برای استفاده از گیت میتوان دستورات مربوط به گیت را در ترمینال وارد کرد. برای کار با گیت ابزارهای گرافیکی هم موجود هستند اما کار با ترمینال سرعت و قدرت بیشتری دارد. اولین کاری که بعد از نصب گیت باید انجام دهید، شناساندن خودتون به گیت هستش. برای اینکار از دستور config استفاده میشود. دستور config وظیفه ایجاد و تغییر optionها بروی پروژه را دارد. زمانی پروژه جدیدی ایجاد میکنید میتوانید با دستور init به گیت بفهمانید که تغییرات اعمال شده بروی این پروژه باید تحت کنترل باشند. زمانی که با دستور init یک پروژه را به گیت فهماندید، گیت آن پروژه را به یک مخزن(repository) تبدیل میکند. مخزن پروژه ای است که git تغییرات آن را کنترل میکند.

Git help ‘command’

این دستور توضیحاتی درباره command مورد نظر میدهد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git config user.name “sajjad”

با استفاده از این دستور تنها در پروژه جاری نام شمارو set میکند

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git config --global user.name “sajjad”

این دستور به صورت global نام شما را در تمام پروژه ها set میکند

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

بعد از دستورات باید ایمیل خود را نیز وارد کنید :

Git config -- global user.email [sajjad@gamil.com](mailto:sajjad@gamil.com)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git config –list

نمایش optionها و configهایی که set کرده اید ماننده اسم و ایمیل

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git init

ساخت یک repository خالی برای پروژه ی شما . بعد از زدن این دستور خود گیت پوشه ی جدیدی به نام .git را میسازد که config های این پروژه را در آن ذخیره میکند

=====================================================

زمانی که تغییری در پروژه شما ایحاد میشود میتوانید با کمک دستور git status تغییرات رو ببینید. بعد از اینکه گیت تغییرات پروژه شما رو فهمید اگر از تغییرات راضی بودید باید اقدام به ثبت تغییرات کنید. در گیت سه مرحله برای ثبت تغییرات وجود دارد. مرحله اول untracked یا modified است که نشاندهنده تغییرات یک فایل یا ایحاد شدن یک فایل است. بعد با دستور git add باید تغییرات رو به مرحله دوم یعنی staged ارسال کنید. این مرحله نشان دهنده رضایت شما از تغییرات ایجاد شده است. و در آخر برای ثبت کردن تغییرات باید با دستور git commit پروژتون رو به مرحله commited بفرستید. زمانی که به این مرحله رسیدید، پروژه شما در حالت ثبات قرار میگیره.

Untracked files به معنی این است که این فایل تازه به وجود آماده است و تاکنون git آن را دنبال نمیکرده است

Modified به فایل هایی گفته میشونده که git آنهارا دنبال میکرده است (از قبل وجود داشته اند) اما تغییر کرده اند و لازم است آنها را ثبت کنیم



Git add name\_file

Gti commit -m ‘massage’

=====================================================

زمانی که تعداد فایل ها در مرحله اول زیاد بود به جای نوشتن اسم تمام فایل ها میتونید با استفاده از A- به گیت بفهمونید که تمام فایل ها رو به مرحله staged منتقل کنه همچنین میتونید برای راحتی از کاراکتر . نقطه هم استفاده کنید. در این قسمت میبینید که چطور میشه با استفاده از دستور reset فایل ها که داخل مرحله staged هستنید رو به مرحله اول برگردونید. همچنین با روش wildcard برای ارسال فایل ها از مرحله اول به مرحله دوم آشنا میشید

Git add name\_file1 name\_fiel2

Git add -A ======> git add .

انتقال فایل ها از مرحله اول به مرحله دوم

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git reset name\_file

با استفاده از این دستور فایل انتخابی از مرحله staged به مرحله اول برمیگرد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Wildcard (عبارات باقاعده) مثلا :

Git add \*.txt

این دستور به این معنی است که فقط تمام فایل هایی که پسوند txt دارند را به حالت staged منتقل کن

=====================================================

بعد از اینکه چندین commit در گیت ایجاد کردید به احتمال زیاد دلتون میخواد که بتونید یک تاریخچه از تغییراتی که داشتید رو ببینید. برای اینکار میتونید از دستور log در گیت استفاده کنید. به صورت پیشفرض اگر هیچ آپشنی به دستور ندید، تمام commitها از جدید به قدیمی نشان داده میشود. این دستور یک هش چهل کاراکتری از نوع SHA-1 به همراه، name, email و تاریخ و پیغام هر commit را نشان میدهد. هر commit دارای یک هش است که این هش بین تمام commitها یکتا بوده و با استفاده از آن میتوان در تاریخچه تغییرات حرکت کرد. همچین دستور log در git آپشنهای زیادی داره که با استفاده از اونها میتونید شکل نمایش log رو تغییر بدین. در این قسمت با دستور log در گیت آشنا میشید

Git log

Git log –oneline

Git log –stat

هر سه ی این دستورات تمام commit های مارا نمایش میدهد اما هرکدام با جزئیات متفاوت

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git log –patch ===> git log -p

این هم ماننده 3 تای بالایی میباشد با این تفاوت که تغییرات هر commit نسبت به commit قبل را به صورت کامل نمایش میدهد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEAD همیشه به آخرین commit اشاره دارد

=====================================================

branchها در واقع یک کپی از برنامه شما هستند. زمانی که بخواید به پروژتون یک ویژگی جدیدی اضافه کنید به جای کار کردن روی سورس پروژه اصلی با استفاده از گیت میتونید یک branch از برنامه خودتون ایجاد کنید. با این کار گیت یک کپی کامل از سورس پروژه شما رو کپی میکنه و شما میتونید روی اون branch کار کنید و اگر از تغییرات راضی بودید میتونید اون branch رو با سورس پروژه اصلی که به صورت پیشفرض روی branch master هستش ترکیب کنید. اما اگر از تغییرات راضی نبودید میتونید خیلی راحت اون branch رو حذف کنید بدون اینکه تغییری روی سورس اصلی برنامتون اعمال بشه

Git branch ===> git branch –list

نمایش لیست branch ها

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git branch name\_branch

برنچی با نام دلخواه شما (name\_branch) ساخته میشود

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git checkout name\_branch

وارد برنچ مورد نظر شما میشود (تغییر branch)

=====================================================

برای تغییر نام branch میتونید از آپشن m استفاده کنید. برای ترکیب کردن دو branch مختلف میتونید از دستور merge استفاده کنید. برای حذف کردن branch دو آپشن مختلف وجود دارد. اگر branch قبلا merge شده بودید میتونید از آپشن d استفاده کنید در غیر اینصورت میتونید از آپشن D استفاده کنید.

Git branch -m name\_origin name\_new

تغییر نام برنچ از name\_origin به name\_new

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git checkout master

Git merge name\_branch

برای اینکه تغییراتی که در برنچی اعمال کردیم را بر روی برنچ مستر بیاوریم باید ابتدا وارد master بشوید بعد با زدن دستور دوم اینکار انجام میشود

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git branch -d name\_branch

برنچی که مرج شده است را با استفاده از -d پاک میکند

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git branch -D name\_branch

برنچی که هنوز مرج نشده است با این دستورپاک میشود

=====================================================

دستور checkout کاربرد متفاوتی دارد. قبلا دیدید که با استفاده از checkout میتونستید branch های مختلفی رو فعال کنید. در این جلسه با کاربرد دیگری از checkout آشنا میشید. با استفاده از این دستور میتونید بین commit ها حرکت کنید و پروژتون رو به commit خاصی برگردونید. دقت کنید که دستور checkout تغییرات رو حذف نمیکنه و بلکه فقط پروژه ما رو به زمان خاصی برمیگردونه تا ما بتونیم وضعیت پروژمون رو در commit خاصی ببینیم. در این حالت تغییراتی که ایجاد میکنید، تاثیری در سورس اصلی نخواهد داشت و این حالت read-only میباشد