git

در دنیای نرم افزار یکی ار بزرگترین دغدغه های برنامه نویسان ورژن بندی فایل ها و کدهاست. به منظور رفع این مشکل ابزاری به اسم گیت اختراع شد.

بعد ها گیت خیلی فراتر رفت و امروزه به یک ابزار بسیار مهم و ضروری در دنیای نرم افزار تبدیل شده است. به همین منظور ما تعدادی از دستورات مهم آن را با هم بررسی می کنیم.

init:

این دستور جهت ایجاد یک پروژه گیت استفاده می شود.

```
git init project_name
```

clone :

در پروژه های آنلاین برای دانلود فایل ها و دسترسی به repo از این دستور استفاده می شود.

```
git clone project_url
```

status :

فایل ها در گیت می توانند وضعیت متفاوتی داشته باشند. بعضی از آن ها را git نمی شناسد چون هنوز به آن معرفی نشده اند که به آن ها unstaged میگوند.

به فایل هایی که به گیت شناسانده شده و منتظر commit هستند staged میگوند.

اگر فایلی تغییر کند هم گیت متوجه آن تغییر می شود و فایل را در حالت modified قرار می دهد.

دستور status اطلاعات دیگری مانند نامbranch ، تعداد commit هاو ... را نیز در اختیار کاربر قرار میدهد.

```
git status
```

add :

از این دستور به منظور اضافه کردن فایل های جدید به پروژه استفاده می شود.

```
git add filename
```

دستور add از regex هم پشتیبانی می کند.

مثال: فقط فایل هایی با پسوند txt اضافه شوند.

```
git add *.txt
```

commit :

از این دستور به منظور دائمی کردن تغییرات در فایل ها استفاده می شود که ه هر commit یک message هم تعلق می گیرد که بیان گر خلاصه ای از اتفاقات آن commit است.

```
git commit -m “message“
```

نکته:

چنانچه تغییری در هر یک از فایل های پروژه اتفاق بیوفتد، قبل از commit نیاز هست که آن فایل مجدد به git شناسایی شود که برای این کار از دستور add استفاده می کردیم.

اما اگر بخواهیم همه ی تغییرات را یکجا commit کنیم می توانیم از میانبر a- استفاده کنیم.

```
git commit -a -m “message“
```

log :

با این دستور می توان تمام کامیت های پروژه را به صورت یکجا دید.

```
git log
```

restore :

از این دستور برای بازنشانی تغییرات یک فایل استفاده می شود.

git restore filename

branch :

branch بسیار مفهوم مهمی هستند. در پروژه های واقعی افراد به صورت همزمان بر روی یک پروژه کار می کنند و تغییرات بسیار زیاد و سریع هستند و این تغییرات زیاد در زمان های متخلف می تواند عملکرد محصول را مختل کند.

برای جلوگیری از این مشکل از branch استفاده می کنیم.

بعد از ایجاد پروژه به صورت خودکار یک branch هم در آن ایجاد می شود که به آن main یا master میگوییم.

بقیه ی branch ها از روی آن ایجاد می شوند.

یک branch در اصل یک کپی از روی branch اصلی است که به ما اجازه می دهد تغییرات را در فایل ها اعمال و تست کنیم بدون اینکه کارکرد پروژه مختل شود و در انتها چنانچه تغییرات مطلوب بود می توانیم branchرا با branch اصلی ادغام کنیم.

git branch → لیست branch های موجود در پروژه

git branch branch_name → ساخت یک branch جدید از روی branch ی که در آن قرار داریم.

checkout :

از این دستور جهت جا به جایی روی branch ها استفاده می شود.

```
git chechout branch_name
```

pull :

در پروژه های واقعی ما نیاز داریم تا تغییرات ایجاد شده در فایل ها را روی repo قرار دهیم و تغییراتی که دیگران ایجاد کرده اند را نیز از repo بگیریم.

دستور pull جهت دانلود یک branch استفاده می شود.

```
Git pull origin branch_name
```

push :

از این دستور جهت آپلود branch روی repo استفاده می شود.

pull request :

زمانی که میخواهیم یک branch را روی دیگری merge کنیم ابتدا باید یک pull request ایجاد کنیم.

merge :

زمانی که از تغییرات در یک branch راضی هستیم و میخواهیم آن تغییرات را به یک branch دیگر اضافه کنیم از این دستور استفاده می کنیم.

این دستور یک branch را در در برنچی که روی آن قرار داریم ادغام می کند.

```
git merge branch_name
```