<= Git =>

مدیریت کدمنبع

اصطلاح کنترل کدمنبع (source control) یا کنترل نسخه(version control) ، همچنین (revision control به عمل مدیریت، کنترل و ثبت و ضبط مستنداتی همچون کدها و مستندات نرم افزاری، اطلاعات وبسایت‌ها، تاریخچه‌ی تغییرات تصاویر و مواردی از این دست اطلاق می‌شود. تغییرات معمولا با یک شماره نسخه ذخیره می‌شوند .سیستم‌های کنترل‌نسخه (version control systems) هم بصورت مجزا برای کنترل تغییرات نو شته‌ها و کدها و در مواردی کنترل تصاویر و فایل‌های ویدیویی ساخته و عرضه میشوند و هم در برخی از نرم‌افزارهای واژه‌پرداز یا ویرایشگرهای تصویر و سایر انواع مستندات بصورت درونی پیاده سازی شده‌اند .نرم‌افزارهای کنترل نسخه، پایه‌ی کارهای تیمی در تیم‌های نرم افزاری متشکل از چند توسعه‌دهنده است .در تیم‌های نرم‌افزاری، انتشار نسخه‌های مختلف از نرم‌افزار، توسعه بصورت تیمی و رفع کردن باگها بصورت اشتراکی و مواردی از این دست همواره درحال انجام هستند. این امور بدون حضور نرم‌افزاری قدرتمند برای کنترل امور، بسیار سخت، با پتانسیل بالای خطا و طاقت فرسا ست و البته نتیجه‌ی حاصل هم آنچه که باید باشد، نیست. سیستم‌های کنترل نسخه و مدیریت کد منبع امور ذکر شده را از هر زمانی آسانتر کرده‌اند.

مدیریت کد منبع باgit

کنترل نسخه روشی برای ثبت تغییرات کدها و نوشته‌هایمان است، با این امکان که هرگاه بنا برنیازمان به مرحله‌ی خاصی از تغییرات ثبت شده برویم.

روش ها و همچنین ابزارهای زیادی برای این کار معرفی شده اند و git یکی از این ابزارهاست که یکی از بهترین برنامه‌نویسان یعنی لینوس توروالدز -خالق لینوکس- آن را برای اولین بار برای توسعه‌ی کرنل لینوکس توسعه داد و هم اکنون به ابزاری کلیدی در عمده‌ی پروژه‌های برنامه‌نویسان بدل شده است.

در سرتا سر اینترنت می‌توانید دا ستان‌های زیادی از دلیل و تاریخچه‌ی ایجاد گیت بخوانید، این نو شتار تنها نگاهی کوتاه و کاربردی به این موضوع دارد.

چرا نیاز داریم از ابزارهای مدیریت کد منبع استفاده کنیم؟

اگر تابحال تجربه‌ای هرچند کوتاه در برنامه‌نوی سی دا شته با شید احتمالا تجربه از د ست دادن بخ شی از پروژه را به دلایل گوناگون مثل ذخیره ناخوا سته در هنگام قطعی برق یا تغییراتی که بعدا باعث بروز م شکل شده‌اند و دلایل بی شمار دیگر را داشــته‌اید. به‌عنوان راه‌حل هم شــاید روش‌هایی مانند کپی‌گرفتن از کل فایل‌ها و اطلاعات پروژه در زمان‌های مختلف برای حفظ حالت خاصی از تغییرات را استفاده کرده باشید.

این کار تا حدی جواب می‌دهد، اما در این صــورت با انبوهی از دایرکتوری‌هایی که مانند یک غول بی‌شــاخ و دم بزرگ می‌شوند چه میکنید؟ و از آن بدتر چگونه آن را با افراد دیگری که با شما در انجام آن همکاری میکنند مشترک می‌شوید؟

ابزارهای مدیریت کد منبع پاسخی برای این شلختگی‌ها و شلوغی‌هاست. با استفاده از این ابزارها می‌توانید هرلحظه ای که مایل بودید تغییرات خود را ثبت کنید، به تغییرات ثبت شده در گذ شته برگردید و به راحتی با دو ستان و افراد تیمتان روی پروژه ای همکاری کنید، بدون اینکه نگران بهم‌ریختگی و نامنظم شدن کدهای پروژه با شید. میتوانید برای هر تغییر که ثبت میکنید توضــیحاتی بنویســید، تغییراتی که دیگران در پروژه لحاظ نموده اند را ببینید و البته منشــاء باگ‌ها و خطاهای احتمالی را به سادگی یافته و رفع و رجوع کنید.

در این میان ابزارهای زیادی برای این‌د ست کارها ساخته شده ا ست که هرکدام نگاه و شیوه‌ی متفاوتی را برای حل این م شکل در پیش گرفته ا ست. معروفترین این ابزارها عبارتند ازgit ، svn، mercurialو cvsکه البته در این میان gitیکی از جوانترین و پرطرفدارترین ابزارهای مدیریت کدمنبع اســت و ویژگی‌های ســاده و متمایز آن باعث شــده عمده‌ی تیم‌های نرم‌افزاری در دنیا به استفاده از این ابزار خوب روی بیاورند. در این کتاب با گیت آشنا میشویم.

گیت چیست؟

گیت یک سیستم مدیریت کد منبع توزیع شده است که میتوانید نوشته‌ها و کدهایتان را با آن در سیستم شخصی خودتان مدیریت کنید و تغییرات کدهایتان را داشــته باشــید، به تغییراتی در گذشــته برگردید مثلا به ریلیز خاصــی از پروژه، کدها را

روی یک سرور گیت (مانند گیت هاب) با دیگران سهیم شوید و گروهی روی توسعه‌ی یک پروژه همکاری کنید و از تغییراتی که هر عضو روی پروژه میدهد آگاهی یابید.

به پروژه‌هایی که در آنها گیت اســـتفاده میشـــود مخزن (repository) میگویند. یک مخزن گیت حاوی تمامی کدها، تغییرات کد و تنظیمات گیت برای آن پروژه است.

در سیستم‌های توزیع‌شده هرکسی که به مخزن اصلی دسترسی دارد، می تواند یک کپی از مخزن اصلی را در اختیار داشته و تغییرات خود را روی آن اعمال کند و همچنین می‌تواند این تغییرات را با تغییرات بقیه‌ی اعضـــا ترکیب کرده و یا به مخزن اصلی اضافه کند.

چراgit

امروزه اشخاص و شرکتهای کوچک و بزرگ زیادی از سیستم مدیریت کد منبع گیت برای کنترل کدهای نرم‌افزارها، پروژه ها و مستندات استفاده میکنند. کرنل لینوکس ، زبان برنامه‌نویسی روبی و فریم‌ورک لاراول نمونه ای از پروژه هایی هستند که بر روی سرویس گیت هاب نگهداری می شوند.

گیت به تغییرات کدها تنها به عنوان چند خطی که تغییر میکنند نگاه نمیکند و پس از ثبت هر تغییر) اصـــطلاحا کامیت) ((commitیک تصـویر کلی از پروژه را در لحظه‌ی آن تغییر ذخیره میکند و بررســی تغییرات با اطمینان و سـرعت بالاتری امکانپذیر میشود.

همچنین گیت برخلاف بسیاری از سیستم‌های مدیریت کد منبع وابستگی به سرور اصلی ندارد و تقریبا هرکاری را میتوان بصورت محلی (local) انجام داد، در هر لحظه ای و هر شرایطی کافیست تنها گیت را روی سیستم خود داشته باشید، تغییراتتان را ثبت کنید و هر زمان که به شبکه‌ای که سرور گیت شما در آن قرار دارد (مانند اینترنت!) دسترسی داشتید میتوانید تغییرات ذخیره شده را به مخزن روی سرور اضافه کنید و البته تمام تغییرات پروژه را هم بدون نیاز به اینترنت در مخزن محلی خود داشته باشید.

استفاده از گیت بسیار ساده است. تیم توسعه‌ی گیت یک نرم‌افزار تحت خط فرمان برای استفاده از گیت ساخته است. همچنین پروژه های زیادی هم برای کار با گیت چه به‌عنوان افزونه (plugin) برای ویرایشـــگرها و محیط های مجتمع

توسعه و چه بصورت برنامه‌های جدا با رابط گرافیکی توسعه داده می شوند و امروز تقریبا در هرمحیطی میتوان روشی برای استفاده از گیت یافت.

نصب گیت

پیش‌نیازها

قبل از شروع کار با گیت لازم ا ست مخت صر آ شنایی با د ستورات خط فرمان در سی ستم عامل خود دا شته با شید. اگر از سی ستم‌عامل‌های خانواده‌ی یونیکس (مثل لینوکس و OSX) ا ستفاده می‌کنید و با محیط ترمینال و د ستورات اولیه‌ی آن آ شنا نی ستید، می‌توانید از ضمیمه‌ی ۱ کتاب برای آ شنایی با د ستورات خط فرمان ا ستفاده کنید.اگر هم از ویندوز ا ستفاده میکنید سعی کنید گیت را به همراه بسته‌ی خط فرمان یونیکس نصب کنید یا با دستورات پایه خط فرمان ویندوز آشنا شوید. توجه داشته باشید که این آشنایی بیشتر از چند دقیقه از شما زمان نمیگیرد و شما تنها نیاز دارید ساختار دستورها و دستورات اولیه جابجایی بین دایرکتوری ها را بدانید.

نصب گیت

برای نصب گیت میتوانید در وبگاه رسمی آن به صفحه‌ی دانلود بروید و نسخه‌ی سیستم‌عامل خود را دانلود و سپس نصب کنید، با اینحال در برخی سیستم‌عامل ها همچون لینوکس، اینکار با جزییاتی همراه است که البته برای کاربر آشنا به اکوسیستم آن سیستم‌عامل دشواری محسوب نمی‌شود.

نصب روی لینوکس

نصب گیت روی توزیع‌های مختلف لینوکس به شیوه‌های مختلف امکان پذیر است که متداول‌ترین روش، نصب از روی مدیربسته‌های توزیع مورد نظر است.

دستورات نصب از مدیربسته ها در توزیع های معروف:

نصب روی دبیان و اوبونتو:

```
apt-get install git
```

نصب روی جنتو:

```
emerge --ask --verbose dev-vcs/git
```

نصب روی آرچ‌لینوکس:

```
pacman -S git
```

نصب روی اوپن‌زوزه:

```
zypper install git
```

نصب روی فدورا (تا نسخه‌ی ۲۱):

```
yum install git
```

نصب روی فدورا(نسخه‌ی ۲۲ و بالاتر):

```
dnf install git
```

نصب روی mac os

نصب روی osx با پکیج منیجر هوم برو:

```
brew install git
```

نصب روی ویندوز

برای نصب گیت روی سیستم‌عامل ویندوز، به صفحه‌ی دانلود گیت در وبگاه رسمی پروژه بروید و نسخه‌ی ۳۲ یا ۶۴ بیتی آن را، متناسب با معماری مورد استفاده‌ی سیستم‌عامل خود دانلود و نصب کنید.

کانفیگ

بعد از نصب گیت لازم است تنظیماتی را انجام دهید و همچنین خودتان را به گیت معرفی کنید.اینکار بخاطر ثبت تغییرات لحاظ شده توسط شما به نام شما و همچنین شخصی‌سازی‌کردن ویژگی‌های گیت است.

تنظیمات گیت در سه سطح قابل انجام است system : ، local،global

- سطح :system اگر تنظیمات را در این سطح انجام دهید تنظیمات شما روی مخازن تمام کاربرهای آن سیستم اعمال خواهد شد. این تنظیمات تغییرات در `/etc/config ` ذخیره می‌شوند.
- سطح :global اگر تنظیمات را در سطح global انجام دهید تنظیمات شما روی مخازن کاربر فعلی سیستم اعمال خواهد شد. این تنظیمات در *~/.gitconfig* یا *~/.config/git/config* ذخیره می‌شوند.
- سطح :local تنظیمات در این سطح تنها روی مخزنی که در آن قرار دارید اعمال می‌شود. این تنظیمات در دایرکتوری مخزن در فایل *git/config*.ذخیره می‌شود.

تنظیمات اساسی در گیت ازین قرار هستند:

برای اضافه کردن نام خود دستور زیر را بزنید:

```
git config --global user.name "نام شما"
```

و برای ثبت ایمیل خود دستور زیر را استفاده کنید:

```
git config --global user.email آدرس ایمیل شما
```

توجه کنید که این دو تنظیم بعد از نصب گیت اجباریست و اگر انجام ندهید در مراحل بعد کار با گیت با مشکل مواجه می‌شوید.

و البته تنظیمات زیادی نیز به اختیار می‌توانید انجام دهید برای مثال برای تنظیم ویرایشگر پیش‌فرض، برای استفاده توسط گیت از دستور زیر استفاده می‌کنیم:

```
git config --global core.editor ویرایشگر
```

برای مثال برای تنظیم ویرایشگر ایمکس به عنوان ویرایشگر پیش‌فرض از دستور زیر استفاده میکنیم:

```
git config --global core.editor emacs
```

توجه کنید که شما بجای global میتوانید هرکدام از سطوح معرفی شده در بالا را استفاده کنید اما توصیه همان global است مگر در شرایط خاص. با این روش تنظیم گیت تنها یکبار و برای همیشه برای کاربر فعلی سیستم انجام شده است.

اگر می‌خواهید محتویات یک تنظیم خاص برای مثال user.name را ببینید دستور زیر را استفاده کنید:

```
git config user.name
```

و برای بررسی تمام تنظیماتی که انجام داده‌اید دستور زیر را استفاده کنید:

```
git config --list
```

برای راهنمایی بیشتر می‌توانید از صفحات راهنما برای مشاهده‌ی تنظیمات بیشتر گیت استفاده کنید:

```
man git-config
```

aliasها

در گیت دستورات مختلفی داریم از جمله دستور *config* که با آن آشنا شدید اما این دستورات برای استفاده به تعدد در طول روز ممکن است گاهی طولانی و خسته کننده بنظر بیایند. طراحان گیت امکانی تحت عنوان *alias* در تنظیمات گیت گنجانده‌اند که میتوانید برای دستورات پرکاربرد خود نام‌های مستعاری تعریف کنید و با آن‌ها دستور خود را اجرا کنید مثلا بجای *git init* بنویسید *git i*.

برای ساخت این *alias*ها از قالب زیر استفاده کنید:

```
git config --global alias.i init
```

که در این دستور نام مستعار i برابر اجرای دستور *init* بصورت سراسری قرارداده شد که زین‌پس بعد از اجرای دستور *git i* دستور *git init* اجرا میشود و میتوانید دستورات دیگر را هم با همین روش و به هر نامی که میخواهید برای خود تنظیم کنید.

تنظیم گیت برای زبان فارسی

وقتی که پرونده‌های فارسی را به گیت اضافه می‌کنیم و فرمان git status را اجرا می‌کنیم، گیت نام این پرونده‌ها را به درستی نمایش نمی‌دهد. برای رفع این مشکل از دستور زیر استفاده کنید.

```
git config --global core.quotepath false
```

مطالعه بیشتر

در صفحات راهنمای گیت در ساختار یونیکس میتوانید اطلاعات بیشتری در این مورد بیابید. صفحه‌ی راهنمای تنظیمات گیت در سیستم‌های خانواده‌یونیکس را با دستور *man git-config* بخوانید.

شروع یک پروژه با گیت

دایرکتوری پروژه هایی که از سیستم‌های مدیریت کد منبع مثل گیت استفاده می‌کنند، با عنوان یک) repository در اصطلاح عامیانه ریپو، گاهی در فارسی مخزن) می‌خوانند. برای شروع کار با گیت در یک دایرکتوری به عنوان یک مخزن گیت، لازم است که ابتدا به گیت بگویید که میخواهید این دایرکتوری یک مخزن گیت باشد .دستور *git init* یک مخزن جدید گیت ایجاد می‌کند. مخزنی که درون آن میتوانید از امکانات گیت استفاده کنید و دستورات را در آن اجرا کنید .با اجرای این دستور یک دایرکتوری با نام *git/* .درون دایرکتوری حاضر شما ایجاد میشود که حاوی فایلهای کانفیگ و فایلها و بلابهای تغییرات ثبت شده توسط گیت است.

برای اضافه کردن گیت به یک پروژه، داخل دایرکتوری پروژه دستور زیر را وارد کنید:

```
git init
```

یا برای ساختن یک پروژه‌ی جدید دستور زیر را استفاده می‌کنیم:

```
git init نام‌دایرکتوری
```

که این دستور یک دایرکتوری با نامی که وارد کرده‌اید می‌سازد و گیت را به آن اضافه می‌کند.

وقتی که گیت به یک دایرکتوری اضافه می‌شود درحقیقت یک دایرکتوری درون دایرکتوری پروژه به نام git. ساخته می‌شود که فایل هایی که سیستم گیت می‌سازد، در آن قرار می‌گیرند که بعداً بیشتر با آن آشنا می‌شوید. با این دستور پروژه‌ی شما به یک مخزن گیت تبدیل می‌شود.

نمایش تغییرات

یکی از اساسی‌ترین کارهایی که با گیت می‌کنیم بررسی و ثبت تغییرات ایجاد شده در فایل‌های متنی پروژه است.

هر زمان که تغییرات ثبت نشده‌ی پروژه را مایل بودید بررسی کنید، دستور زیر را وارد کنید:

```
git status
```

این دستور فایل‌هایی که تغییر داده‌اید را برای شما لیست می‌کند .تغییرات شامل: ویرایش یک فایل، اضافه‌کردن فایل جدید یا حذف یک فایل و همچنین فایلهای استیج شده یا استیج نشده که دو مورد اخر را بعدا بررسی خواهیم کرد.

برای نمایش استاتوس به‌صورت خلاصه از گزینه ی *s*-با آن استفاده کنید:

```
git status -s
```

اضافه‌کردن فایل‌ها

ثبت تغییرات در گیت بصورت معمول دو مرحله دارد .1 :اضافه کردن فایل‌های تغییر داده شده‌ی مورد نظر .2ثبت تغییرات با یک پیام یا توضیح

برای اضافه‌کردن فایل‌ها به گیت از دستور add استفاده می‌کنیم. به‌صورت زیر:

```
git add فایل۳ فایل۲ فایل۱
```

و برای افزودن تمام فایل‌های تغییرداده‌شده دستور را به‌صورت زیر استفاده می‌کنیم:

```
git add -A
```

د ستور git status که پی شتر تو ضیح داده شد، برای یافتن فایل‌های تغییر داده شده و ا ضافه‌کردن آن‌ها به شما کمک می‌کند.

ثبت تغییرات

برای ثبت تغییرات یا به اصطلاح کامیت‌کردن تغییرات، از دستور commit استفاده می‌کنیم. در این مرحله فایل‌هایی را که با دستور add به حالت stage برده‌ایم در سیستم گیت ثبت می‌کنیم. برای ثبت هر تغییر نیاز است یک پیام هم با آن ثبت شود تا معلوم شود در این قسمت از تغییرات لحاظ شده چه کار کرده‌ایم، یا چه تغییراتی داده‌ایم.

برای مثال اگر ما یک فایل متنی برای نو شتن تو ضیحات پروژه به نام readme.md ساخته با شیم و با د ستور add آن را برای کامیت‌شدن آماده کرده باشیم، می‌توانیم به همراه کامیت خود یک پیام با مضمون add read me file ثبت کنیم که تغییرات، برای مطالعه در آینده شفاف‌تر باشند.

دستور کامیت بصورت زیر است:

```
git commit
```

که با اجرای این د ستور ویرایش‌گرِ فایل شما باز شده و میتوانید پیام خود را در آن بنویسید و ذخیره کنید و به این صورت تغییرات فایل شما ثبت می‌شود و یک کامیت صورت می‌گیرد.

یک راه متداول‌تر هم که برای پیام‌های کامیت یک خطی کاربرد دارد، بصورت زیر است:

```
git commit -m پیام شما
```

که در این روش کامیت شما با همان پیام ذخیره شده، و نیازی به باز شدن ویرایش‌گر متن ندارید.

قرار داد استاندارد برای پیام کامیت . 1 : در زمان حال نوشته شود . 2در هنگام استفاده از سوییچ m- کمتر پیام کمتر از ۵۰ کاراکتر باشد.

تغییر محتوای آخرین کامیت

فرض کنیم که ا شتباهی در نو شتن پیام یک کامیت دا شته اید و یا به هر دلیل دیگر قصد تغییر پیام آخرین کامیت را دارید و این مورد را پس از انجام کامیت متوجه شـــده اید. برای تغییر دوباره‌ی پیام آخرین کامیت از گزینه‌ی *amend*–به همراه دستور کامیت استفاده میکنیم. مانند مثال:

```
git commit --amend -m "new commit message"
```

حذف فایل‌ها

اگر پس از ثبت یک فایل آن فایل را حذف کنیم، برای ا ضافه کردن تغییرات فایل حذف شده، دیگر امکان ا ستفاده از add را نداریم و باید با دستور rm آن فایل را حذف کنیم:

```
git rm فایل۳ فایل۲ فایل۱
```

دقت دا شته با شید برخلاف د ستورadd ، ا ستفاده از . git rm تمام فایل‌های پروژه را حذف می‌کند نه فقط فایل‌های حذف شده را پس از آن استفاده نکنید(: .

اگر اشـــتباهاً فایل‌های زیادی را دســـتی پاک کردیم بدون اینکه از دســـتور git rm اســـتفاده کنیم، لازم اســـت که تک‌تک فایل‌هایی را که دستی پاک کرده‌ایم را با دستور git rm به حالت stage در آوریم. یا اینکه از دستور زیر استفاده کنیم:

```
git rm $(git ls-files --deleted)
```

اگر بین نام فایل‌های پاک شده، نویسهٔ فاصله وجود داشته باشد از دستور زیر استفاده می‌کنیم:

```
git ls-files --deleted -z | xargs -0 git rm
```

نمایش تغییرات

برای برر سی تغییرات صورت گرفته در چند کامیت یا تغییرات کامیت‌ نشده در مقابل تغییرات کامیت شده از د ستور diff استفاده می‌کنیم.

حالت ساده‌ی استفاده از این دستور بصورت زیر است:

```
git diff
```

که این دستور تغییرات کامیت نشده‌ی تمام فایل‌ها را به شما نشان می‌دهد .همچنین به‌صورت زیر می‌توانید تغییرات یک فایل خاص را نیز ببینید:

```
git diff نام‌فایل
```

لاگ

برای نمایش لیست کامیت‌ها از دستور log استفاده می‌کنیم. با این دستور می‌توان لیست کامیت‌های یک برنچ خاص، یا تمامی برنچ‌ها یا مقایسه‌ی بین برنچ‌ها را دید.

شکل کلی این دستور بصورت زیر است.. code-block:: bash :

```
git log
```

که لیستی از تمام کامیت‌ها را به شما نمایش می‌دهد.

می‌توانید کامیت‌های یک برنچ مشخص را نیز بصورت جداگانه با مشخص‌کردن نام برنچ ببینید:

```
git log نام‌برنچ
```

چند گزینه‌ی کاربردی

- از *graph*–برای نمایش کامیت‌ها و برنچ‌ها بصورت گراف تغییرات استفاده می‌کنیم.
- از *grep=<pattern>*–برای جستجو در میان کامیت‌ها با استفاده از رجکس‌ها استفاده می‌کنیم.
- از *merges*–برای نمایش کامیت‌هایی که در ادغام برنچ‌ها ثبت شده‌اند استفاده می‌کنیم.
- از آپشن `oneline–` برای نمایش هر کامیت در یک خط استفاده می‌کنیم.

توضیحات بیشتر در این مورد در صفحه ی راهنمای گیت.*git log –help* :

برای نمایش خلاصه و منظم لاگ ها هم میتوانید از دستور زیر استفاده کنید:

```
git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit
```

ممکن ا ست در دایرکتوری پروژه فایل‌هایی دا شته با شید، که نخواهید گیت آن‌ها را در ا ستاتوس‌ها نشان دهد، و همچنین نخواهید در مخزن ا صلی ا ضافه شوند.برای این‌کار باید در دایرکتوری پروژه یک فایل به نام gitignore. بـ سازید و در آن، لیستِ فایل‌ها و دایرکتوری‌هایی را که گیت باید نادیده بگیرد را بنویسید.

برای مثال لیست زیر:

```
# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

```
eggs/
lib/
lib64/
parts/
sdist/
var/
*.egg-info/
.installed.cfg
*.egg
```

لیستی از فایل‌ها و دایرکتوری‌ها ست، که گیت بعد از ساخته شدن فایل gitignore. نادیده میگیرد . توجه کنید که خود فایل gitignore. باید توسط دستور add به پروژه اضافه و کامیت شود . توجه کنید در این فایل، خطوطی که با # شروع می‌شوند، به عنوان توضیحات (کامنت) در نظر گرفته می‌شوند.

برَنچ‌ها

برنچ‌ها شاخه‌های مختلفی را برای توسعه ایجاد می‌کنند. فرض کنید که در حال توسعه یک اپلیکیشن هستید و قصد دارید نسخه‌ی آینده اپلیکیشن خود را همزمان با نسخه‌ی فعلی توسعه دهید. اضافه‌کردن تمام این تغییرات با هم ممکن است مسبب شلوغی و بی‌نظمی روند توسعه، و همچنین تداخل فایل‌های هم‌نام شود .با استفاده از برنچ‌ها در گیت می‌توانید یک مسیر جدید برای توسعه هر ویژگی ایجاد کنید و همچنین میتوانید در پایان ویژگی‌های کامل‌شده را به برنچ اصلی اضافه کنید. در گیت به‌طور پیش‌فرض، برنچ اصلی به‌نام master است.

برای ساختن یک برنچ توسعه‌ی جدید، از دستور زیر استفاده کنید code-block:: bash .. :

```
git branch نام‌برنچ
```

برای نمایش لیستی از برنچ‌ها از دستور زیر استفاده کنید code-block:: bash .. :

```
git branch
```

برای آماده به‌کارکردن یک برنچ از دستور زیر استفاده کنید:

```
git checkout نام‌برنچ
```

همچنین برای حذف یک شاخه از دستور زیر استفاده می‌شود:

```
git branch -d نام‌برنچ
```

و برای ساخت یک برنچ و هم‌زمان به حالت آماده به کار رفتن آن برنچ از دستور زیر می‌توانید استفاده کنید:

```
git checkout -b نام‌برنچ
```

برای پاک‌کردن برنچ مخزن ریموتی که قبلاً با آپشن d- آن را از مخزن محلی پاک کرده‌ایم دستور زیر را به کار برید:

```
git push origin :<branch-name>
```

ترکیب

در بخش قبل نحوه‌ی استفاده از برنچ‌ها گفته شد. حال اگر بخواهیم این شاخه‌های جدا را دوباره با شاخه‌ی اصلی ترکیب کنیم، باید از دستور merge استفاده کنیم:

```
git merge شاخه
```

با این دستور محتویات شاخه‌ی داده شده به دستور، با شاخه‌ی فعلی ترکیب می‌شود.

در صورتی که بخواهیم تاریخچهٔ کامیت‌های شاخهٔ مورد نظر در شاخهٔ اصلی وارد نشود و فقط با شاخهٔ اصلی ترکیب شود از آپشن squash– به صورت زیر استفاده می‌کنیم:

```
git merge <branch_name> --squash
```

در صورتی که عملیّات مرج را روی فایل مشخصی اشتباه انجام دادید، برای اینکه بتوانید دوباره دستور git mergetool را اجرا کنید از دستور زیر برای گرداندن به وضعیّت اوّل استفاده کنید.

```
git checkout -m <filename>
```

استفاده از ریموت

معمولا پروژه‌هایی که چندین نفر روی آن کار می‌کنند مخزن اصـلی را در یک سـرور قرار می‌دهند و افراد همه‌ی تغییرات را روی آن مخزن اضافه می‌کنند و سایرین همیشه تغییرات را برای هماهنگی با یکدیگر دریافت می‌کنند.

می‌توانید از سـرویس‌های اینترنتی مانند گیت‌هاب، بیت‌باکت و بسـیاری سـرویس‌های دیگر اسـتفاده کنید یا سـرور گیت خودتان را راه بیاندازید، اما بحث این قسمتِ ما، ارتباط با مخازن ریموت است نه ساخت آن‌ها.

remote

برای نمایش، اضافه و حذف کردنِ تغییرات در یک مخزن ریموت از دستور remote استفاده می‌شود.

برای نمایش لیستی از ریموت‌ها از دستور زیر استفاده میشود:

```
git remote
```

که این دستور ریموت های موجود در مخزن را لیست می‌کند.

برای اضافه کردن یک مخزن از دستور remote add بصورت زیر استفاده میشود.. code-block:: bash :

```
git remote add [name] [url]
```

که در این دستور بجای[name] ، یک نام برای ریموت مورد نظر و به‌جای [url]، آدرس مخزن مورد نظر را قرار می‌دهیم.

برای حذف یک ریموت رپوزیتوری هم دستور بالا را بصورت زیر تغییر می‌دهیم:

```
git remote rm [name]
```

push

برای افزودن تغییرات کامیت‌شده در مخزن به مخزن ریموت از دستور push استفاده می‌کنیم به‌صورت زیر:

```
git push [remote-repo-name] [branch-name]
```

که به جای remote-repo-name نام ریموت سرور مورد نظر (که قبلا باید ا ضافه کرده با شید) و به جای branch-name نام شاخه‌ی مورد نظر که تغییرات را در آن لحاظ کرده‌ایم.

pull

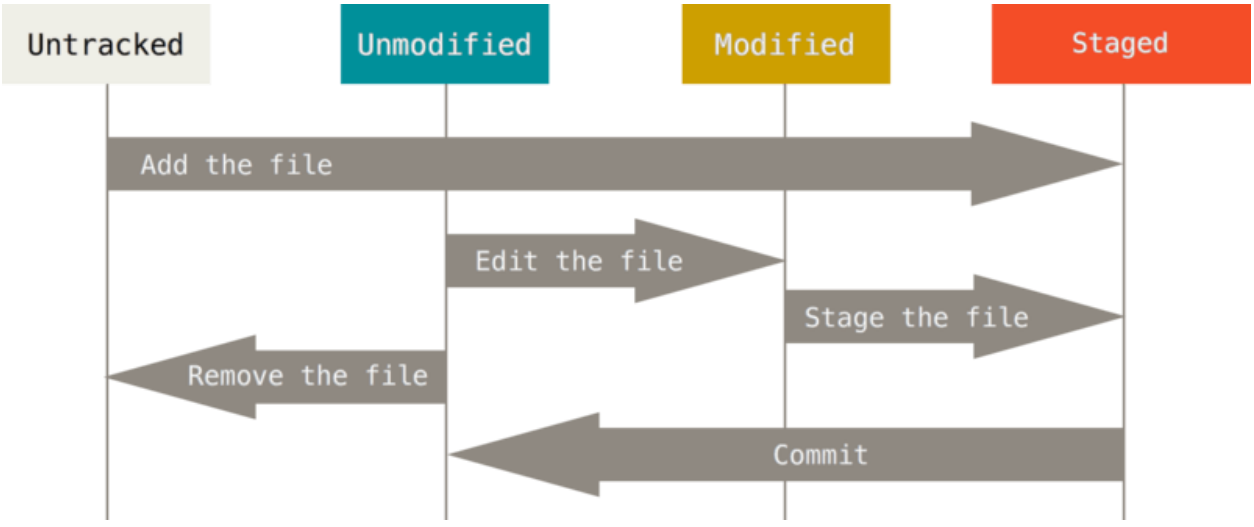
برای دریافت تغییرات کامیت‌شده به مخزن ریموت از دستور pull بصورت زیر استفاده می‌کنیمbash ::code-block .. :

```
git pull [remote-repo-name]
```

که به‌جایremote-repo-name ، نام ریموت سرور مورد نظر را می‌نویسیم.

بازگردانی تغییرات

]!چرخه‌ی زندگی در گیت



تصویر بالا نمایانگر چرخه‌ی تغییرات در گیت است. فایل‌ها در گیت تنها در دو حالت می‌توانند باشند: ثبت‌شده و ثبت‌نشده. تغییرات ثبت‌شده، تغییراتی است که پیش‌ازاین به گیت اضافه شده‌اند که می‌توانند تغییر نکرده، تغییر کرده یا آماده‌ی ثبت باشند و هر فایلی خارج ازین حالت‌ها به‌عنوان ثبت‌نشده در نظر گرفته می‌شود.

ریست فایل های استیج شده

در حالتی که چند فایل تغییر یافته را به مخزن add یا rm میکنیم، آن فایل ها به حالت اسـتیج شـده میروند.اگر بخواهیم یک فایل را از این حالت به حالت قبل خود یعنی حالت اضافه نشده برگردانیم از دستور زیر استفاده می‌کنیم:

```
git reset HEAD نام فایل
```

سرویس‌های میزبانی مخازن گیت

شما می‌توانید گیت سرور را در سرورهای خود راه اندازی کرده و پروژه های خود را مدیریت کنید اما سرویس‌های زیادی برای میزبانی مخازن گیت، با امکاناتی بسیار ویژه وجود دارند که کار شما را برای استفاده‌ی ریموت مخازن آسان می‌کنند.

در این قسمت چند سرویس معروف و معتبر را بررسی می‌کنیم.

گیت‌هاب

گیت‌هاب به آدرس معروف‌ترین ســـرویس میزبانی پروژه‌های گیت اســـت که پروژه‌های بزرگ و معتبر زیادی مانند [کرنل لینوکس](https://github.com/torvalds/linux)، جی کوئری[https://github.com/jquery/jquery])، بوت استرپ [https://github.com/twbs/bootstrap](و بسیار پروژه‌ی دیگر روی گیت‌هاب توسعه داده می‌شوند.

رابط کاربری خوب و برنامه‌های سوم شخص و api قدرتمند گیت‌هاب دلیل خوبی برای انتخاب آن ا ست، اما توجه دا شته باشید که این سرویس برای مخازن خصوصی رایگان نیست اما استفاده از آن برای پروژه های متن باز کاملا رایگان است و جامعه‌ی کاربری متن‌باز قدرتمندی دارد.

بیت‌باکت

بیت‌باکت هم سـرویس‌دهنده ی دیگری اسـت که از مخازن گیت و مرکوریال پشـتیبانی می‌کند و امکان جالب آن رایگان بودن پروژه های خصوصی برای افراد یا تیم های کوچک است.

تگ‌ها

برای تگ‌زدن یک کامیت، از دستور زیر استفاده کنید:

```
git tag -a v2.0 -m 'version v2.0'
```

و برای ارسال همهٔ تگ‌ها به مخزن از راه دور، فرمان زیر را وارد کنید.

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

و از فرمان زیر برای ارسال یک تگ مشخص استفاده کنید:

```
git push origin <tag_name>
```

پاک‌کردن یک تگ از مخزن محلّی:

```
git tag -d <tag_name>
```

پس از این‌که تگ محلّی پاک شد، می‌توانیم تگ را از روی مخزن origin به روش زیر پاک کنیم:

```
git push origin :tagname
```

ترفندهای گیت

برای آن‌استیج کردن فایل‌ها از دستور زیر استفاده کنید:

```
git reset HEAD <file-name>
```