دستورات قابل استفاده در گیت

نصب گیت و تنظیمات اولیه در مخزن محلی

```
$ sudo  yum install git
$ git  –version


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

ساخت پوشه گیت

```
$ mkdir myproject
$  cd myproject

# initialize Git
$ git init
```

بررسی وضعیت repository

```
$ git status
```

با استفاده از –short تغییرات را فشرده تر به ما می دهد.

اضافه کردن فایل و کامیت

```
$ git add  “file or dir”
```

از پارامتر –all و -A برای اضافه کردن تمام فایل های پوشه گیت استفاده می شود.

دستور git rm برای حذف فایل ها از مخزن Git استفاده می شود. می توان آن را معکوس دستور git add در نظر گرفت.

```
$ git rm file
```

اضافه کردن commit .پیشرفت و تغییرات ما را در حین کار پیگیری می کند. Git هر نقطه تغییر commit یا "save point" را در نظر می گیرد. این نقطه ای از پروژه است که اگر باگ پیدا شد یا می خواهید تغییری ایجاد کنید، می توانید به آن بازگردید.

```
$ git commit  -m “message”
```

منظور -m پیامی هست که همیشه باید همراه کامیت باشد کامیت ها با پیام ها از هم متمایز می شوند.
گاهی اوقات تغییرات کوچکی اعمال می شود و و می توان به صورت مستقیم تغییرات را اعمال کنیم برای این کار از -a استفاده میکنیم

```
$ git  commit -a  -m  “update new line file”
```

برای مشخص شدن تمام commit در یک مخزن می توان از مخزن log گرفت

```
$ git log
$ git shortlog # summarize log
$ git show # show type object
```

به هر کامیت علاوه بر پیام کامیت می توان یک سری یادا داشت نیز اضافه نمود

```
$ git notes add -m “message” commitsha
```

ساخت شاخه (branch)

فرض کنید یک پروژه بزرگ دارید که دارای بخش های و قسمت های مختلف می باشد و می خواهید هر طرح و قسمت را جداگانه به روز رسانی کنید برای همین گیت با استفاده از branch این امکان را به ما می دهد.

مزیت استفاده از branch در گیت

- با یک شاخه جدید به نام new-design، کد را مستقیماً بدون تاثیر بر شاخه اصلی ویرایش کنید
- رفع خطا در شاخه ی دیگر و ادغام آن با شاخه اصلی
- ایجاد طراحی جدید و کار روی ان و ادغام با شاخه اصلی

```
$ git  branch -M namebranche
```

برای استفاده به عنوان main یا پیشفرض از -m استفاده میکنیم.

برای switch کردن در branch های مختلف از دستور زیر استفاده می شود.

```
$ git checkout namebranch
$ git switch
```

همچنین با استفاده از این دستور می توان ورژن قبلی یک فایل را برگرداند

```
git checkout 8a7b201 index.html
```

شماره commitshagit را باید داشته باشید

برای ایجاد و انتقال به یک branch جدید از -b استفاده می شود

برای حذف کردن شاخه از دستور زیر استفاده می شود.

```
$ git branch –delete namebranch
```

ادغام branch

```
$ git checkout master
$ git merg namebranchmerg
```

زمانی که بین دو branch یک فایل در تضاد باشد در زمان ادغام کردن conflict رخ می دهد پس باید ابتدا فایل مدنظر کامیت کرده و سپس ادغام شود.

دستور جا به جایی کامیت ها در شاخه ها

یک دستور قدرتمند است که به شما امکان می‌دهد تا commitهای دلخواه Git را با مرجع انتخاب کرده و به HEAD فعلی اضافه کنید.به عنوان مثال، بگوییم commit به طور تصادفی به شعبه اشتباهی انجام شده است. می‌توانید به شاخه درست بروید و commit را به جایی که باید تعلق داشته باشد انتخاب کنید.

```
$ git cherry-pick commitSha
```

دستور reset

دستور git reset یک ابزار پیچیده و همه کاره برای لغو تغییرات است. سه شکل اصلی فراخوانی دارد. این فرم ها با آرگومان های خط فرمان مطابقت دارند

```
$ git reset –hard # new state repo
$ git reset –mixed  # reset add
$ git reset –soft  # reset commit
```

دستور revert

یک عملیات برگرداندن commit مشخص شده را می گیرد، تغییرات آن commit را معکوس می کند و یک "revert commit" جدید ایجاد می کند. سپس نشانگرهای ref به روز می شوند تا به ارتکاب بازگشت جدید اشاره کنند و آن را به نوک شاخه تبدیل کنند

```
$ git revert HEAD
```

دستور restor

برای بازیابی یا حذف تغییرات محلی غیرمتعهد فایل ها استفاده می شود.
فرض کنید تغییراتی را در برخی فایل‌ها انجام داده‌اید و سپس اگر می‌خواهید آن تغییرات محلی را کنار بگذارید، می‌توانید با خیال راحت از git restore استفاده کنید. یکی دیگر از موارد استفاده این است که اگر می خواهید یک فایل را unstage کنید می توانید از این دستور استفاده کنید. به عبارت دیگر، برای خنثی کردن اثرات git add استفاده می شود.

```
$ git restore  filename
```

دستور clean

این دستور تا حدی یک دستور 'Undo' است. Git clean را می توان مکمل دستورات دیگری مانند git reset و git checkout دانست. در حالی که این دستورات دیگر روی فایل‌هایی که قبلاً به فهرست ردیابی Git اضافه شده‌اند عمل می‌کنند، دستور git clean روی فایل‌های ردیابی نشده عمل می‌کند. فایل های ردیابی نشده فایل هایی هستند که در دایرکتوری کاری مخزن شما ایجاد شده اند اما هنوز با استفاده از دستور git add به فهرست ردیابی مخزن اضافه نشده اند.

```
$ git clean -n  #show file  not add
Git clean -f    #remove file not add
git clean -d   #remove dir not add
```

دستور mv

این دستور برای انتقال فایل و یا عوض کردن نام استفاده می شود.

```
$ git mv filename1 filename2
```

دستور gc

این دستور مخزن را از داده های ضروری پاک میکند عملیات هایی مانند فشرده‌سازی ویرایش‌های فایل (برای کاهش فضای دیسک و افزایش کارایی)، حذف اشیاء غیرقابل دسترس که ممکن است از فراخوان‌های قبلی git add ایجاد شده باشند را به روز می کند

```
$ git gc
```

دستور describe

این دستور جدیدترین برچسبی را که از یک commit قابل دسترسی است پیدا می کند. اگر تگ به commit اشاره کند، فقط تگ نشان داده می شود. در غیر این صورت، نام تگ را با تعداد commit های اضافی در بالای شی برچسب گذاری شده و نام شی مخفف آخرین commit پسوند می کند.

```
$ git describe
```

دستور grep

این دستور برای پیدا کردن یک خط در فایل ها می باشد این کار برای سرعت در کد نویسی است.

```
$ git grep 'line string' – 'file'
```

آشنایی با github و اضافه کردن مخزن local به server

بعد از ساختن Repository و گرفتن ادرس ان باید آدرس را به git در مرحله قبل ساخته ایم اضافه کنیم

```
$ git remote add origin url
```

برای اضافه کردن به صورت زیر عمل میکنیم

```
$ git push origin master
```

گرفتن پروژه از github و آپدیت نگه داشتن

هنگام کار به عنوان تیم روی یک پروژه، مهم است که همه به روز بمانند.هر زمان که شروع به کار روی یک پروژه می کنید، باید جدیدترین تغییرات را در نسخه محلی خود دریافت کنید.

دستور pull ترکیبی از دو دستور مختلف می باشد fetch , merg که می تواند پروژه را از منبع بگیرد.

دستور fetch تمام تاریخچه تغییرات یک شعبه/ مخزن ردیابی شده را دریافت می کند.بنابراین، در Git محلی خود، به‌روزرسانی‌ها را واکشی کنید تا ببینید چه چیزی در GitHub تغییر کرده است:

```
$ git fetch origin
```

با این دستور می توان آخرین تغییرات را داشته باشید . همچنین می توان تغییرات منابع سرور و محلی را نمایش داد

```
$ git diff origin/master git.lo
```

همچنین می توان با استفاده از range-diff دو کامیت را با هم مقایسه نمود و یا در دو branch lojgt این کار را انجام داد.نکته قبلا از این دستور باید دستور diff زده شود تا اعداد زیر از آن بدست آید

```
$ git range-diff @{2} @{1} @
```

دستور merge شاخه فعلی را با یک شاخه مشخص ترکیب می کند.با fetch تأیید کرده‌ایم که به‌روزرسانی‌ها مطابق انتظار هستند، و می‌تواند شاخه فعلی (مستر) خود را با مبدا/مستر ادغام کنیم:

```
$ git merg origin/master
```

اما اگر فقط بخواهید مخزن محلی خود را بدون گذراندن تمام آن مراحل به روز کنید از دستور pull استفاده می کنیم

```
$ git pull origin
```

دستور rebase

یکی از دو ابزار Git است که در ادغام تغییرات از یک شاخه به شاخه دیگر تخصص دارد. ابزار ادغام تغییرات دیگر git merge است. ادغام همیشه یک رکورد تغییر رو به جلو است. روش دیگر، rebase دارای ویژگی های قدرتمند بازنویسی تاریخ است
فرآیند انتقال یا ترکیب دنباله ای از commit ها به یک commit پایه جدید است. Rebasing بسیار مفید است و به راحتی در زمینه یک جریان کاری شاخه‌بندی ویژگی قابل مشاهده است. روند کلی می تواند باشد

```
$ git rebase namebranch
```

مشارکت در پروژه شخصی دیگران

یک fork یک کپی از یک مخزن است. ,زمانی مفید است که می خواهید در پروژه شخص دیگری مشارکت کنید یا پروژه خود را بر اساس پروژه او شروع کنید. fork یک فرمان در Git نیست، بلکه چیزی است که در GitHub و سایر میزبان های مخزن ارائه می شود. l

```
$ git  clone url.io
```

بعد از گرفتن یک پروژه و ایجاد تغییرات روی یک پروژه به صورت مشارکتی می توان آن را push نمود

```
$ git push origin
```

آرشیو کردن git

برای اینکه بتوانیم پروژه را آرشیو کنیم از دستور زیر استفاده می کنیم

```
git archive --output=./example_repo_archive.tar --format=tar HEAD
```

باید دایرکتوری خروجی و همچنین فرمت و همچنین کدام commit مدنظر را مشخص کنیم