# Microsoft CMD Notes

The command prompt is a text based utility that allows tasks usually accomplished via an operating system GUI through text commands.

Some common commands are moving or deleting files, configuring settings, killing tasks, and even git related source control actions.

## System Commands

### DIR

The dir command specifies the list of files and subdirectories in the current directory.

### CD

Cd (short for change directory), changes the current operating directory to the path specified, and any other directory specific commands will be executed on the new directory.   
If “cd..” is used, it will move up one level in the current path (i.e. C:\OceanLiners\Titanic, will change to C:\OceanLiners)

MKDIR

The MKDIR command makes the specified directory in the current directory.

I.e. if the current working directory is C:/Projects typing MKDIR Ships/Titanic will create the subdirectory ship under Projects, with a folder named “Titanic” inside of it.

You can then navigate to the new folder using CD.

### Del

The del command is used to delete files from the current working directory. It’s behavior is rather limited by itself, but can easily be enhanced using various flags.

|  |  |
| --- | --- |
| Flag | Explanation |
| /F | Forces the deletion of any read-only files. These would normally be skipped. |
| /Q | Runs the command in quiet mode. In regular mode, a prompt would be given for every file found. |
| /S | This will encapsulate all files in the delete operation. Any folder found under the current directory will have it’s file removed. |
| \*.\* | Acts as a filter to specify which files or types of files to delete. In this example, all files are deleted. However, this can be amended to limit it to specific files types (\*.txt to delete only text files), or (Titanic.\* to delete files of any type named “Titanic”) |
| >Nul | This disables console output messages regarding the delete operations. This can substantially speed up the process. |

Assuming the mkdir command example was run earlier, running DEL /F /Q /S\*.\*>NUL would delete any and all files in the Ships and Titanic folders without prompting, and with console output disabled.   
  
Del will NOT delete files and should be paired with the RMDIR command if folders need to be removed.

Source: <https://www.ghacks.net/2017/07/18/how-to-delete-large-folders-in-windows-super-fast/>

### RMDIR

RMDIR does for folders what DEL does for files: deletes them. Like DEL, RMDIR can be used with flags to customize and enhance its functionality. RMDIR will only work on empty folders and so should be paired with the DEL Command.

|  |  |
| --- | --- |
| FLAG | Explanation |
| /Q | Runs in quiet mode, disabling prompts to delete folders. |
| /S | Automatically runs the operations on all folders in the path, deleting any empty directories. |
| Foldername | This is an optional parameter that specifies a path to a folder to be deleted |

Source: <https://www.ghacks.net/2017/07/18/how-to-delete-large-folders-in-windows-super-fast/>

### Move

Move is used to take a file from one location, and place it another so that it lives in the new location.

Move can be used from any directory to move between any two directories. However, the full path of the directory will need to be specified if it lives outside of the current working directory. If it lives in the current directory, or a subdirectory, the source can be given as only the filename, or the first subdirectory.

The destination will always be specified in the same manner.

Filenames with spaces will need to be enclosed in quotation marks. Wildcards can also be applied to move many files at once.

Move source destination

Move C:\TestRepos\test.txt C:\Test will move the file test.txt from the TestRepos folder to the test folder.

Move “This is a test.txt” C:\Test\SomeFolder

### Copy

Copy functions very similarly to the move command except that the file will exist in both locations instead of leaving the source folder. Otherwise the same rules apply.

The source directory must be fully specified unless it is the current working directory or a sub-directory there of.

The destination must be fully specified unless it is the current working directory or a sub-directory there of, otherwise only the filename, or the first sub-directory can be specified.

Filenames with spaces must be encased in “”.

Copy source destination

Copy C:\TestFolder\Test.txt C:\ProdFolder

Copy “This is a test.txt” C:\Thisis\ATest\Folder

## GIT Commands

All commands are prefaced with GIT

### Init

This will initiate an empty git folder in a directory allowing it to be used as a git repository locally and synched with a remote repository.

To uninitiate a repository, simply navigate to it and remove the git directory.

### Add

Normally followed by a file name(s), this commands adds a file to git tracking which must be done before it can be committed.

Multiple filenames can be specified by separating them with a comma. File names containing spaces must be enclosed in “”.

### Git add test.txt, “This is a test.txt” will both those files to be tracked by git. Add all files

If there are multiple new files in the repo, (Or this is the first commit), all new files can be added with .   
  
Git add . will add all new files to the commit. And allow commits to be made.

### Git Stage

Stage will add existing files that have been changed since the last commit to your dog.

Git stage filename will track the new change of the specified file names.

Git stage . will add open files.

Commit   
This command “stages” files for later integration in a remote repository. Any new files must be added via git add before they can be committed.

Each commit requires a message. Typing commit without a message will open an editor where a message can be entered.

The “-m” flag will allow a message to be specified after it in “”.

Git Commit –m “Adding Test Files”

### Git Push

Push will send the latest commit and any changes to the remote repository and branch specified in the push command.

Git push origin master

That command will push any staged changes to the master branch of the origin.

Push has some built in security checks which will refuse to commit if the current commit is not part of the repository. It is possible to override this using the -f command, but this may cause the existing repository to be overwritten and any previous commits to be lost. ONLY USE THIS WHEN ABSOLUTELY SURE.

### Git Set UpStream

Git –set-upstream is useful when you have a new local branch (created with the git branch command) that doesn’t have a corresponding remote branch. If the current local branch does not have a remote branch with the same name, it cannot be committed.

By appending –set-upstream to a push command, a new branch with a matching name will be created, allowing the changes to be added and tracked in the origin.

It uses the syntax Git push –set-upstream origin BranchName

Git –set-upstream origin NewFeatureTest, will create a new branch in the origin (remote) repository called “NewFeatureTest” and push your changes there.   
  
These changes can be merged into the master branch when development and testing is complete.

## GIT PULL

Pull will create a copy of the remote branch in the current, initialized directory. The remote and branch are specified within the command.   
  
To avoid issues with pushing to a branch, a git pull should always be used with a new repository before pushing to a remote

Git pull origin master

### Remote

This command is used to specify actions relating to the remote source of a local repository. It is typically followed by one or more additional commands/flags to specify and describe the action.

Git remote add

The above command will add the specified repository as the origin for the current local repository.

There are a number of ways to specify a remote repository including HTTPS: and SSH.

To add a repository using HTTPS, type https:// followed by the username : [password@github.com](mailto:password@github.com) and the exact repository and sub repository afterward.

Git remote add origin <https://username:password@github.com/username/TestRepo> will login to git hub with username using the password “password” and if successful, set username/TestRepo as the origin. Username refers to the username of the user who owns the repository. In this case, it is our repository.

### Git Branch

Branch will create a new branch (version or offshoot) in your local repository. This is useful to maintain a “clean” production or current branch, and then a new branch for each feature, which can then be merged into the main branch when development and testing is complete.

Git branch name will create a new local branch of the name.   
  
Git branch “NewFeatureTest” will create a new branch of “NewFeatureTest in your local repo.

## Git Checkout

Git checkout switches between local branches.   
  
It uses the syntax “Git Checkout BranchName”

Git Checkout NewFeatureTest will switch from the current branch to New Feature Test and all changes will now be done on that branch until it is changed.