Git clone

git clone --recursive https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/architecture-app

git config --global http.proxy http://cloudproxy.zf-world.com:8080

git pull --recurse-submodules

git lfs lock -->
```
git lfs lock <filename>
git lfs unlock <filename>
```

From <https://confluence.atlassian.com/bitbucketserver/working-with-git-lfs-files-970595880.html>

git lfs locks

:qw

From <

[2:52 PM] Karbowiak Lukasz CZS UDAS131
Task 141547 Example task for PR (azure.com)

Info:
To update commit -> task, commit message must contain task number
eg commit message:
"Task #123123 add user right"
Mandatory is **#123123**

From <https://teams.microsoft.com/multi-window/?agent=electron&version=22060614805>

```
ZF-WORLD+Z0145624@HDRC01548 MINGW64 /d/Sandbox/GIT_Sandbox/MRGen7_Parts/architecture-app/020_UmlModel (master)
$ git pull
Already up to date.

ZF-WORLD+Z0145624@HDRC01548 MINGW64 /d/Sandbox/GIT_Sandbox/MRGen7_Parts/architecture-app/020_UmlModel (master)
$ git pull --recurse-submodules
Fetching submodule 900_Profiles
Already up to date.
```

```
/ app-r5
git pull
git branch -a
git switch orchardr/task_155784_cv_os_updates
git add .
git commit -m "<message>"
git fetch origin integration
git merge integration
git push origin HEAD:orchardr/task_155784_cv_os_updates
```

Rhapsody architecture git workflow - Overview (azure.com)

Git Command MRGen7

To copy from BAASH terminal in windows just select and write click.
''' Comments
variable_V : To be replace by meaningful content
- For line brake, use two or more space and then **Enter**

To initialize a directory
```git init```

To clone a repository
```git clone --recursive url_V```

To add a new file to the repository - copy the new file in the repository and run command to index it
```git add filepath_V```

To commit means prepare the file locally to push to the repository
```git commit -m "Task #12345678 My head line_COMMENTS_V"```

Can be set before start of activities after ```"git init"```
May required to set username and email address
```git config --global user.name "Your Name_V"```
```git config --global user.email you@example.com_V```
After doing this, you may fix the identity used for this commit with:
```git commit --amend --reset-author```

To push the changes to the repository
```git push```
```git push --set-upstream origin master```   ''' Example below
```git push --set-upstream http://github.de/uids/GitPractice.git V master```

To list all branches
```git branch -r```
```git ls-remote --heads <remote-name>```
'''git checkout -t''' # To create and switch to a new branch locally

```git add filepath_V```

To commit means prepare the file locally to push to the repository
```git commit -m "My head line_COMMENTS_V" -m "My content line_COMMENTS_V."```

Can be set before start of activities after ```"git init"```
May required to set username and email address
```git config --global user.name "Your Name_V"```
```git config --global user.email you@example.com_V```
After doing this, you may fix the identity used for this commit with:
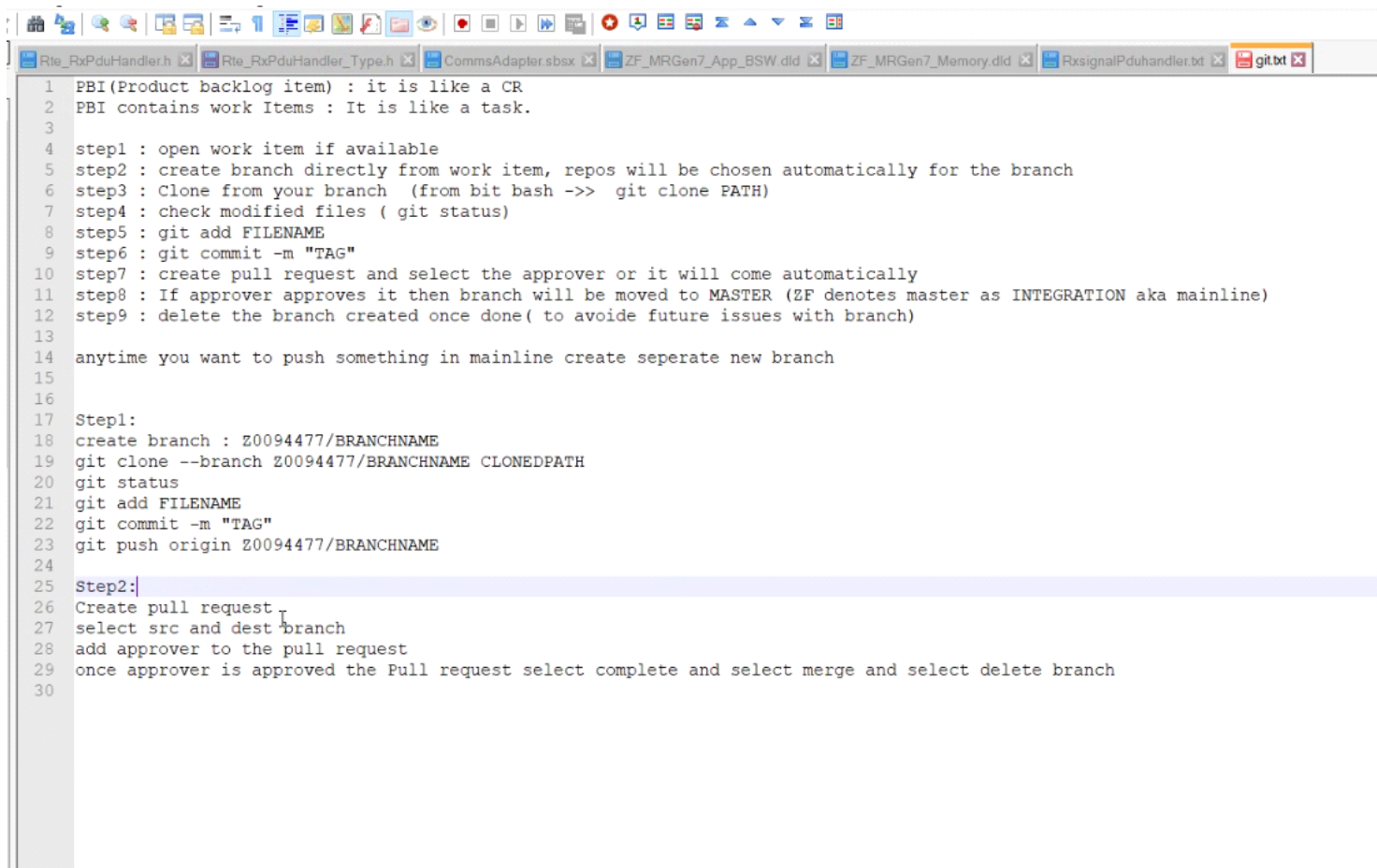```git commit --amend --reset-author```

To push the changes to the repository
```git push --set-upstream origin master```   ''' Example below
```git push --set-upstream http://github.de/uids/GitPractice.git_V master```

To list all branches
```git branch -r```
```git ls-remote --heads <remote-name>```
git checkout -t

```
 1   PBI(Product backlog item) : it is like a CR
 2   PBI contains work Items : It is like a task.
 3
 4   step1 : open work item if available
 5   step2 : create branch directly from work item, repos will be chosen automatically for the branch
 6   step3 : Clone from your branch  (from bit bash ->>  git clone PATH)
 7   step4 : check modified files ( git status)
 8   step5 : git add FILENAME
 9   step6 : git commit -m "TAG"
10   step7 : create pull request and select the approver or it will come automatically
11   step8 : If approver approves it then branch will be moved to MASTER (ZF denotes master as INTEGRATION aka mainline)
12   step9 : delete the branch created once done( to avoide future issues with branch)
13
14   anytime you want to push something in mainline create seperate new branch
15
16
17   Step1:
18   create branch : Z0094477/BRANCHNAME
19   git clone --branch Z0094477/BRANCHNAME CLONEDPATH
20   git status
21   git add FILENAME
22   git commit -m "TAG"
23   git push origin Z0094477/BRANCHNAME
24
25   Step2:
26   Create pull request
27   select src and dest branch
28   add approver to the pull request
29   once approver is approved the Pull request select complete and select merge and select delete branch
30
```

Git Branch

See What Branch You're On
- Run this command:
  - git status

List All Branches
NOTE: The current local branch will be marked with an asterisk (*).
- To see **local branches**, run this command:
  - git branch
- To see **remote branches**, run this command:
  - git branch -r
- To see **all local and remote branches**, run this command:
  - git branch -a

Create a New Branch
- Run this command (replacing **my-branch-name** with whatever name you want):
  - git checkout -b my-branch-name
- You're now ready to commit to this branch.

Switch to a Branch In Your Local Repo
- Run this command:
  - git checkout my-branch-name

Switch to a Branch That Came From a Remote Repo
1. ~~To get a list of all branches from the remote, run this command:~~
   - ~~git pull~~
2. Run this command to switch to the branch:
   - git checkout --track origin/my-branch-name

Push to a Branch
- If your local branch **does not exist** on the remote, run either of these commands:
  - git push -u origin my-branch-name
  - git push -u origin HEAD
  NOTE: HEAD is a reference to the top of the current branch, so it's an easy way to push to a branch of the same name on the remote. This saves you from having to type out the exact name of the branch!
- If your local branch **already exists** on the remote, run this command:
  - git push

Merge a Branch

1. You'll want to make sure your working tree is clean and see what branch you're on. Run this command:
 - git status
2. First, you must check out the branch that you want to merge another branch into (changes will be merged into this branch). If you're not already on the desired branch, run this command:
 - git checkout master
 - **NOTE:** Replace **master** with another branch name as needed.
3. Now you can merge another branch into the current branch. Run this command:
 - git merge my-branch-name
 - **NOTE:** When you merge, there may be a conflict. Refer to **Handling Merge Conflicts** (the next exercise) to learn what to do.

Delete Branches

- To delete a **remote branch**, run this command:
- git push origin --delete my-branch-name
- To delete a **local branch**, run either of these commands:
- git branch -d my-branch-name
- git branch -D my-branch-name
- **NOTE:** The -d option only deletes the branch if it has already been merged. The -D option is a shortcut for --delete --force, which deletes the branch irrespective of its merged status.

From <https://www.nobledesktop.com/learn/git/git-branches>

Checkout Remote Branch on Git

In some cases, you may be interested in checking out remote branches from your distant repository.

In order to switch to a remote branch, make sure to fetch your remote branch with "git fetch" first. You can then switch to it by executing "git checkout" with the "-t" option and the name of the branch.

```
$ git fetch
$ git checkout -t <remote_name>/<branch_name>
```

The "**-t**" option in checkout stands for "**track**" and it is used to create your branch and setting up the upstream branch automatically to the remote branch.

As an example, let's say that you have a branch named "remote-branch" on the "origin" remote.

From <https://devconnected.com/how-to-switch-branch-on-git/>

git branch -should show all the local branches of your repo. The starred branch is your current branch.

If you want to retrieve only the name of the branch you are on, you can do:
git rev-parse --abbrev-ref HEAD
or with Git 2.22 and above:
git branch --show-current

From <https://stackoverflow.com/questions/6245570/how-to-get-the-current-branch-name-in-git>

Show difference between local and remote branches
git fetch origin
git diff --name-only master origin/master
git diff master origin/master

From <https://coderwall.com/p/bey19w/git-show-difference-between-local-and-remote-branches>

git branch -vv

To list all branches
git branch -a
git branch -a --list *DIAG*   -> which contains DIAG in its name
git for-each-ref --format='%(authorname) %(refname:short)' refs/heads | grep "Z0145624"

To difference between two branches
git diff <mainbranch_path> <remotebranch_path>

To merge master into any other branch (dev_branch)
git checkout dev_branch
git reset --hard master
git push --force

From <https://superuser.com/questions/716818/how-to-overwrite-a-branch-in-git-with-master>

From <https://superuser.com/questions/716818/how-to-overwrite-a-branch-in-git-with-master>

how to add a repo to track an existing project
$ git remote add origin <repo-URL>
git remote add origin https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/app-r5
 git remote add origin https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/tools-dev

git remote -v

$ git add .
$ git commit -m "Initial commit"
$ git checkout -b Z0145624/Diag_Test_Branch_4K_RTE_3   (Create a new local branch)

git push --set-upstream <remote-name> <local-branch-name>
git push --set-upstream origin Z0145624/Diag_Test_Branch_4K_RTE_3 - worked (Created a remote branch in integration/origin)

Git Pull

git pull

[git remote](#) [git fetch](#) [git push](#) [git pull](#)
The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows. The git pull command is actually a combination of two other commands, [git fetch](#) followed by [git merge](#). In the first stage of operation git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be-created and HEAD updated to point at the new commit.

Git pull usage
How it works
The git pull command first runs git fetch which downloads content from the specified remote repository. Then a git merge is executed to merge the remote content refs and heads into a new local merge commit. To better demonstrate the pull and merging process let us consider the following example. Assume we have a repository with a main branch and a remote origin.

In this scenario, git pull will download all the changes from the point where the local and main diverged. In this example, that point is E. git pull will fetch the diverged remote commits which are A-B-C. The pull process will then create a new local merge commit containing the content of the new diverged remote commits.

In the above diagram, we can see the new commit H. This commit is a new merge commit that contains the contents of remote A-B-C commits and has a combined log message. This example is one of a few git pull merging strategies. A --rebase option can be passed to git pull to use a rebase merging strategy instead of a merge commit. The next example will demonstrate how a rebase pull works. Assume that we are at a starting point of our first diagram, and we have executed git pull --rebase.
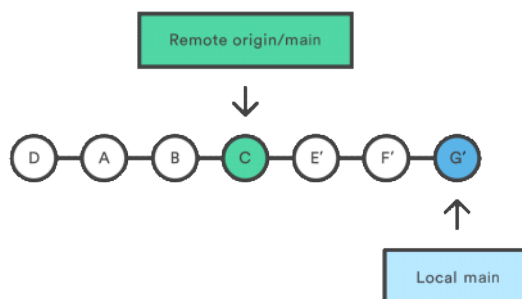
In this diagram, we can now see that a rebase pull does not create the new H commit. Instead, the rebase has copied the remote commits A--B--C and rewritten the local commits E--F--G to appear after them them in the local origin/main commit history.

Common Options
`git pull <remote>`
Fetch the specified remote's copy of the current branch and immediately merge it into the local copy. This is the same as git fetch <remote> followed by git merge origin/<current-branch>.

`git pull --nc-commit <remote>`
Similar to the default invocation, fetches the remote content but does not create a new merge commit.

`git pull --rebase <remote>`
Same as the previous pull Instead of using git merge to integrate the remote branch with the local one, use git rebase.

`git pull --verbose`
Gives verbose output during a pull which displays the content being downloaded and the merge details.

Git pull discussion

You can think of git pull as Git's version of svn update. It's an easy way to synchronize your local repository with upstream changes. The following diagram explains each step of the pulling process.

You start out thinking your repository is synchronized, but then git fetch reveals that origin's version of main has progressed since you last checked it. Then git merge immediately integrates the remote main into the local one.

Git pull and syncing

git pull is one of many commands that claim the responsibility of 'syncing' remote content. The git remote command is used to specify what remote endpoints the syncing commands will operate on. The git push command is used to upload content to a remote repository.

The git fetch command can be confused with git pull. They are both used to download remote content. An important safety distinction can be made between git pull and get fetch. git fetch can be considered the "safe" option whereas, git pull can be considered unsafe. git fetch will download the remote content and not alter the state of the local repository. Alternatively, git pull will download remote content and immediately attempt to change the local state to match that content. This may unintentionally cause the local repository to get in a conflicted state.

Pulling via Rebase

The --rebase option can be used to ensure a linear history by preventing unnecessary merge commits. Many developers prefer rebasing over merging, since it's like saying, "I want to put my changes on top of what everybody else has done." In this sense, using git pull with the --rebase flag is even more like svn update than a plain git pull.

In fact, pulling with --rebase is such a common workflow that there is a dedicated configuration option for it:

`git config --global branch.autosetuprebase always`
After running that command, all git pull commands will integrate via git rebase instead of git merge.

Git Pull Examples

The following examples demonstrate how to use git pull in common scenarios:

Default Behavior
`git pull`
Executing the default invocation of git pull will is equivalent to git fetch origin HEAD and git merge HEAD where HEAD is ref pointing to the current branch.

Git pull on remotes
`git checkout new_feature`
`git pull <remote repo>`

This example first performs a checkout and switches to the branch. Following that, the git pull is executed with being passed. This will implicitly pull down the newfeature branch from . Once the download is complete it will initiate a git merge.

Git pull rebase instead of merge

The following example demonstrates how to synchronize with the central repository's main branch using a rebase:

```
git checkout main
git pull --rebase origin
```

This simply moves your local changes onto the top of what everybody else has already contributed.

From <https://www.atlassian.com/git/tutorials/syncing/git-pull>

VIM Command

Wednesday, June 29, 2022   5:31 PM

Vim has two modes.
1. Insert mode (Where you can just type like normal text editor. Press i for insert mode)
2. Command mode (Where you give commands to the editor to get things done . Press ESC for command mode)
Most of them below are in command mode

- x - to delete the unwanted character
- u - to undo the last the command and U to undo the whole line
- CTRL-R to redo
- A - to append text at the end
- :wq - to save and exit
- :q! - to trash all changes
- dw - move the cursor to the beginning of the word to delete that word
- 2w - to move the cursor two words forward.
- 3e - to move the cursor to the end of the third word forward.
- 0 (zero) to move to the start of the line.
- d2w - which deletes 2 words .. number can be changed for deleting the number of consecutive words like d3w
- dd to delete the line and 2dd to delete to line .number can be changed for deleting the number of consecutive words

The format for a change command is: operator [number] motion
-operator - is what to do, such as d for delete
- [number] - is an optional count to repeat the motion
- motion - moves over the text to operate on, such as w (word),
$ (to the end of line), etc.

- p - puts the previously deleted text after the cursor(Type dd to delete the line and store it in a Vim register. and p to put the line)
- r - to replace the letter e.g press re to replace the letter with e
- ce - to change until the end of a word (place the cursor on the u in lubw it will delete ubw)
- ce - deletes the word and places you in Insert mode
- G - to move you to the bottom of the file.
- gg - to move you to the start of the file.
  Type the number of the line you were on and then G
- % to find a matching),], or }
- :s/old/new/g to substitute 'new' for 'old' where g is globally
- / backward search n to find the next occurrence and N to search in opposite direction
- ? forward search
- :! to run the shell commands like :!dir, :!ls
- :w - TEST (where TEST is the filename you chose.) . Save the file
- v - starts visual mode for selecting the lines and you can perform operation on that like d delete
- :r - Filename will insert the content into the current file
- R - to replace more than one character
- y - operator to copy text using v visual mode and p to paste it
- yw - (copy)yanks one word
- o - opens a line below the cursor and start Insert mode.
- O - opens a line above the cursor.
- a - inserts text after the cursor.
- A - inserts text after the end of the line.
- e - command moves to the end of a word.
- y - operator yanks (copies) text, p puts (pastes) it.
- R - enters Replace mode until <ESC> is pressed.
- ctrl-w to jump from one window to another

type a command :e and press ctrl+D to list all the command name starts with :e and press tab to complete the command

From <https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started>

Extra Git Command

Wednesday, June 29, 2022   6:14 PM

To list updated files of a commit
git diff-tree --no-commit-id --name-only -r 7c4307b81f161f7093926efb9ba7278d11f86810

To list all branches
git branch -a

To difference between two branches
git diff <mainbranch_path> <remotebranch_path>

To apply the remote repo's configuration to your local submodule repos.
git submodule sync
git submodule update

git submodule update updates the *contents* of the submodules. It is effectively running a "git fetch" and "git checkout" in each of your submodules.
git submodule sync updates the *metadata* about a submodule to reflect changes in the submodule URL. It re-synchronizes the information in .git/config with the information in .gitmodules.

BeyondCompare

Wednesday, June 29, 2022   6:29 PM

GIT FOR WINDOWS
BC3 logo BC version 3 or 4
Diff
At a Windows command prompt, enter the commands:
  git config --global diff.tool bc
  git config --global difftool.bc.path "C:\Program Files\Beyond Compare 4\BCompare.exe"
Note: For Git versions older than 2.2 (git --version) replace "bc" with "bc3" in the above instructions.

3-way Merge Pro only
At a Windows command prompt, enter the commands:
  git config --global merge.tool bc
  git config --global mergetool.bc.path "c:/Program Files/Beyond Compare 4/bcomp.exe"
Note: For Git versions older than 2.2.0 (git --version) replace "bc" with "bc3" in the above instructions.

Launching Diffs and Merges
File Diff:
  git difftool filename.ext

Folder Diff:
  git difftool --dir-diff

3-way Merge:
  git mergetool filename.txt

Advanced Settings
To disable the "Launch 'bc3' [Y/n]?" prompt, run the command:
  git config --global difftool.prompt false

Git's default settings retain merge files with *.orig extensions after a successful merge. To disable this
safety feature and automatically delete *.orig files after a merge, run the command:
  git config --global mergetool.keepBackup false

Git difftool ThirdParty/Vector/Dynamic/Application/GenData/Rte_Hook.h

Add & Commit

Thursday, June 30, 2022   10:12 AM

To list all commit
```
git log
git fetch --depth=<number-of-commits>
```

From <https://stackoverflow.com/questions/38191107/how-can-i-show-more-than-the-last-3-commits>
From <https://stackoverflow.com/questions/1710894/using-git-show-all-commits-that-are-in-one-branch-but-not-the-others>

```
git log remotename/branchname
```
Will display the log of a given remote branch in that repository, but only the logs that you have "fetched" from their repository to your personal "copy" of the remote repository.

```
git log HEAD..remote/branch
```
which will show you the commits that are in the remote branch, but not in your current branch
From <https://stackoverflow.com/questions/13941976/commit-history-on-remote-repository>

To list updated files of a commit
```
git diff-tree --no-commit-id --name-only -r
d0844e1bbc0f4c647bd15435a81218486b00c45f
git diff-tree --no-commit-id --name-only -r
bb80cdf30383c3b63e3104cf81a6a813a685da88
```

```
git show d0844e1bbc0f4c647bd15435a81218486b00c45f--name-only
```

From <https://mkyong.com/git/git-how-to-list-committed-files-that-are-going-to-push/>

To add all files except few
```
git add --all -- :!path/to/file1 :!path/to/file2 :!path/to/folder1/*
git add --all -- :!.cproject :!.project :!.settings/*
git add -A ':!<file_path>'
git add -A ':!ThirdParty/Vector/Dynamic/MRGen7_awr29xx.dpa'    - Worked
```

To add all files except untracked
```
git add -u
```

To update commit message
```
```git commit --amend -m "New commit message"```
```

To reset last commit
```
git reset --soft HEAD~
```

To add muti line commit message
```
git commit -m '1. line1
git 2. Line 2
3. line 3'
```

Undo In Git

Thursday, June 30, 2022   12:18 PM

```
#### TO undo - git add
git reset filename.txt

#### To undo - git commit message
```git commit --amend```
```git commit --amend -m "New commit message"```
```

[On undoing, fixing, or removing commits in git]
(https://sethrobertson.github.io/GitFixUm/fixup.html#badrebase)

Reset

Monday, August 29, 2022   3:59 PM

[git checkout](#) [git clean](#) [git revert](#) [git reset](#) [git rm](#)

Can I git stash only one file?

In some cases, you may want to stash a specific file in order to retrieve it later on. To stash a specific file, **use the "git stash push" command and specify the file you want to stash**. However, the other tracked files that may be modified in your current working directory are untouched

```
git stash push pathTo/MyFile

#### TO undo - git add
git reset filename.txt

#### To undo - git commit message
```git commit --amend```
```git commit --amend -m "New commit message"```
```

[On undoing, fixing, or removing commits in git]
(https://sethrobertson.github.io/GitFixUm/fixup.html#badrebase)

```
### If git pull cause - Merge conflicts
    # Mostly due to a local commit in main/branch
```
git reset --hard 620f888de5c7cb18137f98d1868fddce88b8a460 (any older valid commit)

```
#### TO hard reset a file
git restore pathTo/MyFile
git restore -s master~2 pathTo/MyFile
```

BASH & CMD

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.
A key preceded by a caret indicates the Ctrl key; thus ^K is ctrl-K.

h H   Display this help.
q :q Q :Q ZZ   **Exit**.

MOVING

e ^E j ^N CR * Forward one line   (or N lines).
y ^Y k ^K ^P * Backward one line   (or N lines).
f ^F ^V SPACE * Forward one window (or N lines).
b ^B ESC-v   * Backward one window (or N lines).
z   * Forward one window (and set window to N).
w   * Backward one window (and set window to N).
ESC-SPACE   * Forward one window, but don't stop at end-of-file.
d ^D   * Forward one half-window (and set half-window to N).
u ^U   * Backward one half-window (and set half-window to N).
ESC-) RightArrow * Right one half screen width (or N positions).
ESC-(LeftArrow * Left one half screen width (or N positions).
ESC-} ^RightArrow   Right to last column displayed.
ESC-{ ^LeftArrow   Left to first column.
F   Forward forever; like "tail -f".
ESC-F   Like F but stop when search pattern is found.
r ^R ^L   Repaint screen.
R   Repaint screen, discarding buffered input.

    Default "window" is the screen height.
    Default "half-window" is half of the screen height.

SEARCHING

/pattern   * Search forward for (N-th) matching line.
?pattern   * Search backward for (N-th) matching line.
n   * Repeat previous search (for N-th occurrence).
N   * Repeat previous search in reverse direction.
ESC-n   * Repeat previous search, spanning files.
ESC-N   * Repeat previous search, reverse dir. & spanning files.
ESC-u   Undo (toggle) search highlighting.
ESC-U   Clear search highlighting.
&pattern   * Display only matching lines.

    A search pattern may begin with one or more of:
    ^N or ! Search for NON-matching lines.
    ^E or * Search multiple files (pass thru END OF FILE).
    ^F or @ Start search at FIRST file (for /) or last file (for ?).
    ^K   Highlight matches, but don't move (KEEP position).
    ^R   Don't use REGULAR EXPRESSIONS.
    ^W   WRAP search if no match found.

JUMPING
HELP -- Press RETURN for more, or q when done

```
ZF-WORLD+Z0145624@HDRC01548 MINGW64 /d/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5 (integration)
$ cmd
Microsoft Windows [Version 10.0.19042.1766]
(c) Microsoft Corporation. All rights reserved.

D:\Sandbox\GIT_Sandbox\MRGen7_Parts\app-r5>ADAS-Build.exe
ADAS-Build.exe
INFO:11:53:24 Initialization done, starting process.
INFO:11:53:24 Starting native build...
INFO:11:53:25 [INFO] Using Compiler: D:/Sandbox/MRGen7/20_Radar/50_Tools/Software/SWConstructionTools/ARM_TI_compiler
INFO:11:53:25 [INFO] Using C PreProcessor: D:/Sandbox/MRGen7/20_Radar/50_Tools/Software/SWConstructionTools/cygwin_root/cmd
INFO:11:53:26 -- Toolchain file: D:\Sandbox\GIT_Sandbox\MRGen7_Parts\app-r5\ZZZ_Tool_Configuration\CmakeModules\TC_LTS.cmake
INFO:11:53:26 NVM Include: D:/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5/ThirdParty/Vector/Static/Crc/Implementation
INFO:11:53:26 NVM Include: D:/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5/ThirdParty/Vector/Static/MemIf/Implementation
INFO:11:53:27 -- ***************************************** WARNING *****************************************
INFO:11:53:27 -- No Post_Build_Action found under D:/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5/../tools-config/Post_Build_Actions
INFO:11:53:27 -- *********************************************************************************************
INFO:11:53:27 -- Configuring done
INFO:11:53:29 -- Generating done
INFO:11:53:29 Generate graphviz: D:/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5/ZZZ_Intermediates/graphviz/MRGen7.dot
INFO:11:53:30 -- Build files have been written to: D:/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5/ZZZ_Intermediates
INFO:11:53:30 ninja: Entering directory `D:\Sandbox\GIT_Sandbox\MRGen7_Parts\app-r5\ZZZ_Intermediates'
INFO:11:54:00 [1/9] Generating Dependency Image files ...
INFO:11:55:51 [2/9] Preprocessing Os_Hal_EntryAsm_Lcfg.asm using D:/Sandbox/MRGen7/20_Radar/50_Tools/Software/SWConstructionTools/cygwin_root/cmd/clang.exe
INFO:11:55:51 [3/9] Preprocessing Os_Hal_ContextAsm.asm using D:/Sandbox/MRGen7/20_Radar/50_Tools/Software/SWConstructionTools/cygwin_root/cmd/clang.exe
INFO:11:55:53 [4/9] Building ASM object ThirdParty\Vector\Static\Os\CMakeFiles\ar_os.dir\Implementation\Os_Hal_ContextAsm_preprocessed.asm.obj
INFO:11:55:53 [5/9] Building ASM object ThirdParty\Vector\Dynamic\CMakeFiles\ar_os_cfg.dir\Application\GenData\Os_Hal_EntryAsm_Lcfg_preprocessed.asm.obj
INFO:11:55:55 [6/9] Building C object CMakeFiles\MRGen7_BSW.dir\bsw_version.c.obj
INFO:11:55:56 [7/9] Linking C static library ThirdParty\Vector\Static\Os\libar_os.a
INFO:11:55:57 [8/9] Linking C static library ThirdParty\Vector\Dynamic\libar_os_cfg.a
INFO:11:56:07 [9/9] Linking C executable MRGen7_BSW
INFO:11:56:07 <Linking>
INFO:11:56:07


#############################
# Process ended with SUCCESS
#############################
# CompilerErrors:     0
# CompilerWarnings:   0
# CompilerInfo:       0
# LinkerErrors:       0
# LinkerWarnings      0
# LinkerInfo          0
#############################
# Start Time : 2022-07-25 11:53:21.861422
# End Time   : 2022-07-25 11:56:07.473041
# -------------------------
# Total Time : 0:02:45.611619
#############################

D:\Sandbox\GIT_Sandbox\MRGen7_Parts\app-r5>exit
exit

ZF-WORLD+Z0145624@HDRC01548 MINGW64 /d/Sandbox/GIT_Sandbox/MRGen7_Parts/app-r5 (integration)
$ |
```

Links

Thursday, July 7, 2022   10:29 AM

Remote

Thursday, July 7, 2022   4:08 PM

 git remote show origin
git remote -v

```
ture-app/020_OilrModel (master)
$ git remote show origin
* remote origin
  Fetch URL: https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/architecture-app
  Push  URL: https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/architecture-app
  HEAD branch: master
  Remote branches:
    master                              tracked
    refs/remotes/origin/z0011803/Test_LFS stale (use 'git remote prune' to remove)
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

Git remove Untracked files

Git remove Untracked files
git clean -n

From <https://koukia.ca/how-to-remove-local-untracked-files-from-the-current-git-branch-571c6ce9b6b1>

git clean -f
git clean -fd

From <https://koukia.ca/how-to-remove-local-untracked-files-from-the-current-git-branch-571c6ce9b6b1>

git restore .

From <https://stackoverflow.com/questions/52704/how-do-i-discard-unstaged-changes-in-git>

Git Stash

Tuesday, September 6, 2022   11:37 AM

[A practical guide to using the git stash command | Opensource.com](#)

Can I git stash only one file?

In some cases, you may want to stash a specific file in order to retrieve it later on.
To stash a specific file, **use the "git stash push" command and specify the file you want to stash**.
However, the other tracked files that may be modified in your current working directory are untouched

git stash list
$ **git stash list**
stash@{0}: WIP on Task_191591_SigProcDiagnostics_Implementation: ed7a8ce5 Task #191591 SigProcDiagnostic Implementation
stash@{1}: WIP on integration: acb0a764 Merged PR 16633: Task #181225 Datalog_status signal implemented
stash@{2}: WIP on Diag_Test_Branch: 1d77ad78 Task #165482 DIAG - Dummy implementation for feasibility study - 1400 Bytes of DID
stash@{3}: WIP on Diag_Test_Branch: 0d6573a3 Task #165482 DIAG - Dummy implementation for feasibility study - 2000 Byte of DID
stash@{4}: WIP on Task_163501_PM_Provide_Battery_Interface: 393adcd Task_163501 PM Providing interface for Vbat

git stash pop stash@{2}

From <https://www.atlassian.com/git/tutorials/saving-changes/git-stash#re-applying-your-stashed-changes>

Rebase
git rebase --abort
git rebase --skip

git remote prune origin

Thursday, September 15, 2022   3:03 PM

No git remote prune origin will only delete the refs to remote branches that no longer exist.

From <https://www.google.com/search?q=git+remote+prune+origin&oq=git+remote+prune+origin&aqs=edge..69i57j0i512l7j69i64.786j0j1&sourceid=chrome&ie=UTF-8>

git remote prune origin

From <https://stackoverflow.com/questions/6656619/git-and-nasty-error-cannot-lock-existing-info-refs-fatal>

Rest one staged file
git reset HEAD -- <file>

From <https://stackoverflow.com/questions/1505948/how-do-i-remove-a-single-file-from-the-staging-area-undo-git-add>

Try cleaning-up your local repository with:

1228

```
$ git gc --prune=now
$ git remote prune origin
```

man git-gc(1):

```
git-gc - Cleanup unnecessary files and optimize the local repository

git gc [--aggressive] [--auto] [--quiet] [--prune=<date> | --no-prune]

        Runs a number of housekeeping tasks within the current repository, such as compres
        (to reduce disk space and increase performance) and removing unreachable objects wl
        created from prior invocations of git add.

        Users are encouraged to run this task on a regular basis within each repository to
        space utilization and good operating performance.
```

Track an existing project

git config --global lfs.https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/architecture-app.git/info/lfs.locksverify true

From <https://dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_wiki/wikis/AZP-095_DivU_Gen7RadarCore.wiki/9075/Rhapsody-architecture-git-workflow>

[Rhapsody architecture git workflow - Overview (azure.com)](#)

git branch --set-upstream local-branch-name origin/remote-branch-name

how to add a repo to track an existing project
$ git remote add origin <repo-URL>
git remote add origin https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/app-r5
 git remote add origin https://SW4ZF@dev.azure.com/SW4ZF/AZP-095_DivU_Gen7RadarCore/_git/tools-dev

git remote -v

$ git add .
$ git commit -m "Initial commit"
$ git checkout -b Z0145624/Diag_Test_Branch_4K_RTE_3   (Create a new local branch)

git push --set-upstream <remote-name> <local-branch-name>
git push --set-upstream origin Z0145624/Diag_Test_Branch_4K_RTE_3 - worked (Created a remote branch in integration/origin)

git push <remote-name> <local-branch-name>:<remote-branch-name>
git push --set-upstream integration Z0145624/Diag_Test_Branch_4K_RTE_3:remotes/origin/Z0145624/Diag_Test_Branch_4K_Rte_2

Cherry-pick

git-cherry-pick - Apply the changes introduced by some existing commits

From <https://git-scm.com/docs/git-cherry-pick>

git cherry-pick

From <https://git-scm.com/docs/git-cherry-pick>

git diff-tree --no-commit-id --name-only -r bb80cdf30383c3b63e3104cf81a6a813a685da88

git cherry-pick bb80cdf30383c3b63e3104cf81a6a813a685da88

From <https://mattstauffer.com/blog/how-to-merge-only-specific-commits-from-a-pull-request/>