**این فایل شامل سرفصل­های زیر است**

* **لیست Consept های مرتبط با git**
* **دستورات مربوط به Git CLI یا Git Bash**
* **نکات و دستورات کلی**
* **لیست موارد گیت که مرتبط با VS code است**
* **لیست ابزار­های کار با git به کمک رابط کاربری گرافیکی(GUI)**
* **الگوریتم نامگذاری branch ها در یک پروژه بزرگ**
* **پروتوکل کلی کار با گیت و گیت­هاب – شماره 1**
* **مواردی که باید تحقیق شود**

**لیست Consept های مرتبط با git**

Remote

Add

Clone

Fork

Star

Checkout

Branch

Pull request

Fetch

Local system is git != Cloud system is github

Main branch = Master branch

U/A/M near to file(U = Untrack, A = Added index, M = Modify)

.git folder in project root

Go to Github.com and create repository

Go to Github.com and get repository link

Git merge

* merge main branch into feature branch
* merge feature branch into main branch

Git rebase

* Git rebase For main into feature
* Git rebase and fast-forward merge

Git merge or git rebase with Squash commits -> sigle merge commit – lose finde details

**دستورات مربوط به Git CLI یا Git Bash**

قبل از هر چیز اگر برای اولین بار است روی سیستم خود گیت را نصب می کنید حتما دو دستور زیر را بزنید تا نام و ایمیل شما به شکل global در گیت ذخیره شود

git config –global user.email **<**[**you@example.com**](mailto:you@example.com)**>**

git config –global user.name “**<your name>**”

تنظیم گیت

git init

اضافه کردن فایل­های readme, licence, gitignore

git add README.md

git add LICENSE.md

git add .gitignore

اضافه کردن هر فایل دیگری در git

git add index.html

تعریف یک Branch جدید

git branch <**BRANCH\_NAME**>

git branch –m <**BRANCH\_NAME**>

git branch –M <**BRANCH\_NAME**>

دیدن لیست تمام Branch های موجود در پروژه

git branch –all

دیدن لیست تمام branch که روی سیستم لوکال می باشد

git branch

دیدن لیست تمام Branch ها که در قالب remote می باشد.

git branch --remote

برای فهمیدن آرگومان –m و –M که در دو دستور بالا به کار رفته است متن زیر را مطالعه کنید.

With a -m or -M option, <oldbranch> will be renamed to <newbranch>. If <oldbranch> had a corresponding reflog, it is renamed to match <newbranch>, and a reflog entry is created to remember the branch renaming. If <newbranch> exists, -M must be used to force the rename to happen.

برای تغییر نام یک branch از دستور زیر استفاده می کنیم.

git branch –m <**BRANCH\_OLD\_NAME**> <**BRANCH\_NEW\_NAME**>

git branch –M <**BRANCH\_OLD\_NAME**> <**BRANCH\_NEW\_NAME**>

اگر در یک branch مشخص باشیم و بخواهیم نام همان branch را تغییر دهیم از دستور زیر استفاده می شود.

git branch –m <**BRANCH\_NEW\_NAME**>

git branch –M <**BRANCH\_NEW\_NAME**>

اضافه کردن یک remote جدید به git

git remote add <**REMOTE\_NAME**><**REMOTE\_URL**>

حذف یک remote مشخص از git

git remote remove <**REMOTE\_NAME**>

تغییر remote\_url در یک remote که در git تعریف شده است

git remote set-url <**REMOTE\_NAME**><**NEW\_REMOTE\_URL**>

لیست تمامی remote ها

git remote -v

لیست تمامی کامند­های مرتبط با گیت

git help

git –help

توضیحات بیشتر در رابطه با یک command مشخص

git <**COMMAND\_NAME**> –help

git help <**COMMAN\_NAME**>

قبل از آنکه بخواهیم کد های خود رو روی ورژن هندلر آنلاین ذخریه کنیم(منظور روی یک سیستم کلود مثل github) باید از یکی از دو دستور زیر استفاده کنیم

دستور اول:

Git fetch <**REMOTE\_NAME**> **<BRANCH\_NAME**>

Git merge <**REMOTE\_NAME**><**YOUR\_BRANCH\_NAME\_IN\_LOCAL\_MACHIN**>

دستور دوم:

git pull <**REMOTE\_NAME**> **<BRANCH\_NAME**>

اگر هنگام دستور pull با ارور "fatal: refusing to merge unrelated histories" مواجه شدیدی از پرچم --allow-unrelated-histories در کنار دستور pull استفاده کنید.

git pull <**REMOTE\_NAME**> **<BRANCH\_NAME**> --allow-unrelated-histories

برای آپلود کردن پروژه از روی ریپازیتور لوکال به ریپازیتوری ریموت از دستور زیر استفاده می کنیم.

git push <**REMOTE\_NAME**><**BRANCH\_NAME**>

git push -u <**REMOTE\_NAME**><**BRANCH\_NAME**>

اگر با ارور "fatal: refusing to merge unrelated histories" مواجه شدیدی دستور زیر را وارد کنید.

git push <**REMOTE\_NAME**><**BRANCH\_NAME**> --allow-unrelated-histories

پرچم –f در اینجا به معنی force می­باشد.

git push <**REMOTE\_NAME**> **<BRANCH\_NAME**> -f

عبارت –u در بالا به معنی –set-upstream می باشد. توضیحات آن در زیر داده شده است. لازم به ذکر است در صورتی که مفهوم upstream را نمی دانید بهتر است متن انگلیسی زیر از پارگاراف دوم به بعد را نیز مطالعه کنید.

-u = --set-upstream

For every branch that is up to date or successfully pushed, add upstream (tracking) reference, used by argument-less [git-pull(1)](file:///C:\Program%20Files\Git\mingw64\share\doc\git-doc\git-pull.html) and other commands. For more information, see branch.<name>.merge in [git-config(1)](file:///C:\Program%20Files\Git\mingw64\share\doc\git-doc\git-config.html).

Like the literal meaning, upstream refers to something which points to where it originated from. In the git world, upstream refers to the original repo or a branch. For example, when you clone from Github, the remote Github repo is upstream for the cloned local copy.

What is Git Upstream Branch? When you want to checkout a branch in git from a remote repository such as GitHub or Bitbucket, the “Upstream Branch” is the remote branch hosted on Github or Bitbucket. It's the branch you fetch/pull from whenever you issue a plain git fetch/git pull basically

You must configure a remote that points to the upstream repository in Git to sync changes you make in a fork with the original repository. This also allows you to sync changes made in the original repository with the fork.

With an upstream branch set, you can simply use the shorthand commands "git pull" and "git push" - instead of having to think about the exact parameters like in "git push origin development". Git can now also tell you about unsynced commits which you haven't pushed or pulled, yet.

همچنین می توان از دستور زیر

برای کامیت کردن یک تغییرات از دستور زیر استفاده می کنیم

git commit -m "first commit"

git commit -message "first commit"

هر دو دستور بالا یکی است می توان به جای –m از –message استفاده کرد عبارت داخل "---" که معادل first commit است می توان هر چیزی باشد. این عبارت کامنت مربوط به کامیت مورد نظر را مشخص می کند. اگر چند بار از –m یا –message استفاده کنیم فکر کنم در چند خط متفاوت می توان کامنت نوشت این رو توی داکیومنت گیت هاب خوندم.

برای جابه­جا شدن از یک branch به branch دیگر دستور زیر را استفاده می کنیم. در زیر <**BRANCH\_NAME**> همان branch می­باشد که به آن می­خواهیم برویم

git checkout <**BRANCH\_NAME**>

جهت مشاهده لاگ ها و کامیت­های انجام شده می­توانید از دستور زیر استفاده کنید

git log

جهت مشاهده ورژن نصب شده از گیت می­توانید از دستور زیر استفاده کنید.

git --version

جهت مشاهده وضعیت تغییرات از این دستور استفاده می کنیم

git status

جهت اضافه کرده تمامی فایل هایی که در وضعیت modify می باشد(فایل هایی که تغییر دادیم) از حالت change به حالت stage از این دستور استفاده می کنیم

git add .

دستور های دیگر در زیر آمده است.

git rm –cached .env

git restore .env

git commit –m “first commit” or git commit –message “first commit”

git commit –m “second commit” or git commit –message “second commit”

git diff .env

git reset --hard “”

**نکات و دستورات کلی**

می توان Branch جدید را خیلی ساده از سیستم لوکال درست کرد یا از سایت خود گیت هاب ولی به نظرم بهرته که روی لوکال باشه

برای ادغام کردن دو Branch مختلف می توانید به صورت local از دستور Merg استفاده کرده ویا به صورتی که غیر لوکال باشه از دستور دیگری به نام دستور Pull request استفاده کنید.

از دستور rebase در repository های public هیچ وقت استفاده نکن. فقط برای repository های private یا آنهایی که لوکال هست و با کسی share نیست

حتما هنگامی که در branch با نام main یا master هستید هنگام انجام merge دقت لازم را بکنید وگرنه ممکنه دچار مشکل شوید.

نکته هر گزر روی سیستم لوکال خود از branch با نام main استفاده نکنید و به جای آن از branch با نام master استفاده کنید. علت این کار این است که وقت از دستور pull استفاده می کنید و branch با نام main را به شکل remote از گیت­ها می گیرید دقیقا کد ها از branch با نام main رو سرور cloude وبسایت github کپی می شود و در سیستم ما قرار می گیرد. در این حالت branch با نام mian در سیستم لوکال ما نیز ایجاد می شود و به همین دلیل اگر بخواهیم دو branch با نام main را در سیستم لوکال با هم merge کنیم به مشکل می خوریم.

**لیست موارد گیت که مرتبط با VS code است**

VS code -> Setting -> git eable

VS code -> View -> command palette ->

نصب اکستنشن git lens. پس از نصب باید دو کار کرد. ابتدا وارد command palatte شوید و عبارت زیر را تایپ کنید

gitlens enable interactive rebase editor

سپس در ترمینال عبارت زیر را تایپ کنید:

git config --global core.editor "code --wait"

در این حالت هنگام merge یا rebase کردن از flag معادل –i که معادل واژه interactive است باید استفاده کنید. مثلا

Git rebase –I main

**لیست ابزار­های کار با git به کمک رابط کاربری گرافیکی(GUI)**

در زیر لیک مروبط به سایت رسمی گیت می­باشد که در آن خودش به شما جند ابزار معرفی کرده

<https://git-scm.com/download/gui/windows>

ابزار زیر با بررسی­هایی که کردم بهترین ابزار برای یادگیری است

[GitKraken](https://www.slant.co/topics/4985/viewpoints/31/~visual-git-guis~gitkraken)

[GitHub Desktop](https://desktop.github.com/)

ابزار زیر رابط کاربری گرافیکی بهتری برای نمایش branch ها دارد.

[gmaster](https://www.slant.co/topics/4985/viewpoints/18/~visual-git-guis~gmaster)

gitVine

[GitKraken](https://www.slant.co/topics/4985/viewpoints/31/~visual-git-guis~gitkraken)

[Fork](https://git-fork.com/)

Ungit

gitAtomic

|  |  |
| --- | --- |
| لیست ابزار­ها | |
| [Anchorpoint](https://anchorpoint.app/) |  |
| [Aurees](https://aurees.com/) |  |
| [CommandGit](https://commandgit.com/) |  |
| [Cong](http://cong.tools/) |  |
| [Cycligent Git Tool](https://www.slant.co/topics/4985/viewpoints/15/~visual-git-guis~cycligent-git-tool) |  |
| Fork |  |
| [ForkHub](https://alternativeto.net/software/forkhub/about/) |  |
| [Fugitive](https://github.com/tpope/vim-fugitive) |  |
| Git Cola |  |
| [Git Extensions](https://gitextensions.github.io/) |  |
| Git Force |  |
| [GitAhead (no longer under active development)](https://gitahead.github.io/gitahead.com/) |  |
| [GitAtomic](https://www.slant.co/topics/4985/viewpoints/24/~visual-git-guis~gitatomic) |  |
| gitbox |  |
| [GitBreeze](https://gitbreeze.dev/) |  |
| [git-cola](https://git-cola.github.io/) |  |
| [GitEye](https://alternativeto.net/software/giteye/about/) |  |
| [GitFinder](https://gitfinder.com/) |  |
| [GitForce](https://blog.devart.com/best-git-gui-clients-for-windows.html#GitForce) |  |
| [Gitfox](https://alternativeto.net/software/gitfox/about/) |  |
| Gitg |  |
| [Gitgui](https://github.com/jjustra/gitgui) |  |
| [GitHub Desktop](https://desktop.github.com/) |  |
| [GitKraken](https://www.gitkraken.com/) |  |
| [GitKraken](https://www.slant.co/topics/4985/viewpoints/31/~visual-git-guis~gitkraken) |  |
| [Gitnuro](https://gitnuro.com/) |  |
| [gitonic](https://github.com/kr-g/gitonic) |  |
| Gittyup |  |
| [GitUI](https://extrawurst.itch.io/gitui) |  |
| GitUp |  |
| [GitVine](https://insanesharpness.gitlab.io/GitVine) |  |
| [git-webui](https://alternativeto.net/software/git-webui/about/) |  |
| GitX |  |
| [Glint](https://glint.info/) |  |
| [gmaster](https://www.slant.co/topics/4985/viewpoints/18/~visual-git-guis~gmaster) |  |
| [Guitar](https://github.com/soramimi/Guitar) |  |
| [IntelliJ](https://www.slant.co/topics/4985/viewpoints/26/~visual-git-guis~intellij) |  |
| v[isual Studio Code](https://www.slant.co/topics/4985/viewpoints/25/~visual-git-guis~visual-studio-code) |  |
| [LabCoat](https://alternativeto.net/software/labcoat/about/) |  |
| [LazyGit](https://github.com/jesseduffield/lazygit) |  |
| [MeGit (based on EGit)](https://github.com/eclipsesource/megit) |  |
| [NitroGit](http://nitrogit.net/) |  |
| [Pragma-git](https://pragma-git.github.io/) |  |
| [QGit](https://github.com/tibirna/qgit) |  |
| [RepoZ](https://github.com/awaescher/RepoZ/) |  |
| SmartGit |  |
| [SourceTree](https://www.sourcetreeapp.com/) |  |
| [Sublime Merge](https://www.slant.co/topics/4985/viewpoints/21/~visual-git-guis~sublime-merge) |  |
| [Thermal](https://alternativeto.net/software/thermal/about/) |  |
| tig |  |
| [TortoiseGit](https://www.slant.co/topics/4985/viewpoints/2/~visual-git-guis~tortoisegit) |  |
| [Tower](https://www.slant.co/topics/4985/viewpoints/3/~visual-git-guis~tower) |  |
| [Ungit](https://www.slant.co/topics/4985/viewpoints/14/~visual-git-guis~ungit) |  |
| [Vershd](https://www.slant.co/topics/4985/viewpoints/27/~visual-git-guis~vershd) |  |

**الگوریتم نامگذاری branch ها در یک پروژه بزرگ**

نصب اکتسنشن git lens git superchrchg

پروتوکل نامگذاری هر branch در کسب و کار­های بزرگ که شامل 10000 نفرم برنامه نویس می­باشد که ممکن است سیستم­های لوکال و کلود مختلفی درگیر این پروژه باشند. چند الگوریم برای نام­گذاری­ها داریم که به شرح زیر است:

1. الگوی نام­گذاری branch ها
2. الگوی نوشتن message مرتبط با هر commite

از الگوریتم کلی زیر برای نام گذاری هر branch استفاده می کنیم.

user\_{*user\_number*}\_{*local\_cloud*}\_[{subbranche\_{*subbranche\_number*}\_[*explantion*]}]

در زیر هر کدام از پارامتر­های الگوی بالا توضیح داده شده است:

user\_number -> can be between **00001** to **10000 (or more)**

system\_model -> can be **local** or **cloud**

subbranche\_number -> can be between **00001** to **10000 (or more)**

explantion -> can be name of **feature\_00001** to **feature\_10000 (or more)** or other names like **feature\_name**

در نام­گذاری قوانین زیر حتما رعایت شود

* هر پروژه شامل یک branch اصلی می باشد که نام آن main یا master است. ابتدا یک Branch را به عنوان branch اصلی انتخاب می­کنیم و نام آن را حتما main یا master می گذاریم. این branch که branch اصلی می­باشد حتما باید روی سیستم های کلود یا ابری باشد. مثلا گیت­ها مناسب این کار است
* شیوه نامگذاری کاربر محور است و به همین دلیل اول هر نام عبارت user\_00001 یا مشابه آن می­آید.
* هر کاربر شامل یک branch اصلی می­باشد که در سیستم آن در قالب لوکال است و به شکل زیر نامگذاری می­شود.
* Explanation می­تواند هیچ چیزی نباشد یا شماره feature مورد نظر باشد یا اسم feature مورد نظر
* در زیر یک مثال جامع از یک پروژه بزرگ زده می­شود که 3 کاربر(user) دارد.

|  |  |
| --- | --- |
| main(or master) | این همان branch اصلی است. |
| user\_00001 | |
| user\_00001\_local\_main | این همان branch اصلی کابر شماره 00001 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است. |
| user\_00001\_local\_main\_subbranch\_00001\_featuer\_00001 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00001 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با featuer\_00001 است |
| user\_00001\_local\_main\_subbranch\_00002\_featuer\_00002 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00001 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با featuer\_00002 است |
| user\_00001\_local\_main\_subbranch\_00003\_featuer\_00001 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00001 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با featuer\_00003 است |
| user\_00002 | |
| user\_00002\_local\_main | این همان branch اصلی کابر شماره 00002 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است. |
| user\_00002\_local\_main\_subbranch\_00001\_login\_feature | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00002 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با login\_feature است |
| user\_00002\_local\_main\_subbranch\_00002\_signup | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00002 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با signup است |
| user\_00003 | |
| user\_00003\_local\_main | این همان branch اصلی کابر شماره 00003 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است. |
| user\_00003\_local\_main\_subbranch\_00001\_profile\_panel | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00003 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا explanation برابر با profile\_panel است |
| user\_00003\_local\_main\_subbranch\_00002 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00003 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا چیزی برای explanation تعریف نشده |
| user\_00003\_local\_main\_subbranch\_00003 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00003 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا چیزی برای explanation تعریف نشده |
| user\_00003\_local\_main\_subbranch\_00004 | این همان subbranch ایجاد شده از branch اصلی کابر شماره 00003 می­باشد که بر روی سیستم local (مثلا لپ­تاپ) است.  همانطور که می­بینید در اینجا چیزی برای explanation تعریف نشده |

**الگوریتم نامگذاری commit ها در یک پروژه بزرگ**

برای هر کامیت یک پیام نیاز است که الگوی نام­گذاری پیام به شرح زیر وجود دارد.

از الگوریتم کلی زیر برای نام گذاری هر branch استفاده می کنیم.

user\_{*user\_number*}\_{*local\_cloud*}\_[{subbranche\_{*subbranche\_number*}\_[*explantion*]}] write the message in this commit:

در زیر هر کدام از پارامتر­های الگوی بالا توضیح داده شده است:

user\_number -> can be between **00001** to **10000 (or more)**

system\_model -> can be **local** or **cloud**

subbranche\_number -> can be between **00001** to **10000 (or more)**

explantion -> can be name of **feature\_00001** to **feature\_10000 (or more)** or other names like **feature\_name**

**پروتوکل کلی کار با گیت و گیت­هاب – شماره 1**

<**BRANCH\_NAME**>= main\_local or master

<**REMOTE\_NAME**> = origin

<**GITHUB\_USERNAME**> = bardiadelagah

<**REPOSITORY\_NAME**> = ?

<**REMOTE\_URL**> = [https://github.com/<**GITHUB\_USERNAME**>/<**REPOSITORY\_NAME**>.git](https://github.com/%3cGITHUB_USERNAME%3e/repository_name.git)

<**BUFFER\_NUMBER**> = 2000000000 or 2097152000 or 524288000 or 1048576000

* Make a github repository on github.com
* Git HTTPS url for repository
* git init
* git add .
* git commit -m “first commit”
* echo “” > README.me
* echo “” > LICENSE.md
* echo “” > .gitignore
* git branch -M <**BRANCH\_NAME**>
* git status
* git add .
* git commit -m "first commit"
* حتما باید اولین کامیت را انجام داد وگرنه نمی­توانید برنچ جدید تعریف کنید و صرفا می­توانید اسم برنچ اول را تغییر دهید.
* git status
* git log
* git remote add <**REMOTE\_NAME**><**REMOTE\_URL**>
* git remote -v
* git pull <**REMOTE\_NAME**><**BRANCH\_NAME**>

اگر در مرحله قبل با ارور "fatal: refusing to merge unrelated histories" مواجه شدیدی دستور زیر را وارد کنید. همچنین اگر پس از دستور pull متوجه شدید که کد­های شما با کد­هایی که به کمک دستور pull در سیستم لوکال شما وارده شده است merge نشده است باید از یکی از دو دستور زیر استفاده کنید

دستور اول:

* git pull <**REMOTE\_NAME**> **<BRANCH\_NAME**> --allow-unrelated-histories

دستور دوم:

* git checkout main
* git merge **<BRANCH\_NAME**> --allow-unrelated-histories

پس از استفاده از دو دستور فوق باید push کنیم.

* git commit -m "first commit"
* git checkout main
* git push -u <**REMOTE\_NAME**> **<BRANCH\_NAME**>

اگر در مرحله قبل با ارور " error: unable to rewind rpc post data - try increasing http.postBuffer" یا ارور " fatal: the remote end hung up unexpectedly" یا " error: RPC failed; curl 56 Recv failure: Connection was reset" مواجه شدید دستور زیر را بزنید

* git config --global http.postBuffer <**BUFFER\_NUMBER**>
* git config --global https.postBuffer <**BUFFER\_NUMBER**>

اگر در مرحله قبل با ارور مواجه شدید دستور زیر را بزنید پرچم –f در اینجا به معنی force می­باشد.

* git push -u <**REMOTE\_NAME**> **<BRANCH\_NAME**> -f

اگر باز هم در مرحله قبل با ارور مواجه شدید دستور زیر را بزنید.

* git push <**REMOTE\_NAME**> **<BRANCH\_NAME**> -f

اگر باز هم در مرحله قبل با ارور مواجه شدید دستور زیر را بزنید.

* git push <**REMOTE\_NAME**> **<BRANCH\_NAME**>

**مواردی که باید تحقیق شود**

راجب اینکه چطور یک نفرم پروژه پابلیک منو توش دستکاری نکنه سرچ کن

راجب واژه head 🡪 main یا مثلا head -> feature ویدیو بساز

راجب کد کانفلیکت حتما توضیح بده و قبلش مطالعه کن روی حالت لوکال

راجب دستور pull request و اینکه باید ادمین روی گیت هاب یکسری کار ها انجام بده تحقیق کن.

راجب echo توضیح بده

echo "# aaa" >> README.md

در رابطه با upstream تحقیق کن و دستور -u

git push -u <**REMOTE\_NAME**> **<BRANCH\_NAME**>

problem with same name in local git branch and github branch

Local branch name conflicts with remote branch name

<https://www.datacamp.com/tutorial/git-push-pull>

<https://docs.github.com/en/get-started/using-git/dealing-with-non-fast-forward-errors>

راج رفع ارور the remote end hung up unexpectedly و لینک های زیر تحقیق شود

<https://stackoverflow.com/questions/70276233/github-issue-fatal-the-remote-end-hung-up-unexpectedly>

<https://stackoverflow.com/questions/15240815/git-fatal-the-remote-end-hung-up-unexpectedly>

راجب موارد زیر و تفاوت هاش تحقیق شود

GitHub, Bitbucket, GitLab

راجب دو متن زیر و لینک های زیر که احتمالا مربوط به آن است تحقیق شود

how to add github repository to visual studio code

the repository has no remotes configured to push

<https://code.visualstudio.com/docs/sourcecontrol/intro-to-git#:~:text=Open%20a%20GitHub%20repository%20in%20a%20codespace&text=Install%20the%20GitHub%20Codespaces%20extension,open%20in%20a%20new%20window>.

<https://code.visualstudio.com/docs/sourcecontrol/overview#_git-support>

<https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories>

<https://code.visualstudio.com/docs/sourcecontrol/overview#_git-support>

<https://docs.github.com/en/get-started/using-git/pushing-commits-to-a-remote-repository>

<https://docs.github.com/en/get-started/using-git/dealing-with-non-fast-forward-errors>

<https://docs.github.com/en/get-started/using-git/pushing-commits-to-a-remote-repository>

<https://docs.github.com/en/get-started/using-git/dealing-with-non-fast-forward-errors>

<https://docs.github.com/en/get-started/quickstart>

<https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories>

<https://docs.github.com/en>

<https://stackoverflow.com/questions/24114676/git-error-failed-to-push-some-refs-to-remote>

<https://komodor.com/learn/how-to-fix-failed-to-push-some-refs-to-git-errors/>

<https://stackoverflow.com/questions/39399804/updates-were-rejected-because-the-tip-of-your-current-branch-is-behind-its-remot>

.