

GIT

گیت چیست؟

(VCS) یک سیستم کنترل نسخه است.

گیت چند بخش دارد:

۱- سیستم کنترل:

گیت یک ردیاب محتوا است ، بنابراین می توان از گیت برای ذخیره محتوا استفاده کرد و بیشترین کاربرد هم در این بخش است.

۲- سیستم کنترل نسخه:

کدهایی که در گیت زده می شوند با افزودن یا کاستن تغییر میکنند و همچنین بسیاری از توسعه دهندگان میتوانند به صورت موازی با گیت کد بزنند.

۳- سیستم کنترل نسخه توزیع شده:

گیت دارای یک مخزن از راه دور است که در یک سرور و مخزن محلی ذخیره میشود که بطور معمول شما میتوانید در رایانه شخصی هر توسعه دهنده ببینید و به این معنی است که کد ها در یک سرور مرکزی ذخیره نمیشود و به صورت کامل در رایانه شخصی کپی میگردد.

رابط های گیت:

گیت در ترمینال لینوکس اجرا میشود و از انجایی که متن باز است و به خوبی طراحی شده ، روش های دیگری برای اجرا شدن آن بکار گرفتند که عبارت است از :

۱-gitHub , ۲-bitBucket , ۳-savannah , ۴-gitLab , ۵-sourceForge

دستورات گیت:

Git config

برای پیکربندی نام و ایمیل کاربر استفاده میشود

۱-git init

با زدن این دستور یک مخزن یا ریپازیتوری ایجاد میکنیم که گیت تمام دادههای داخلی را درونش ذخیره میکند.

۲-git clone repository url

با زدن این دستور یک کپی کامل از مخزن گرفته و به دایرکتوری سیستم شما انتقال میدهد و شما میتوانید تغییرات دلخواه را در آن ایجاد کنید.

۳-git status

آخرین وضعیت مخزن ما را به ما نشان میدهد به طور مثال اگر یک فایل دید ایجاد کنیم به ما نام آن فایل و در لیست ناشناخته ها قرار میدهد.

۴-git add

با زدن این دستور فایل مشخص شده به مخزن گیت اضافه میشود.

۵-git commit

با زدن این دستور چگونگی تغییرات انجام شده را اعمال کنید.

۶-git diff

با زدن این دستور تغییرات ایجاد شده را نسبت به نسخه اولیه به ما نمایش میدهد و میگوید چه تغییراتی ایجاد شده است.

۷-git rm

با زدن این دستور فایل مشخص شده را حذف میکند.

۸-git log

با زدن این دستورات تاریخچه کامل کامیت ها را در توسعه پروژه مشاهده میکنید و خروجی شامل ای دی ، نویسنده ، تاریخ و پیغام به هر کامیت است.

۹-git remote

با استفاده از کد بالا یک مخزن محلی را به سرور آنلاین وصل میکند و کاری که بخواهیم روی آن انجام بدهیم را اعمال میکنیم.

۱۰-git push

با این دستور تغییرات خود را به سمت گیت به اصطلاح هل می دهید.

۱۱-git pull

آخرین نسخه مخزن را به سمت سیستم شخصی خودتان میکشید.

۱۲-git branch

با زدن این دستور به شما اجازه میدهد تا یک شاخه جدید از پروژه اصلی ایجاد کنید و تغییرات و اضافات فایل ها مختص خودتان باشد.

۱۳-git checkout

با زدن این دستور به شما اجازه میدهد تا محتوا شاخه ای که در آن نیستید را بررسی کنید.

۱۴-git merge

هنگامیکه کار با یک شاخه تموم شد و موفقیت آمیز بود میتوانید آن را با شاخه اصلی یکی کنید و در دسترس سایر دوستان قرار بدید.

۱۵-git fetch

با این دستور تنها تغییرات دریافت کرده و ذخیره نمیکند.

گیت هاب چیست؟

گیت هاب یک شبکه اجتماعی در فضا ابری برای برنامه نویسان و یک پلتفرم همکاری برای توسعه دهندگان است که بزرگترین فضای ذخیره سازی برای کارهای اشتراکی و توزیع یافته دنیا به شمار میرود.

مفاهیم مهم گیت هاب:

Repository:

معادل فارسی آن مخزن یا انبار است ، محلی که تمام فایل های یک پروژه در آن ذخیره میشود به عبارتی دیگر هر پروژه یک مخزن مخصوص به خود دارد که با یک آدرس مشخص میگردد.

Fork:

معادل فارسی آن شاخه یا شعبه است ، از آن برای این استفاده میشود که براساس یک پروژه اصلی یک پروژه دیگر ایجاد کرد و تغییراتی را روی آن ایجاد کنیم و در صورت نیاز آن را به عنوان مخزن دوباره انتشار دهیم.

Branch:

بعضی وقت ها به جای استفاده از فورک از برنچ استفاده میکنیم که معادل دقیق همان است و ما در کنار شاخه اصلی یک شاخه برای خود ایجاد میکنیم و تغییرات و اعمال میکنیم و در آخر با شاخه اصلی یکی میکنیم.

Commit:

به زبان ساده هر تغییر یک کامیت محسوب میشود که هرکدام از انها شامل یک توصیف برای علت تغییر است.

Pull request:

درواقع به معنای ادغام و یکپارچگی است ، زمانی بکار میرود که شما میخواهید یک پروژه منشعب شده را که تغییراتی روی آن اعمال کردید در معرض دید سایر برنامه نویسان قرار بدهید و در صورت تایید در پروژه اصلی قرار بگیرد.

سوابق تغییرات :

از ویژگی های مهم گیت هاب این است که سوابق کامل تغییرات در پروژه را حفظ میکند و دیگر برنامه نویسان نگران این نیستند که چه کسی؟ چه چیزی؟ را و چطور؟ تغییر داد.

جنبه های شبکه اجتماعی:

امکانات شبکه اجتماعی که گیت هاب دارد آن را بسیار قدرتمند و متمایز میگرداند در گیت هاب هر شخصی یک رزومه و یک پروفایل مخصوص به خود دارد که همه آثار و فعالیت هایش در آن قابل مشاهده است و همچنین بازبینی در پروژه ها را میتوان به صورت عمومی مورد بحث قرار داد.

تفاوت گیت و گیت هاب :

گیت یک سیستم کنترل نسخه است در حالی که گیت هاب یک پلتفرم میزبانی آنلاین برای خدمات مختلف است.

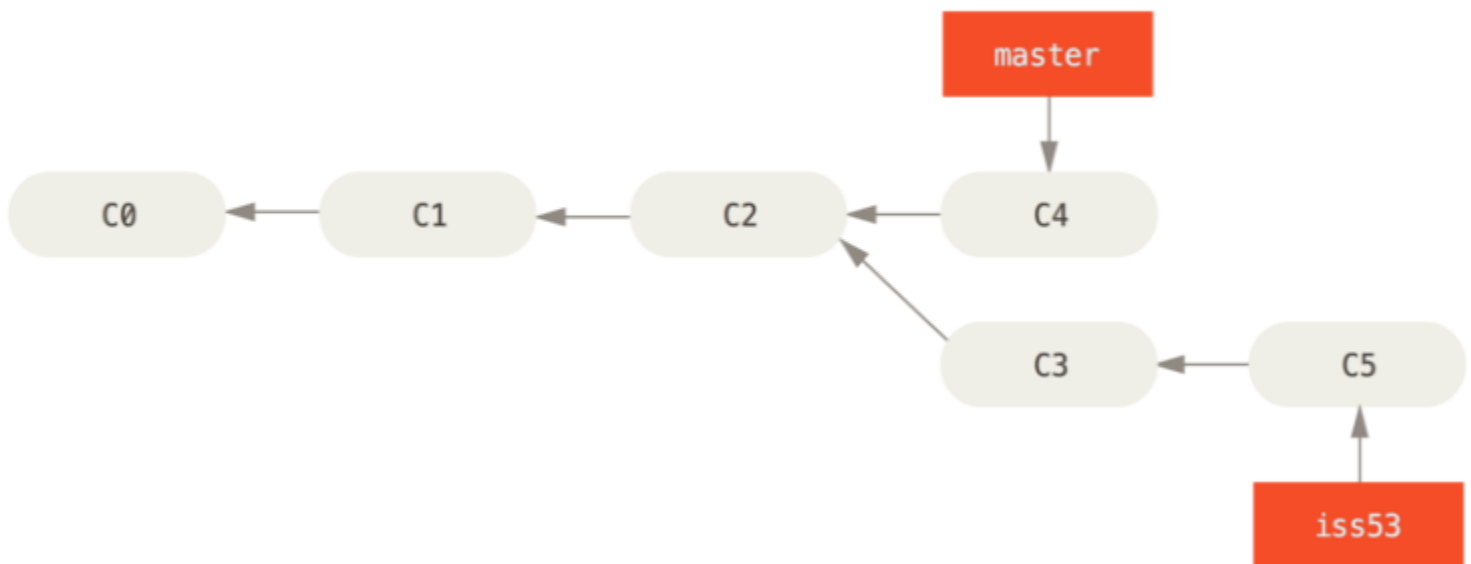
گیت لب :

گیت لب نیز مانند گیت هاب است با این تفاوت که در گیت لب شما اجازه ساخت پروژه و مدیریت پروژه خصوصی و به صورت رایگان به شما میدهد.

کار با شاخه و استفاده از مفهوم Branch , merge

بعضی وقت ها در توسعه نرم افزار لازم است که تغییرات گسترده ای در کد ایجاد شود، فیچری اضافه شود یا کدی به صورت تست به برنامه اضافه شود که گیت برای این مسئله راه حلی در نظر گرفته است.

بطور معمول ما هر کامیتی که انجام میدیم روی شاخه اصلی پروژه اعمال میشه، گیت با استفاده از برنج یه شاخه جدا از شاخه اصلی ایجاد میکنه و هر تغییراتی که داریم روی آن شاخه اعمال میکنیم و در صورت تمایل در انتها با شاخه اصلی یکی میکنیم



در تصویر بالا شاخه ای جدید "iss53" از شاخه اصلی در کامیت C2 جدا شده و کار روی هر دو برنج بدون دخالت در کد همدیگه جلورفته . کامیت های C3 و C5 از تغییراتی که در C4 اتفاق افتاده است بی خبر است و بالعکس.

Branch & checkout

برای ساخت یک شاخه از دستور `git branch gholi` استفاده میکنیم و یک شاخه به اسم قلی در کامیتی که هستیم ایجاد میکند ولی وارد آن شاخه نمی شود برای وارد شدن به آن شاخه باید از دستور `git checkout gholi` استفاده کنیم

Merge

بعد از اتمام کار و نهایی شدن کد لازم است که آن را با شاخه ای دیگه که معمولا `master` است یکی کنیم یا ممکن است تغییراتی که روی شاخه سومی است و وارد شاخه ای که درونش هستیم اعمال کنیم در این مواقع از `merge` استفاده میکنیم

برای استفاده از این دستور ابتدا باید وارد شاخه مقصد بشیم با دستور `checkout` و سپس با دستور مناسب دو شاخه را یکی کنیم

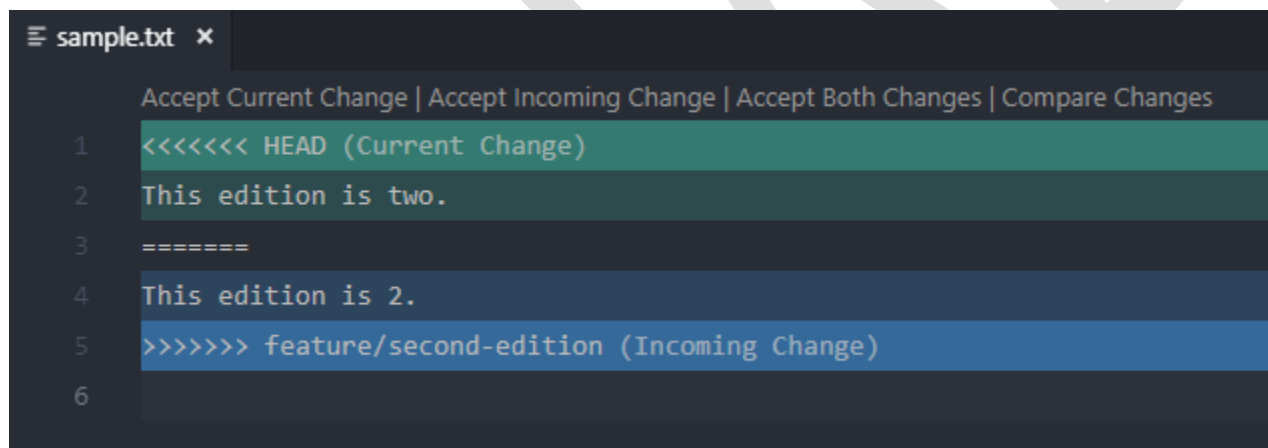
`Git checkout master`

Git marge gholi

شاخه قلی را با شاخه مستر یکی میکند

Marge conflict

فرض کنید که به صورت تیمی روی یک پروژه کار میکنید و همکارتون یک تغییر داخل پروژه انجام داده و اون رو push کرده و شما هم در همین لحظه در local خود یک تغییری روی پروژه انجام دادید و از تغییرات همکارتون خبر ندارید ، بعد از مدتی همکارتون از شما درخواست اطلاعات جدید میکنه و شما هم pull میکنید و پروژه بروز میشود و در این لحظه marge conflict اتفاق می افتد و گیت گیج میشه که آخرین تغییرات شما رو نگه داره یا همکارتون !!!



```
sample.txt x
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<<< HEAD (Current Change)
2 This edition is two.
3 =====
4 This edition is 2.
5 >>>>>> feature/second-edition (Incoming Change)
6
```

موقعی که به marge conflict بر بخوریم با صفحه ای مانند صفحه بالا رو به رو میشویم که ۴ گزینه پیش روی ما هست:

Accept current change با انتخاب این گزینه اون تغییری که در شاخه فعلی هست انتخاب میشه.

Accept incoming change با انتخاب این گزینه اون تغییری که در شاخه دیگر هست انتخاب میشه.

Accept both changes با انتخاب این گزینه، هر دو تغییر بصورت همزمان نگه داشته میشن.

اگر هیچکدام از موارد رو نخواستید، میتونین به راحتی از خط ۱ تا خط ۵ رو بصورت کامل حذف کنید. با این کار هم merge conflict برطرف میشه.

<https://Vlearn.com/tutorials/how-to-resolve-merge-conflict-in-git>

tag

برای ورژن بندی نرم افزار و برنچ ها از تگ استفاده میکنیم

به صورت روبه رو `Git tag -a v۲,۰ -m 'version v۲,۰'`