در بیاورید، کافی است دستور زیر را وارد کنید:

```
git add .
```

Rm

می‌توانید به سادگی فایل‌ها را از قسمت stage نیز حذف کنید

```
git rm --cached index.html style.css
```

اگر می‌خواهید پوشه‌ها را نیز حذف کنید به یک پرچم -r نیازمندید:

```
git rm --cached -r images
```

بعد از اجرای این دستور گیت به شما پیام‌های تاییدیه را می‌دهد، بنابراین شانس حذف اشتباهی فایل‌ها و دایرکتوری‌ها کمتر می‌شود.

برای اینکه تمام فایل‌ها و دایرکتوری‌ها را یکجا حذف کنید می‌توانید به صورت زیر عمل کنید:

```
git rm --cached -r .
```

Commit

می‌توانید از ناحیه stage مربوط به پروژه‌تان در هر زمان یک ذخیره مانند بگیرید. این حالت را کامیت کردن می‌نامند و فایل‌ها را به بانک اطلاعاتی ارسال می‌کنند.

همواره با ارسال فایل‌های stage یک پیغام نیز نوشته می‌شود. در دستور بالا پیغام Initial commit نوشته شده است. برای اینکه بهتر بتوانید کامیت‌های‌تان را قابل استفاده کنید و افراد مختلف کار آن را درک کنند از این پیغام‌ها استفاده کنید.

```
git commit -m "Initial commit"
```

از دستور زیر برای Add کردن و Commit کردن همزمان استفاده می شود.

```
git commit -a -m "Initial commit"
```

Diff

برای اینکه بتوانید تمام تغییرات اتفاق افتاده در یک مخزن گیت را مشاهده کنید، می‌توانید لیست آن‌ها را از طریق دستور زیر بدست بیاورید:

```
git diff
```

این دستور نه تنها نام فایل‌ها را برمی‌گرداند بلکه تغییرات مربوط به آن‌ها را نیز در قالب متن به شما برگشت می‌دهد. دستور git diff موارد اضافی را با +++ و موارد حذف شده را با --- نمایش می‌دهد.

log

گیت همچنین به شما اجازه می‌دهد تا تاریخچه کامل کامیت‌های‌تان را در پروسه توسعه مشاهده کنید. برای انجام این کار می‌توانید دستور git log را وارد کنید

لاگ خروجی شامل آی‌دی، نویسنده، تاریخ و پیغام مربوط به هر commit است.

```
git log
```

جهت نمایش اطلاعات در یک خط از دستور زیر استفاده کنید.

```
git log  --oneline
```

همچنین می توانید با وارد کردن دستور زیر می توانید ۵ تغییر آخر را مشاهده نمایید.

```
git log -2 --oneline
```

جهت نمایش جزئیات تغییرات ایجاد شده نیز می توانید از دستور زیر استفاده کنید.

```
git log -p
```

Checkout

جهت برگرداندن تغییرات انجام شده (قبل از اضافه شدن به stage) به آخرین Commit استفاده می شود.

```
git checkout -- .
```

Reset

در صورتی که تغییرات را add کرده اید و به stage اضافه شده اند می توانید با دستور زیر تغییرات را از stage خارج کنید(UNSTAGE) و بعد از آن از دستور checkout جهت برگرداندن به آخرین commit استفاده کنید.

```
git reset HEAD .
```

در صورتی که بخواهید به commit خاصی برگردید می توانید از دستور زیر استفاده کنید.

```
git reset id
```

Id شناسه کامیت مورد نظر شماست. (قسمتی از id نیز مورد قبول است و نیاز نیست کل id را وارد نمایید.)

نکته: زمانی که از دستور reset استفاده می کنید تغییرات بعد از commit مورد نظر شما نیز وجود دارند و میتوانید آنها را به stage اضافه کرده یا آنها را کلا با دستور checkout حذف نمایید در صورتی که نمیخواهید این تغییرات نمایش داده شوند می توانید دستور reset را به شکل زیر استفاده کنبد.

```
git reset -- hard id
```

Branch

جهت ساخت branch جدید می توانید از دستور زی استفاده کنید. (branch با نام dev ساخته خواهد شد.)

```
git branch dev
```

جهت نمایش branch های پروژه می توانید از دستور زیر استفاده کنید.

```
git branch -a
```

جهت جابجا شدن (سوئیچ) به branch های پروژه می توانید از دستور زیر استفاده کنید. (به branch با نام dev سوئیچ می کنیم.)

نکته: قبل از سوئیچ کردن به یک branch همیشه دقت کنید که تغییرات شما commit شده باشد در غیر اینصورت تغییرات شما بعد از سوئیچ شدن از بین خواهد رفت.

```
Git checkout dev
```

جهت حذف branch از دستور زیر استفاده می کنیم. (به منظور حذف یک branch ابتدا می بایست از آن برنچ خارج شده باشید.)

```
git branch -d dev
```

Merge

جهت merge کردن دو branch از دستور زیر استفاده می کنیم.

```
git merge dev
```

جهت نمایش log ها به صورت گراف از دستور زیر استفاده می کنیم.

```
git log –graph
```

Stash

جهت ذخیره سازی تغییرات هنگامی قصد ندارید تغییرات را commit کنید از دستور زیر می توانید استفاده کنید.

نکته: اگر تغییرات خود را commit نکنید و بین branch ها سوئیچ کنید تغییراتی که داشته اید به آن branch انتقال خواهند یافت اگر قصد دارید تغییرات را به همراه خود به branch دیگر انتقال ندهید بهتر است با دستور stash این تغییرات را به صورت موقت ذخیره تمایید.

نکته: stash ها در branch های دیگر نیز قابل مشاهده و استفاده می باشند.

```
git stash
```

جهت نمایش لیست stash های ذخیره شده از دستور زیر استفاده کنید.

```
git stash list
```

جهت حذف stash های خود می توانید از دستور زیر استفاده کنید.

```
git stash drop  stash@{0}
```

به منظور ثبت یک پیام هنگام دخیره stash می توانید از دستور زیر استفاده کنید.

```
git stash save “Text”
```

جهت نمایش تغییرات موجود در stash میتوانید از دستور زیر استفاده کنید.

```
git stash show stash@{0}
```

نکته جهت نمایش جزئیات تغییرات نیز می توانید از دستور زیر استفاده کنید.

```
git stash show -p stash@{0}
```

جهت اعمال کردن stash ها می توانید از دستورات زیر استفاده کنید.

```
git stash applay
```

```
git stash pop
```

نکته: تفاوت این دو دستور این است که در pop پس از اعمال تغییرات stash نیز پاک می شود ولی applay کردن تغییرات را انجام می دهد اما باعث پاک شدن stash نمیشود و در صورت لزوم خودتان باید با دستور drop آن را پاک کنید.

Ignore

جهت دنبال نشدن یک فایل توسط git مورد استفاده قرار می گیرد

به منظور ساخت فایل gitignore از دستور زیر استفاده می کنیم.

```
touch .gitignore
```

درون فایل ساخته شده می توان نام پوشه یا پوشه ها، فایل یا فایل ها و یا فایل هایی با پسوند مورد نظر خود را که قصد دنبال شدن نداریم وارد کنید. (همانند مثال زیر)

و سپس فایل gitignore را درون پروژه commit می کنیم.

به عنوان مثال:

node_modules /	پوشه و فایل های درون node_modules
Test.php	فایل مورد نظر
*.txt	تمام فایل هایی با پسوند txt
!a.txt	استثنا: فایل مورد نظر توسط git دنبال شود

نکته: دنبال نشدن فایل هایی که از قبل توسط git دنبال شده اند.

ابتدا فایل یا پوشه مورد نظر را در فایل gitignore قرار داده و آن را commit کنید سپس با دستور زیر کش را پاک نموده و بلافاصله یک بار دستورات مربوط به add کردن و commit کردن را اجرا کنید.

```
git rm --cached -r .
git add .
git commit -m “add to ignore”
```

GitHub

به منظور استفاده از github می بایست ابتدا یک اکانت در سایت آن بسازید و ایمیل خود را در آن فعال کنید.

https://github.com

سپس اقدام به ساخت یک repository کرده و اقدامات لازم را برای اتصال و انتقال پروژه به آن انجام می دهیم.

Remote

این دستور جهت ایجاد remote به github (و ساخت یک نام برای ریموت) و به منظوراتصال به github مورد استفاده قرار می گیرد.

```
git remote add origin https://github.com/java1365/project.git
```

نکته: origin نام یک remote است.

جهت مشاهده لیست ریموت ها از دستور زیر می توانید استفاده کنید.

```
git remote
```

Push

جهت انتقال پروژه به مخزن github استفاده می شود.

```
git push -u origin master
```

نکته: origin نام remote و master نام مخزنی که قصد انتقال آن را داریم.

پس از برقراری اولین اتصال با github جهت انتقال مجدد (update) مخزن دیگر می توان دستور را به صورت زیر انجام داد.

```
git push
```

Pull

دریافت آخرین تغییرات پروژه از github و استفاده از آن درون سیستم خود از دستور زیر می توانید استفاده کنید.

```
git pull origin
```

نکته: origin نام remote مورد نظر ماست که قبلا آن را ساخته ایم.

Clone

جهت گرفتن پروژه از github (زمانی که شما پروژه را در سیستم خود ندارید) مورد استفاده قرار می گیرد.

```
git clone https://github.com/jadidiv/project.git myproject
```

نکته: https://github.com/jadidiv/project.git آدرس مخزن بر روی github و myproject نام پوشه ای که پروژه درون آن clone می شود می باشد. (نام پوشه اجباری نیست اگر درج نشود نام repository در github قرار می گیرد.)