**معرفی گیت:**

Git یک سیستم کنترل ورژن (Version Control) می باشد که برای نرم افزارها و سایت های در مقیاس کوچک تا بسیار بزرگ قابل استفاده است. وب سایت مرجع گیت، [git-scm.com](https://git-scm.com/) می باشد.

یکی از کاربردهای گیت، کنترل نسخه های مختلف کدها در پروژه های نسبتا بزرگی مانند [فروشگاه آنلاین](http://mohtavaban.com/category/online-shop-mvc/) می باشد.

**دانلود گیت:**

ابتدا با مراجعه به آدرس <https://git-scm.com/download> گیت را با توجه به سیستم عامل خود (Windows, Linux/Unix, Mac)، دانلود کرده و آنرا نصب کنید.

**راه‌اندازی گیت:**

سپس فولدر حاوی فایل های پروژه خود را باز کرده و راست کلیک می کنیم. اکنون گزینه Git bash here را انتخاب نمائید.

پنجره ای باز می شود همانند Cmd که می توان دستورات گیت را در آن وارد کرد. به این محیط، خط فرمان گیت (Git Command Line) می گویند.

ابتدا git init را می نویسیم و enter می کنیم. با انجام اینکار راه اندازی اولیه گیت را انجام داده ایم و فولدری بنام .git (بصورت مخفی:hidden) در فولدر اصلی پروژه مان ساخته می شود.

**مفاهیم Repository ، Stage و Commit:**

قبل از معرفی دستورات گیت و اجرای آنها، نیاز است مفاهیم stage ، repository و commit را متوجه شوید.

**نکته**: اگر نرم افزار ویژوال استودیو کد (Visual Studio Code) را روی سیستم خود نصب کرده باشید، می توانید در cmd (روی فولدر جاری پروژه) دستور . code را اجرا کنید تا تمام فایل ها و فولدرهای پروژه شما بطور اتوماتیک با VS code باز شوند.

code .

در زمینه ارتباط بین پروژه-stage-repository به تصویر زیر توجه کنید:

در تصویر فوق، سه مرحله مشاهده می کنید که مرحله اول (قرمزرنگ) فولدر حاوی پروژه می باشد. مرحله دوم بخش stage می باشد که شامل یک فایل می باشد که در فولدر git قرار دارد. مواردی که باید در دستورات بعدی در Repository اضافه شوند (commit شوند) در این فایل نگهداری می شوند. پس می توان گفت stage یک مرحله واسط بین پروژه ما و Repository می باشد.

**مشاهده وضعیت پروژه در گیت:**

برای اینکه وضعیت فعلی repository را متوجه شوید، دستور git status را اجرا کنید.

با اجرای دستور git status فایل هایی که در stage وجود ندارند با رنگ قرمز مشخص می شوند و فایل های موجود در stage نیز سبزرنگ خواهند بود.

برای افزودن یک فایل خاص (مثلا index.html) به stage دستور git add index.html را اجرا کنید.

git add index.html

تصویر زیر را در این زمینه مشاهده کنید:

ممکن است تعداد فایل ها و فولدرهایی که در stage وجود ندارند بسیار زیاد باشد. بنابراین نمی توان برای هریک از این فایل ها یکی یکی دستور add را اجرا کرد!

برای افزودن تمامی فایل ها و فولدرهایی که در stage نیستند دستور git add . را وارد کنید.

git add .

اکنون اگر git status را اجرا کنیم، تمامی فایل های پروژه ما با رنگ سبز نمایش داده می شوند.

نکته: برای پاک کردن دستوراتی که در گیت نوشته ایم یا بقول معروف تمیزکاری اسکرین، کافیست دستور clear را اجرا کنیم.

**دستور commit در گیت:**

می توان با اجرای دستور git commit تمام تغییرات را در repository ذخیره کرد، اما پس از اجرای دستور commit پیام خطایی مشاهده می کنیم که اعلام می کند باید بهمراه دستور commit یک پیغام (message) نیز ذخیره کنیم.

پس دستور کامیت را بصورت git commit –m “your message here” را اجرا می کنیم.

git commit –m "your message here"

تصویر زیر را مشاهده کنید:

حال اگر دستور git status را اجرا کنیم، پیام “nothing to commit, working tree clean” را مشاهده خواهیم کرد.

در این مرحله یکی از فایل های پروژه خود را ویرایش کنید (مثلا در فایل index.html یک hello وارد کنید)

اکنون اگر دستور git status را اجرا کنیم، اعلام می کند که یکی از فایل هایی که هم اکنون در repository وجود دارد، تغییر کرده است و پیغام modified index.html را با رنگ قرمز نمایش می دهد.

برای ثبت تغییرات انجام شده در فایل های پروژه خود، می توان مستقیما از دستور git commit استفاده کرد. به این صورت:

git commit –a –m "add hello to index file"

**نکته 1**: پس از هر بار commit کردن، نیاز است که توسط دستور –m یک پیغام نیز همراه آن ذخیره کنید.

**نکته 2**: دستور –a باعث می شود تا کار add کردن تغییرات به stage نیز همزمان با commit انجام شود.

**دستورات بخش اول آموزش گیت:**

دستوراتی که تا این بخش از آموزش گیت (Git) یاد گرفته اید عبارتند از:

* **Git init**: راه اندازی اولیه گیت و ایجاد فولدر .git در پروژه
* **Git status**: مشاهده وضعیت ذخیره فایل های پروژه
* **Git add**: افزودن یک یا چند فایل و فولدر به Stage
* **Git commit**: ذخیره فایل ها و فولدرهای پروژه در Repository

دستورات کاربردی گیت Git

در این بخش از آموزش گیت، نحوه حذف یک یا چند فایل از stage، مشاهده لاگ commit های انجام شده، ریست کردن کامیت ها و غیره… را آموزش می دهیم.

نحوه حذف یک فایل از stage:

برای حذف یک فایل از Stage یا اصطلاحا Unstage کردن، دستور زیر را اجرا می کنیم:

git rm –cashed **<**file**>**

 بعنوان مثال می خواهیم فایل main.js را از فولدر js واقع در مرحله stage حذف کنیم. دستور مربوطه بصورت –git rm –cached js/main.js می شود.

حذف کلیه فایل ها و فولدرها از stage:

برای حذف کلیه فایل های پروژه از stage دستور زیر را خواهیم داشت:

Git rm –cached –r .

 با اجرای این دستور، تمامی فایل و فولدرهای پروژه از stage خارج می شوند.

**نکته**: اگر فایل جدیدی به پروژه خود اضافه کنید، نمی توانید با دستور **“git commit –a –m “message** آنرا در stage و repository اضافه کنید. بلکه ابتدا باید آنرا توسط دستور add به stage اضافه و سپس در repository آنرا commit کرد.

دستور git diff در گیت:

برای مشاهده تفاوت بین آخرین commit انجام شده با وضعیت حال حاضر فایل های پروژه، ابتدا دستور **git status** را اجرا کرده تا وضعیت فعلی را ببینید و سپس دستور **git diff** را اجرا کنید. با انجام اینکار، فایل هایی که دچار تغییر شده اند را بهمراه تفاوت های آنها با وضعیت فعلی آن فایل ها را مشاهده خواهید کرد.

با اجرای دستور git diff –help دستورالعمل جامع استفاده از دستور diff برای شما نمایش داده خواهد شد.

**مشخص شدن تغییرات در نرم افزارهای مختلف:**

در صورتی که از نرم افزار Atom بعنوان IDE استفاده می کنیم، فایل ها و فولدرهایی که دچار تغییر می شوند، به رنگ زرد در می آیند و همچنین در سایر نرم افزارها مانند ویژوال استودیو با علائم دیگری این تمایز نشان داده می شود.

فایل ها و فولدرهایی که جدیدا به پروژه خود اضافه کرده باشیم، در نرم افزار Atom به رنگ سبز نمایان خواهند شد.

به همین ترتیب اگر در یک فایل، خطی را ویرایش کنیم، در کنار آن خط، یک علامت زردرنگ خواهیم دید و اگر خطی جدید به آن فایل اضافه کنیم، رنگ سبزرنگ را در کنار آن خط می بینیم.

**دستور git log در گیت:**

برای مشاهده تاریخچه commit های انجام شده، کافیست دستور git log را اجرا کنیم. با انجام اینکار، تمام commit های انجام شده به ترتیب آخرین کامیت نمایش داده می شوند.

آیتم هایی که در هر لاگ نمایش داده می شوند، عبارتند از:

* آیدی کامیت (Commit ID) که عددی منحصر بفرد و یکتا می باشد.
* نام نویسنده (Author) که آدرس ایمیل کاربر را بهمراه نام او نمایش می دهد.
* تاریخ کامیت (Date) که تاریخ و زمان دقیق اجرای دستور commit را اعلام می کند.
* پیغام (message) که تعریف آن برای هر commit الزامی می باشد.

برای مشاهده راهنمای دستور log کافیست دستور git log –help را اجرا کنیم. با اجرای این دستور، مقاله کاملی راجع به دستور log و انواع مختلف استفاده از آنرا مشاهده خواهیم کرد.

git log –help

**مشاهده لاگ commit ها بصورت خطی:**

روش دیگری که می توان دستور لاگ را اجرا کرد بصورت تک خطی و خلاصه می باشد. بدین منظور کافیست دستور git log –oneline را اجرا کنیم.

git log –oneline

با اجرای دستور فوق، تاریخچه commit های انجام شده را بصورت خلاصه مشاهده خواهیم کرد. بطوریکه ابتدا چند رقم از آیدی کامیت و سپس پیغام هر commit را نمایش می دهد.

اگر دستور git log -2 را اجرا کنیم، دو لاگ آخر را نمایش می دهد.

برای مشاهده جزئیات کامل هر commit (تغییرات انجام شده در هر کامیت) دستور git log –p را اجرا کنید.

دستوراتی که در بخش دوم از آموزش گیت معرفی کردیم، عبارتند از:

* دستور **git log**
* دستور **git log –oneline**
* دستور **git log -p**
* دستور **git diff**
* دستور **git rm –cached –r**

**نحوه بازگردانی تغییرات commit شده (ریست کردن):**

در صورتی که بخواهیم commit های انجام شده را undo کنیم، باید از دستور checkout استفاده کنیم.

در ادامه، 3 سناریوی رایج را بیان می کنیم که ممکن است برای شما در پروژه تان رخ دهد و نیاز باشد تا به کامیت های قبلی خود بازگردید:

**ریست کردن commit – سناریوی اول:**

فرض کنید تغییراتی در فایل index.html و فایل main.css اعمال کرده اید، حالا متوجه شده اید که این تغییرات درست نبوده اند و می خواهید این فایل ها را به حالت قبل از تغییر برگردانید. در این مثال فرض شده است که پس از انجام تغییرات، دستورات افزودن به stage یعنی add و نیز افزودن به repository یعنی commit را انجام نداده اید. اگر دستور git checkout – index.html را اجرا کنیم، تنها فایل index.html به حالت قبل از تغییرات برخواهد گشت.

 اگر بخواهیم تغییرات تمام فایل ها را برگردانیم، باید دستور git checkout — . را اجرا کنیم. علامت نقطه یعنی “.” باعث می شود تا دستور checkout روی تمام فایل های پروژه اجرا شود.

git checkout -- .

**ریست کردن commit – سناریوی دوم:**

فرض کنید تغییراتی که در برخی از فایل های پروژه خود داده اید و به stage اضافه کرده اید (توسط دستور git add .) را به حالت قبل برگردانیم و آنها را commit نکنیم، دستور git reset HEAD index.html و یا بطور کلی git reset HEAD . را باید اجرا کنیم. با اینکار عملیات Unstage را انجام داده ایم.

git reset HEAD index.html

تصویر زیر:

 در این مرحله تغییرات انجام شده از stage خارج شده و تنها در سطح پروژه موجود است. اکنون اگر دستور git status را اجرا کنیم، گیت، فایل هایی که تغییر داده اید را با رنگ قرمز نمایش می دهد و اعلام می کند برای ذخیره این تغییرات باید ابتدا دستور add و سپس دستور commit را اجرا کنید. اکنون برای برگرداندن تغییرات فایل های موردنظر، مانند مثال اول، دستورgit checkout — .  را باید اجرا کنیم.

**ریست کردن commit – سناریوی سوم:**

فرض کنید تغییرات موردنظرمان را در repository اضافه (commit) کرده ایم. اگر بخواهیم تغییرات پروژه را به یک commit خاص برگردانیم، ابتدا با دستور git log –oneline لیست تمام کامیت ها را بصورت خلاصه و تک‌خطی مشاهده می کنیم.

**نکته**: سعی کنید آیدی کامیت ها را در جایی ذخیره کنید تا بتوانید بعدا در صورت بروز اشتباه، به کامیت دلخواه خود بازگردید. اکنون که لیست commit ها را بهمراه آیدی آنها در اختیار داریم، می توانیم به هر کامیتی که می خواهیم بازگردیم. بدین منظور، دستور git reset Commit-ID را اجرا کنید. باید بجای Commit-ID آیدی کامیت موردنظر خود را بنویسید. حال فرض کنید در یک Commit یک یا چند فایل یا فولدر ایجاد کرده ایم و بخواهیم این commit را ریست کنیم (یعنی به commit قبل از آن بازگردیم) باید از دستور git reset –hard Commit-ID استفاده شود.

git reset –hard Commit-ID

تصویر زیر:

 با اجرای این دستور، فایل ها یا فولدرهای مذکور نیز پاک خواهند شد. به همین ترتیب اگر بخواهیم مجددا به همین کامیت برگردیم، پس از اجرای همان دستور، فایلها و فولدرها دوباره ساخته خواهند شد.

با مراجعه به [وب سایت مرجع گیت](https://git-scm.com/docs/git-reset)، می توانید درباره دستور reset در Git اطلاعات بیشتری کسب کنید.

**دستورات branch و merge در گیت**

در این بخش می خواهیم درباره مفهوم بسیار کاربردی و مهم گیت یعنی شاخه یا branch، نحوه جابجایی بین شاخه ها و ادغام یا merge کردن آنها صحبت کنیم.

**کاربرد branch در گیت چیست؟**

فرض کنید در حال توسعه یک اپلیکیشن تحت وب می باشید. می خواهید نسخه های مختلفی از این اپلیکیشن را ارائه کنید. احتمالا برای انجام اینکار تا قبل از آشنایی با Git، برای نگهداری نسخه های مختلف پروژه، از فولدر آن پروژه یک کپی تهیه می کردید و کلمه v1 به معنی version1 را به انتهای نام آن فولدر اضافه می کردید. اما اکنون که در حال یادگیری سیستم ورژن کنترل Git می باشید، توصیه می شود از امکاناتی که گیت برای داشتن ورژن های مختلف پروژه در اختیارتان قرار می دهد استفاده کنید. پس از استفاده از قابلیت تعریف branch در گیت، قادر خواهید بود نسخه اول پروژه خود را ذخیره کنید و به توسعه نسخه دوم آن ادامه دهید.

**آشنایی با مفهوم merge در گیت:**

فرض کنید یک گروه در حال توسعه یک وب سایت می باشد و این تیم شامل 4 نفر می باشد. این افراد بر روی 4 قسمت مجزای وب سایت یعنی هدر، فوتر، بخش اصلی و سایدبار کار می کنند. هنگامیکه کار این 4 نفر به اتمام برسد، باید این 4 بخش را با هم ادغام (merge) کنند و یک وب سایت را تشکیل دهند.

**راهنمای استفاده از دستور branch در Git:**

اکنون باید نحوه تعریف دستور branch را در گیت فرا بگیرید. همانند سایر دستورات گیت، با اجرای دستور git branch –help می توانید وارد مقاله آموزشی دستور branch از وب سایت git شوید و یا توسط دستور git branch –h می توان تمام دستورات مرتبط با branch را در همان کنسول گیت مشاهده کنید.

git branch –h

به منظور تعریف یک branch کافیست دستور ساده git branch dev را اجرا کنیم که در اینجا dev یک کلمه دلخواه است. یعنی ما نام اولین branch پروژه خود را dev قرار دادیم.

**مشاهده لیست شاخه ها در گیت:**

برای مشاهده لیست تمام branch ها، دستور git branch –a را اجرا می کنیم. در هر پروژه ای که از گیت بهره می برد، branch ای بنام master بصورت پیش فرض وجود دارد که شاخه اصلی هر پروژه می باشد. بنابراین در این مثال پس از اجرای دستور git branch –a لیست شاخه ها بصورت master و dev نمایش داده خواهند شد.

**مشاهده لیست branch های پروژه لاراول:**

برای مشاهده branch های یک پروژه واقعی در وب سایت github.com به آدرس <https://github.com/laravel/laravel> مراجعه نمائید. در بخش branch: master با باز کردن لیست بازشو، لیست تمام نسخه های فریم ورک laravel را می توانید مشاهده کنید. با انتخاب هریک از این ورژن ها، ممکن است فایل ها و فولدرهای پروژه لاراول کم و یا زیاد شود.

**جابجایی بین branch ها در گیت:**

به منظور جابجایی بین branch های مختلف در یک پروژه، دستور git checkout dev را اجرا کنید. در این دستور، می توانید بجای dev نام شاخه دلخواه خود را بنویسید تا به آن branch سوئیچ کنید.

git checkout dev

تصویر زیر:

**نحوه حذف یک شاخه در گیت:**

برای پاک کردن شاخه dev باید ابتدا از آن branch خارج شوید، سپس آنرا توسط دستور git branch –d dev حذف کنید.

git branch –d dev

تصویر زیر:

**نکته**: برای اینکه دو عمل ایجاد شاخه و سوئیچ به آن شاخه را بصورت همزمان انجام دهید باید دستور git checkout –b develop را اجرا کنید. نام شاخه ما در این مثال develop می باشد. با اجرای این دستور باعث می شوید هم شاخه develop را ایجاد کرده باشید و هم به شاخه develop سوئیچ کرده باشید.

**کاربرد branch (شاخه) در گیت (Git):**

دقت داشته باشید که در صورت اعمال تغییرات در شاخه dev (مثلا ویرایش یک فایل و افزودن دو فایل جدید) این تغییرات تنها در همین شاخه قابل مشاهده است. بنابراین اگر به شاخه master سوئیچ کنید، پروژه به حالت قبل بازمی گردد. یکی از کاربردهای تعریف branch اینست که شما بخواهید یک نسخه تستی از پروژه تان را در کنار نسخه اصلی داشته باشید و کدهای جدیدی که می خواهید روی پروژه خود تست کنید را در ورژن تستی پروژه اعمال کنید. بدون اینکه نسخه اصلی پروژه شما ضربه ای بخورد. پس از اینکه کدهای جدیدتان را در شاخه تستی امتحان کردید و جواب داد، می توانید این کدها را به شاخه اصلی پروژه خود اضافه کنید. اما آیا باید این کدهای جدید را بصورت دستی از نسخه تستی به نسخه اصلی منتقل کرد (کدهای جدید را از نسخه تستی کپی و در نسخه اصلی paste کرد)؟ پاسخ این سوال خیر است. شما می توانید براحتی با استفاده از دستور merge این دو نسخه را ادغام کنید.

**نکته**: توجه داشته باشید که حتما قبل از استفاده از دستور merge و ادغام نسخه ها با یکدیگر، توسط دستور commit، تغییرات انجام شده در شاخه تستی را به repository اضافه کنید. بنابراین با اجرای دستور git commit –a –m “add changes to test branch” می توان اینکار را انجام داد.

**انجام عملیات ادغام شاخه ها (merge):**

فرض کنید دو شاخه داریم به نام های master و develop و کدهای جدید خود را در شاخه develop نوشته ایم و تست موفقیت آمیزی داشته اند. اکنون می خواهیم این کدها را نیز در شاخه master داشته باشیم. بدین منظور، ابتدا به شاخه master سوئیچ کرده و سپس دستور git merge develop را اجرا کنید. با اجرای این دستور، کدهای جدید و احتمالا فایل هایی که در شاخه develop ایجاد کرده ایم به master منتقل خواهند شد.

برای اینکه مطمئن شویم این انتقال بدرستی صورت گرفته است می توانید کد git log را در شاخه master اجرا کنید و لاگ های آن را مشاهده کنید. برای مشاهده لاگ های انجام شده بصورت گراف، کافیست دستور git log –graph را اجرا کنید.

git log –graph

**دستور Stash و کاربرد آن در گیت**

در این بخش، دستور کار راه انداز stash را در گیت معرفی می کنیم.

**کاربرد**[**دستور stash**](https://git-scm.com/docs/git-stash)**در گیت:**

در هنگام سوئیچ کردن از شاخه ای به شاخه دیگر در Git، ممکن است دچار یک مشکل شوید و این مشکل به شرح زیر است:

فرض کنید مانند بخش قبل، دو branch بنام های develop و master دارید. الان شما در شاخه develop هستید و تغییراتی در کدهای پروژه خود اعمال کرده اید. شما فعلا نمی خواهید این تغییرات را commit کنید و می خواهید به شاخه master سوئیچ کنید. در صورت انجام اینکار، تمام تغییراتی که در شاخه develop انجام داده اید و commit نکرده اید، به شاخه master منتقل می شوند و این قضیه برخلاف خواسته ما می باشد و منطقا نباید این انتقال تغییرات صورت گیرد.

پس صورت مسئله اینست که شما نه می خواهید تغییرات انجام شده در شاخه develop را commit کنید و نه می خواهید این تغییرات به شاخه master منتقل شوند. راه حل چیست؟

**راهنمای استفاده از دستور stash در Git:**

خود سیستم گیت برای رفع این مشکل استفاده از دستور stash را پیشنهاد می کند. بنابراین دستور git stash –help را در گیت اجرا کنید تا راهنمای کامل استفاده از دستور stash را در وب سایت گیت مشاهده کنید.

**روش اجرای دستور stash:**

 روش کار بدین صورت است که پس از انجام تغییرات در شاخه develop دستور git stash را اجرا کنید. با اجرای این دستور، تمام تغییرات شما در stash ذخیره می شوند. خود گیت برای هر stash یک نام و آیدی منحصر بفرد در نظر می گیرد. اکنون اگر git status بگیریم، پیغام “nothing to commit, working tree clean” را مشاهده خواهید کرد.

**مشاهده لیست Stash ها و حذف آنها در گیت:**

 حال برای مشاهده لیست stash ها دستور git stash list را اجرا کنید. برای حذف یک stash نیز دستور git stash drop stash@{0} را اجرا کنید.

git stash drop stash@**{**0**}**

تصویر زیر:

 در اینجا، stash@{0} نام استش موردنظر شما می باشد که توسط دستور git stash list بدست آورده اید.

همچنین شما می توانید در هنگام ذخیره تغییرات در یک stash یک پیغام دلخواه نیز برای آن تعریف کنید. پس نیاز است دستور git stash save “your message here” را اجرا کنید.

**مشاهده محتوای یک Stash در گیت:**

برای مشاهده اینکه در یک stash چه چیزی وجود دارد، ابتدا توسط دستور git stash list آیدی stash موردنظرمان را استخراج می کنیم و سپس با اجرای دستور git stash show stash@{1} تغییرات داخل آن stash را مشاهده نمائید. در اینجا محتوای stash دوم را به نمایش درآورده ایم. با دستور git stash show –p stash@{1} می توان تغییرات داخل stash دوم را بصورت مشخص تری مشاهده کنیم.

git stash show –p stash@**{**1**}**

تصویر زیر:

**اعمال تغییرات واقع در Stash ها توسط pop یا apply:**

برای اینکه تغییرات داخل یک stash را به شاخه جاری (current branch) اعمال کنید، باید از یکی از دستورات pop و یا apply استفاده کنید.

**تفاوت دستور pop با apply:**

 تفاوت pop با apply اینست که در pop ، تغییرات واقع در stash موردنظر در شاخه فعال commit می شود و از لیست stash ها خارج می شود اما در apply تغییرات واقع در stash موردنظر در شاخه فعال commit می شود اما آن stash از لیست stash ها خارج نمی شود و باید خودمان توسط دستور drop آنرا پاک کنیم.

**اجرای دستورات pop یا apply:**

برای apply کردن یک stash دستور git stash apply stash@{0} را اجرا کنید. پس از apply کردن یک stash خاص، باید آنرا در شاخه موردنظر commit کنید تا تغییرات در repository اعمال شوند.

git stash apply stash@**{**0**}**

تصویر زیر:

**دستور gitignore در گیت:**

در این بخش، به معرفی دستور gitignore و کاربرد آن در گیت و نحوه استفاده از آن خواهیم پرداخت.

**کاربرد دستور gitignore:**

گاهی اوقات پیش می آید که می خواهید گیت از بررسی برخی از فایل ها و یا فولدرهای پروژه‌تان صرف نظر کند و تغییرات آنها را نادیده بگیرد. در واقع آنها را دنبال نکند (Untrack). مانند هنگام استفاده از nmp (node package manager) که فولدرها و فایل های زیادی را در پروژه تان ایجاد می کند و نیازی نیست توسط سیستم کنترل ورژن Git بررسی شوند.

**تعریف دستور gitignore در گیت:**

فرض کنید در پروژه ما فولدری بنام node\_modules وجود دارد که نمی خواهیم توسط Git دنبال شود و همچنین می خواهیم تمام فایل هایی که دارای فرمت txt می باشند توسط گیت نادیده گرفته شوند. ابتدا دستور git touch .gitignore را اجرا کنید تا فایل .gitignore در فولدر اصلی پروژه تان ساخته شود.

git touch .gitignore

سپس فایل .gitignore را باز کنید و عبارات زیر را داخل آن بنویسید:

node\_modules/

\*.txt

تصویر زیر:

**نکته1**: node\_modules بدلیل اینکه فولدر می باشد و نمی خواهیم فایل ها و فولدرهای داخل آن هم توسط گیت دنبال شوند، علامت / را بعد از آن تعریف کرده ایم. در خط دوم نیز اعلام کرده ایم که هر فایل با پسوند txt را دنبال نکند. مهم نیست نام آن فایل چه باشد.

اکنون اگر git status را اجرا کنیم، ملاحظه می شود که گیت فقط فایل .gitignore را بعنوان فایل جدید شناسایی می کند و می گوید باید آنرا به مخزن (repository) commit کنید. و فایل های متنی با فرمت txt و فولدر node\_modules را نادیده می گیرد. پس در این مرحله فقط کافیست فایل .gitignore را کامیت کنید. بدین منظور، دستور git add . و سپس git commit –m “add gitignore file” را اجرا کنید.

**صرف نظر کردن از فایل های متنی در گیت:**

نکته2: ممکن است بخواهیم گیت تمام فایل های متنی با فرمت txt را دنبال نکند اما در مورد یک فایل خاص متنی استثناء قائل شود و تغییرات آنرا بررسی کند. فرض کنید نام فایل موردنظرمان trackit.txt می باشد. کافیست عبارت !trackit.txt را در فایل .gitignore تعریف کنید. اکنون اگر git status بگیریم، می گوید تغییرات فایل trackit.txt در مخزن گیت درج نشده است و باید commit شود.

**دنبال نکردن (untrack) فایل یا فولدر در گیت:**

نکته3: در مواقعی ممکن است بخواهید فایل یا فولدری که قبلا توسط گیت دنبال میشد را دیگر دنبال نکنید (untrack). در این مثال می خواهیم فولدر js را بهمراه تمام محتویاتش untrack کنیم. بدین منظور ابتدا  در فایل .gitignore عبارت js/ را بنویسید. سپس دستور git rm –cached –r . را اجرا کنید. در این مرحله نباید هیچ تغییری در فایل یا فولدری انجام دهید. بلافاصله دستور git add . را اجرا کنید. اکنون فولدر js بهمراه فایل ها و فولدرهای داخل آن untrack شده است و اگر تغییری در آنها اعمال کنید، گیت دنبال نخواهد کرد. در انتها برای ثبت تغییرات کافیست commit کنید.

بنابراین بطور خلاصه کافیست دستورات زیر را به ترتیب اجرا کنید:

git rm –cached –r .

git add .

git commit –m “Untrack js folder”

برای دریافت اطلاعات بیشتر درباره دستور gitignore به [مقاله مرجع دستور gitignore در سایت git](https://git-scm.com/docs/gitignore) مراجعه کنید.

تعریف نام و ایمیل در گیت

در این بخش، نحوه تعریف نام کاربر و آدرس ایمیل او را در سیستم گیت آموزش می دهیم.

**تعریف نام و آدرس ایمیل در گیت توسط دستور git config:**

در صورتی که فایل config موجود در فولدر .git را ویرایش کنید و نام و ایمیل خود را در آنجا وارد کنید، اطلاعات شما فقط در سطح همان پروژه ذخیره می شود. اما اگر می خواهید نام و ایمیل شما بصورت سراسری (Global) تعریف شوند، باید به مسیر زیر بروید:

C:\Users\your\_user\.gitconfig

در مسیر فوق، بجای your\_user نام یوزر ویندوز سیستم خود را بنویسید. فایل .gitconfig را توسط nodepad باز کنید. برای مشاهده راهنمای استفاده از دستور config ، دستور git config –help را اجرا کنید.

**تعریف نام توسعه دهنده در تنظیمات گیت:**

با اجرای دستور git config user.name “Ehsan Safari” نام شما بصورت محلی (local) و تنها در همان پروژه تعریف خواهد شد و در پروژه های بعدی که با گیت کنترل می کنید، مجددا باید نام خود را تعریف کنید. پس از اجرای دستور فوق، به فولدر .git بروید و فایل config را توسط notepad باز کنید. مشاهده می کنید که نام شما به این فایل افزوده شده است. برای اینکه نام خود را بعنوان توسعه دهنده پروژه، بصورت سراسری (Global) تعریف کنید، باید ابتدا خطی که با اجرای کد بالا در فایل config درست شده است را بصورت دستی پاک کرده و دستور git config –global user.name “Ehsan Safari” را اجرا کنید.

git config –global user.name "Ehsan Safari"

**تعریف ایمیل توسعه دهنده در تنظیمات گیت:**

برای درج ایمیل توسعه دهنده در گیت بصورت Global، دستور git config –global user.email “EhsanSafari@hotmail.com را اجرا کنید.

git config –global user.email "EhsanSafari@hotmail.com"

در ادامه مقاله، وارد وب سایت github خواهیم شد و پس از ایجاد یک حساب کاربری و لاگین به آن، پروژه خود را آپلود خواهیم کرد.

**نحوه آپلود پروژه در گیت‍هاب (Github)**

در این بخش، می خواهیم آموزش دهیم که چگونه در سایت گیت هاب ثبت نام کنید و پروژه خود را بر روی آن آپلود کنید.

**ثبت نام در وب سایت گیت هاب:**

اکنون نوبت آنست که یاد بگیرید چگونه پروژه خود را بر روی وب سایت Github قرار دهید تا سایر کاربران و توسعه‌دهندگان بتوانند پروژه شما را مشاهده، ویرایش و توسعه دهند. به این منظور وارد وب سایت Github.com شوید و روی لینک sign up کلیک کنید. در فرم ثبت نام سایت گیت‌هاب، نیاز است فیلدهای username و email و password را وارد کنید. پس از فشردن دکمه signup ایمیلی برای تائید مشخصات ورودی شما به آدرس ایمیل شما ارسال می شود. پس لازم است inbox ایمیل خود را چک کنید و روی لینک فعالسازی اکانت خود کلیک کنید. پس از تائید آدرس ایمیل خود، به پروفایل خود در سایت گیت هاب مراجعه کرده و آنرا تکمیل کنید.

**ایجاد یک مخزن جدید (New Repository):**

در این مرحله می توانید اولین پروژه خود را در گیت ایجاد کنید. روی دکمه پلاس (+) کلیک کرده و گزینه New Repository را انتخاب کنید.

در صفحه جدیدی که باز می شود باید نام مخزن (repository) را وارد کنید (بصورت username/repository\_name) در بخش زیرین نام مخزن، می توان توضیحات کوتاهی برای مخزن پروژه خود تعریف کنید.

**تنظیم دسترسی و یا عدم دسترسی سایر کاربران به مخزن پروژه:**

 سپس یکی از گزینه های private یا public را برای تنظیم دسترسی و یا عدم دسترسی سایر کاربران به مخزن پروژه خود را انتخاب کنید. دقت کنید که گزینه private پولی می باشد و نیاز به یک master card می باشد. ما در این مثال، گزینه public را انتخاب می کنیم که رایگان می باشد. در نهایت بر روی دکمه Create Repository کلیک کنید تا مخزن شما ایجاد شود.

فیلد Description که در هنگام ایجاد یک مخزن تعریف کرده اید، بعنوان فایلی بنام description در فولدر .git پروژه شما قابل دسترسی است.

**لود کردن پروژه در سایت Github:**

برای قرار دادن پروژه خود در مخزنی که در سایت گیت هاب تعریف کرده اید، بصورت زیر عمل کنید:

وارد مخزنی که در بالا تعریف کرده اید شوید و مطابق تصویر زیر، در بخش push an existing repository from the command line دستور مربوط به ریموت زدن به پروژه در گیت را در کامند لاین گیت کپی کرده و اجرا کنید. در این مثال، دستور موردنظر بصورت git remote add origin <https://github.com/Safari66/project-test.git> می باشد.

**ریموت زدن به پروژه در گیت هاب:**

اکنون روی فولدر اصلی (root directory) پروژه خود کلیک کنید و سپس گزینه git bash here را انتخاب کنید. دستور فوق را در commandline  گیت نوشته و اجرا کنید. سپس باید پروژه را در وب سایت گیت لود کنیم. فرض کنید ما دراین مثال می خواهیم شاخه master را در گیت هاب لود کنیم. برای انجام اینکار، دستور git push –u origin master را اجرا کنید. این دستور می گوید شاخه master پروژه از طریق ریموت origin که قبلا با وب سایت Github برقرار کرده ایم، در گیت هاب لود شود.

git push –u origin master

تصویر زیر:

پس از اجرای دستور PUSH گیت، یک فرم بصورت پاپ آپ (POPUP) برای شما نمایش داده خواهد شد که از شما نام کاربری (UserName) و رمز عبور (PassWord) اکانت گیت شما را می خواهد. اگر این فرم را کنسل کنید، اطلاعات user و pass را در همان command line گیت از شما مجددا می پرسد. پس از لاگین به اکانت گیت هاب، پروژه ما داخل مخزن وب سایت Github لود می شود.

**دسترسی به repository های پروژه در github:**

برای دسترسی به repository های خود در سایت گیت هاب، می توانید به لینک Repositories واقع در پروفایل خود مراجعه کنید. در این بخش با کلیک روی نام مخزن پروژه خود (در این مثال project-test) می توانید فایل ها و فولدرهای پروژه خود را مشاهده کنید.

نکته: فایل ها و فولدرهایی را که در فایل .gitignores تعریف کرده اید، در مخزن گیت هاب لود نخواهند شد. در این مثال، فولدرهای js و node\_modules و فایل های متنی txt لود نمی شوند.

یکی از امکانات جالب و کاربردی گیت اینست که می توانید لیست تمام کامیت (commit) ها را مشاهده کنید و با کلیک روی لینک آنها، قادر خواهید بود تغییراتی که در آن کامیت خاص انجام داده اید را بصورت زیبا و رنگی مشاهده کنید.

**افزودن یک شاخه (branch) به مخزن سایت گیت هاب:**

برای افزودن یک شاخه (branch) به مخزن سایت گیت هاب، دستور git push –u origin develop را در خط فرمان گیت (Git Command Line) اجرا کنید. با اجرای این دستور، تعداد شاخه های پروژه ما در مخزن گیت هاب، 2 عدد می شود (master و develop)

به این ترتیب می توانید اطلاعات گیت (Git) را به گیت‌هاب (Github) منتقل کنید.

**نکته**: دقت کنید که هنگام درج اطلاعات خود در گیت، ایمیلی که در سایت Github وارد می کنید با آدرس ایمیلی که در کامند لاین گیت تعریف می کنید یکی باشد تا گیت شما را بعنوان اولین توسعه‌دهنده پروژه‌تان شناسایی کند.

**نحوه لود پروژه از گیت هاب به سیستم لوکال**

در این بخش، چگونگی لود پروژه از وب سایت Github به سیستم لوکال کاربران را آموزش می دهیم. یعنی کار برعکس آپلود پروژه در گیتهاب.

**ویرایش فایل ها در وب سایت گیت‌هاب:**

برای ویرایش و توسعه فایل ها در وب سایت گیت‌هاب، روی نام فایل دلخواه خود کلیک کنید تا بتوانید کدهای آن فایل را مشاهده کنید. سپس روی آیکون مداد که در بالای این بخش قرار دارد کلیک کرده تا کدهای فایل به مد ویرایش بروند.

اکنون می توانید تغییرات لازم را روی آن انجام داده و در نهایت commit کنید. به هر commit باید یک نام اختصاص دهید و توضیحات کامیت اختیاری می باشد. در آخر دکمه commit را کلیک کنید.

**استفاده از دستور pull برای لود پروژه:**

برای اینکار باید از دستور pull استفاده کنید. برای مشاهده راهنمای استفاده از این دستورالعمل، دستور git pull –help را اجرا کنید.

 برای شروع لود پروژه (ادغام یا merge کردن پروژه واقع در وب سایت گیت هاب با پروژه واقع در سیستم خود) ابتدا روی فولدر حاوی پروژه خود راست کلیک کرده سپس git bash here کلیک کنید. در این مرحله کافیست دستور git pull origin را در command line نمایش داده شده نوشته و اجرا کنید. با اینکار، تمام فایلها و فولدرهای پروژه شما با پروژه موجود در گیت هاب، سینک (Sync) می شوند.

git pull origin

تصویر زیر:

**دانلود پروژه واقع در گیت هاب:**

اگر فولدر پروژه شما از روی سیستم‌تان به هر دلیلی پاک شود، در صورتی که آنرا در وب سایت Github قرار داده باشید، می توانید مجددا آنرا دانلود کنید. بدین منظور، روی دکمه سبزرنگ Clone or Download کلیک کنید. در پنجره ظاهر شده، می توانید از دو طریق، پروژه خود را از گیت هاب دریافت کنید. یکی اینکه آنرا در قالب یک فایل Zip دانلود کنید. دیگری اینکه url ارائه شده در این پنجره را کپی کرده (مثلا آدرس <https://github.com/Safari66/project-test.git>) سپس در فولدری که می خواهید پروژه تان دانلود شود راست کلیک کرده و git bash here را انتخاب نمائید. در این مرحله باید دستور <https://github.com/Safari66/project-test.git> git clone  را اجرا کنید.

**نکته**: در صورت تمایل می توانید نام فولدر این پروژه را در سیستم خود تغییر دهید. برای اینکار، پس از دستور فوق، می توانید نام دلخواه را برای پروژه خود تعریف کنید و سپس آنرا اجرا کنید.

اکنون اگر توسط دستور cd به فولدر همین پروژه منتقل شوید، و سپس دستور لاگ (git log –oneline) را اجرا کنید، می توانید تمام commit های انجام شده را ملاحظه کنید.

بنابراین نباید از پاک شدن فولدر پروژه تان از روی سیستم نگران شوید. زیرا می توانید براحتی آنرا از گیت هاب دانلود نمائید. (البته در صورتیکه قبلا آنرا به سایت Github منتقل کرده باشید!)

کاربرد دیگر دستور pull و clone اینست که می توانید ویرایش هایی که دیگر توسعه دهندگان روی پروژه شما اعمال کرده اند را به پروژه خود انتقال دهید و از آن بهره ببرید.

**چگونگی همکاری در پروژه‌های وب‌سایت گیت‌هاب**

**قابلیت Fork در گیت هاب:**

ممکن است بخواهید در برخی از پروژه های موجود در وب سایت گیت‌هاب (Github) همکاری داشته باشید و شما نیز بعنوان یکی از توسعه‌دهندگان آن پروژه مشغول بکار شوید. گیت این امکان را دارد که توسط قابلیت Fork ، چندین توسعه دهنده روی یک پروژه کار کنند. برای هر یک از این توسعه دهندگان یک مخزن (repository) مجزا درنظر گرفته می شود.

پس از لاگین به اکانت گیت هاب خود، در صفحه پروژه موردنظر، روی دکمه Fork کلیک کنید. Fork به معنی چنگال است. پس از کلیک روی Fork، یک نسخه جداگانه از پروژه بنام کاربری شما ایجاد می شود. اکنون اگر روی دکمه سبز رنگ Clone or Download کلیک کنید، آدرس پروژه با نام کاربری شما بازنویسی شده است. بعنوان مثال آدرس زیر:

https://github.com/NewGitUser/project-test.git

**دانلود پروژه از گیت هاب:**

علاوه بر اینکه می توانید پروژه موردنظرتان را با کلیک روی دکمه Download ، دانلود کنید، می توانید با اجرای دستور git clone https://github.com/New-Username/project-test.git  در فولدر موردنظرتان، پروژه ای را که می خواهید توسعه دهید را لود کنید.

git clone https://github.com/New-Username/project-test.git

اکنون تغییراتی در فایل ها و فولدرهای پروژه مذکور اعمال کنید (مثلا فولدری اضافه کنید و یا خطی به یکی از فایلها اضافه کنید) سپس در فولدر این پروژه راست کلیک کرده و git bash here را انتخاب کنید. اکنون اگر git status بگیریم، تغییراتی که انجام داده ایم را مشاهده خواهیم کرد. در این مرحله، تغییرات را باید به stage اضافه کنیم (با دستور git add .) سپس این تغییرات را از stage به مخزن (repository) کامیت (commit) کنیم (توسط دستور git commit –m “add some changes to project”)

**آپلود پروژه توسعه داده شده به گیت هاب:**

تا کنون پروژه ای را از گیت هاب گرفته ایم (توسط Fork) و تغییراتی روی آن اعمال کرده ایم. حال می خواهیم این پروژه را مجددا به گیت هاب منتقل کنیم. ابتدا باید به این پروژه در گیت هاب ریموت (Remote) بزنیم. بدین منظور، دستور  git remote add origin-NewGitUser https://github.com/NewGitUser/project-test.git را اجرا کنید.

git remote add origin-NewGitUser https://github.com/NewGitUser/project-test.git

دقت کنید که در دستور فوق، NewGitUser همان نام کاربری توسعه دهنده دوم می باشد که می خواهد پروژه آپدیت شده را مجددا در گیت آپلود کند.

سپس دستور git push origin-NewGitUser را اجرا نمائید. با اجرای این دستور، تمام تغییراتی که در فایل ها و فولدرهای پروژه اعمال کرده ایم، به پروژه اصلی واقع در گیت هاب (البته Fork مربوط به ما) منتقل خواهند شد.

git push origin-NewGitUser

در صورتی که در حین ریموت زدن به گیت هاب، نیاز به ورود یوزر و پسورد بود، اطلاعات نام کاربری و رمز عبور حساب NewGitUser یعنی توسعه دهنده دوم را وارد نمائید.

**اعمال تغییرات انجام شده روی پروژه اصلی:**

تغییراتی که در پروژه اعمال کرده اید، در پروژه اصلی قابل مشاهده نمی باشد و این تغییرات فقط در پروفایل خودتان قابل دسترسی و مشاهده می باشد. برای اینکه تغییرات NewGitUser روی پروژه اصلی یعنی یوزر Safari66 نیز اعمال شوند، باید یک درخواست pull برای توسعه دهنده اول یعنی Safari66 ارسال کنید. بدین منظور در پروفایل Safari66 روی دکمه سبزرنگ New Pull Request کلیک کنید. در صفحه ای که نمایش داده می شود، باید یک عنوان و یک توضیح برای درخواست خود تعریف کنید و در نهاینت روی دکمه Create Pull Request کلیک کنید. معنی این درخواست اینست که “من یعنی NewGitUser تغییراتی را در جهت بهبود کارایی پروژه شما یعنی Safari66 اعمال نمودم و الان درخواست می کنم تا این تغییرات را در پروژه اصلی خود ادغام (merge) کنید“

**ایجاد یک درخواست pull:**

در صورتی که کاربر اول یعنی Safari66 به اکانت Github خود لاگین کند، در پروفایل خود و با کلیک روی نام پروژه، در مقابل تب Pull Request مقدار 1 را مشاهده می کند. با کلیک روی این تب، Safari66 لیست درخواست های pull را مشاهده می کند که فعلا یک درخواست دارد. این لیست شامل عنوان درخواست است که بصورت لینک می باشد. کاربر Safari66 با کلیک روی لینک هر درخواست، توضیحات مربوط به درخواست کاربر NewGitUser را مشاهده خواهد کرد.

**ادغام درخواست pull:**

Safari66 یعنی توسعه دهنده اول و اصلی، می تواند این درخواست را ببندد (Close) و یا اینکه برای کاربر دوم (NewGitUser) کامنتی ارسال کند و مشکلات و نواقص پروژه را به او اعلام کند. در صورت close کردن درخواست کاربر دوم توسط کاربر اول، تغییراتی که کاربر دوم انجام داده است با پروژه اصلی ادغام نخواهد شد. در صورتی که کاربر اصلی یعنی Safari66 بخواهد تغییرات NewGitUser را در پروژه خود ادغام کند، باید روی دکمه سبزرنگ Merge Pull Request کلیک کند. با کلیک روی این دکمه، گزینه های زیر نمایش داده می شود:

* Create a merge commit
* Squash and merge
* Rebase and merge

اولین گزینه (Create a merge commit) را انتخاب کنید. باکسی برای شما باز می شود که باید عنوان کامیت و شرح آنرا درج کنید. در نهایت Confirm Merge را بفشارید.

با تائید ادغام دو پروژه، تغییراتی که NewGitUser در پروژه اعمال کرده در پروژه اصلی قابل مشاهده و در دسترس عموم می باشد.

**نکته**: برای پاک کردن یک ریموت و زدن ریموتی دیگر، بصورت زیر عمل کنید:

* دستور git remote remove origin را برای حذف ریموت اول اجرا کنید.

git remote remove origin

* دستور git remote add origin-NewGitUser https://github.com/NewGitUser/project-test.git را اجرا کنید.

git remote add origin-NewGitUser https://github.com/NewGitUser/project-test.git

بدین ترتیب با انجام مراحل مذکور در این مقاله، قادر خواهید بود در هر یک از پروژه های واقع در سایت گیت هاب Github بعنوان یک توسعه دهنده مشارکت داشته باشید. در اینجا، آموزش جامع گیت و گیت هاب (Git – Github) به پایان می رسد.

امیدوارم این آموزش که در 9 بخش برای شما آماده شده بود، مورد استفاده شما عزیزان قرار گرفته باشد و همچنان با وب سایت **Mohtavaban.com** همراه باشید. لطفا در صورت داشتن هرگونه سوال و یا پیشنهاد و انتقادی، با ایمیل contact@mohtavaban.com با ما در تماس باشید. یا کامنت خود را در بخش ارسال دیدگاه پایین همین مقاله برای ما ارسال کنید. لطفا ما را در شبکه های مجازی با آیدی **mohtavaban** دنبال کنید و در صورتی که مقاله را دوست داشتید یک امتیاز از یک ستاره (ضعیف) تا پنج ستاره (قوی) به مقاله بدهید.