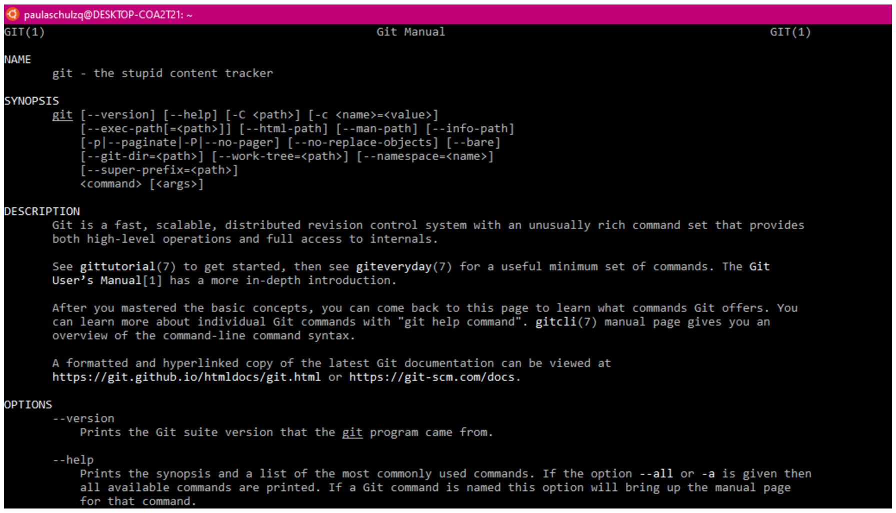
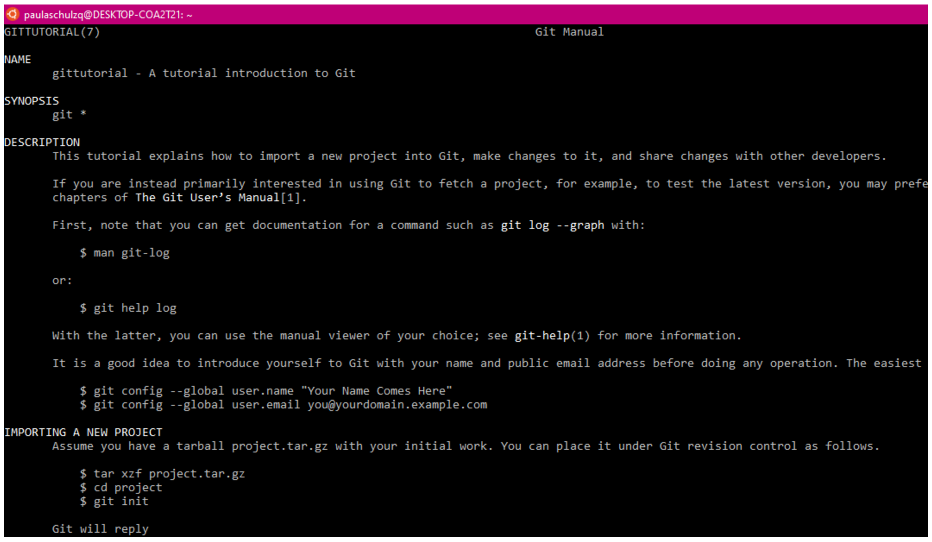
Tarea lee el manual

- ➔ Man git



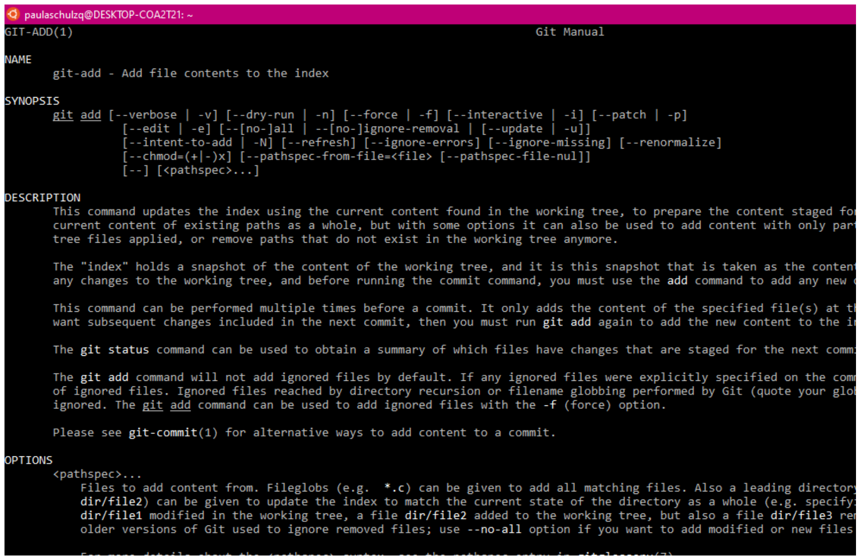
```
paulaschulzq@DESKTOP-COA2T21: ~
GIT(1)                                                      Git Manual                                                      GIT(1)

NAME
       git - the stupid content tracker

SYNOPSIS
       git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>]
           <command> [<args>]

DESCRIPTION
       Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides
       both high-level operations and full access to internals.

       See gittutorial(7) to get started, then see giteveryday(7) for a useful minimum set of commands. The Git
       User’s Manual[1] has a more in-depth introduction.

       After you mastered the basic concepts, you can come back to this page to learn what commands Git offers. You
       can learn more about individual Git commands with "git help command". gitcli(7) manual page gives you an
       overview of the command-line command syntax.

       A formatted and hyperlinked copy of the latest Git documentation can be viewed at
       https://git.github.io/htmldocs/git.html or https://git-scm.com/docs.

OPTIONS
       --version
           Prints the Git suite version that the git program came from.

       --help
           Prints the synopsis and a list of the most commonly used commands. If the option --all or -a is given then
           all available commands are printed. If a Git command is named this option will bring up the manual page
           for that command.
```

- ➔ Man gittutorial



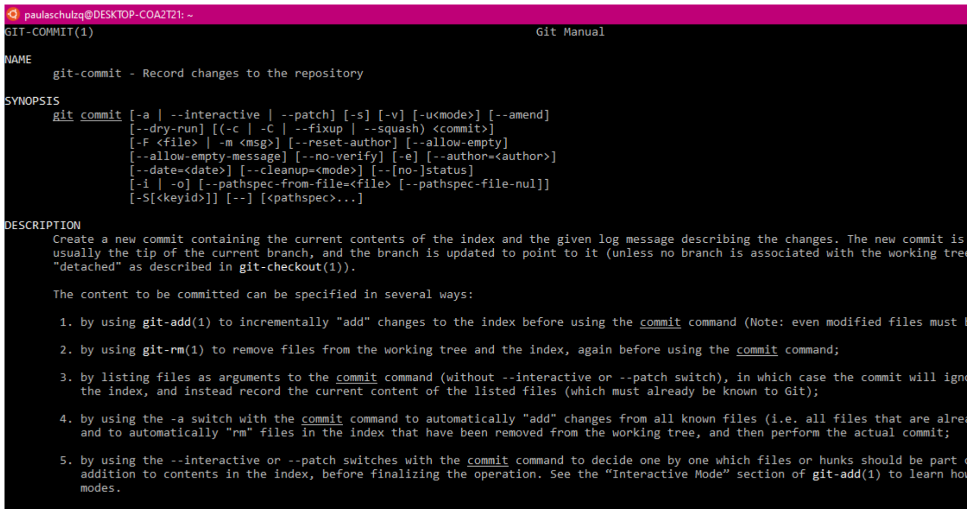
```
paulaschulzq@DESKTOP-COA2T21: ~
GITTUTORIAL(7)                                                             Git Manual

NAME
       gittutorial - A tutorial introduction to Git

SYNOPSIS
       git *

DESCRIPTION
       This tutorial explains how to import a new project into Git, make changes to it, and share changes with other developers.

       If you are instead primarily interested in using Git to fetch a project, for example, to test the latest version, you may prefe
       chapters of The Git User’s Manual[1].

       First, note that you can get documentation for a command such as git log --graph with:

           $ man git-log

       or:

           $ git help log

       With the latter, you can use the manual viewer of your choice; see git-help(1) for more information.

       It is a good idea to introduce yourself to Git with your name and public email address before doing any operation. The easiest

           $ git config --global user.name "Your Name Comes Here"
           $ git config --global user.email you@yourdomain.example.com

IMPORTING A NEW PROJECT
       Assume you have a tarball project.tar.gz with your initial work. You can place it under Git revision control as follows.

           $ tar xzf project.tar.gz
           $ cd project
           $ git init

       Git will reply
```

- ➔ Man git-add



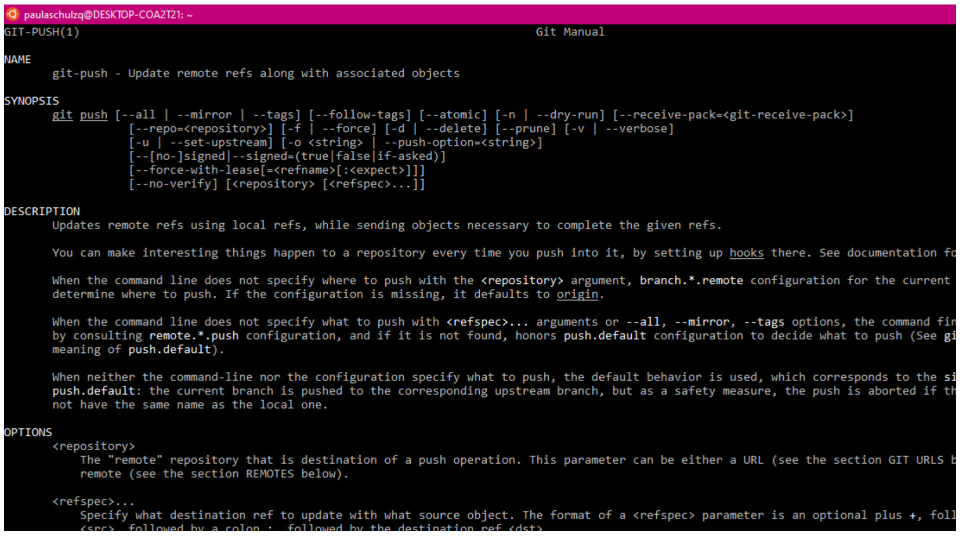
```
paulaschulzq@DESKTOP-COA2T21: ~
GIT-ADD(1)                                                                 Git Manual

NAME
       git-add - Add file contents to the index

SYNOPSIS
       git add [--verbose | -v] [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]
                 [--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]]
                 [--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing] [--renormalize]
                 [--chmod=(+|-)x] [--pathspec-from-file=<file> [--pathspec-file-nul]]
                 [--] [<pathspec>...]

DESCRIPTION
       This command updates the index using the current content found in the working tree, to prepare the content staged fo
       current content of existing paths as a whole, but with some options it can also be used to add content with only par
       tree files applied, or remove paths that do not exist in the working tree anymore.

       The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the conten
       any changes to the working tree, and before running the commit command, you must use the add command to add any new 

       This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at t
       want subsequent changes included in the next commit, then you must run git add again to add the new content to the i

       The git status command can be used to obtain a summary of which files have changes that are staged for the next comm

       The git add command will not add ignored files by default. If any ignored files were explicitly specified on the com
       of ignored files. Ignored files reached by directory recursion or filename globbing performed by Git (quote your glo
       ignored. The git add command can be used to add ignored files with the -f (force) option.

       Please see git-commit(1) for alternative ways to add content to a commit.

OPTIONS
       <pathspec>...
           Files to add content from. Fileglobs (e.g.  *.c) can be given to add all matching files. Also a leading director
           dir/file2) can be given to update the index to match the current state of the directory as a whole (e.g. specify
           dir/file1 modified in the working tree, a file dir/file2 added to the working tree, but also a file dir/file3 re
           older versions of Git used to ignore removed files; use --no-all option if you want to add modified or new files
```

➔ Man git-commit



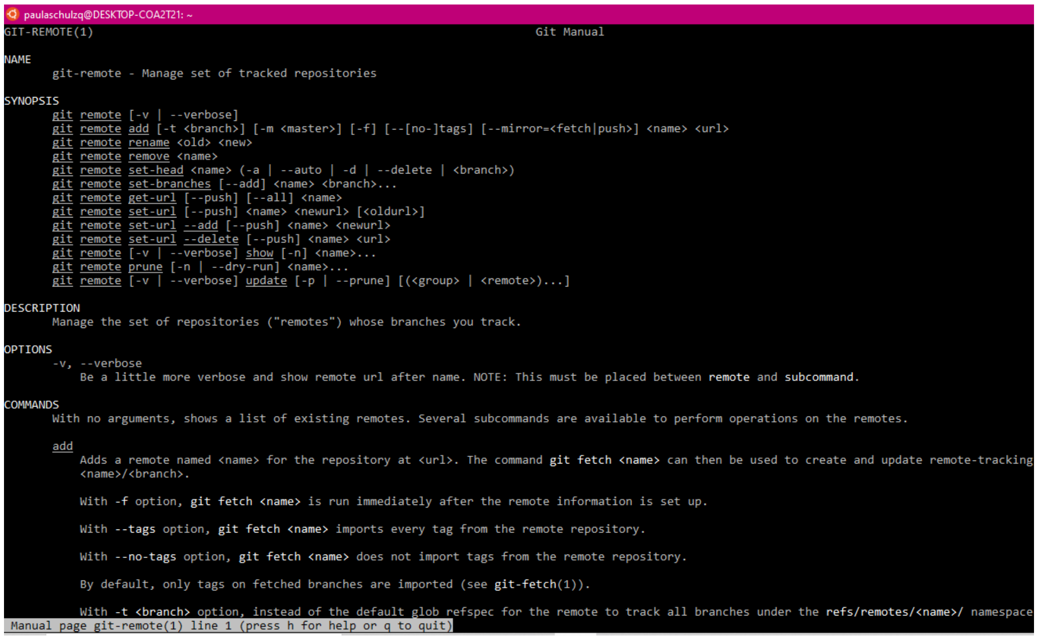
```
GIT-COMMIT(1)                                                                            Git Manual

NAME
       git-commit - Record changes to the repository

SYNOPSIS
       git commit [-a | --interactive | --patch] [-s] [-v] [-u<mode>] [--amend]
                  [--dry-run] [(-c | -C | --fixup | --squash) <commit>]
                  [-F <file> | -m <msg>] [--reset-author] [--allow-empty]
                  [--allow-empty-message] [--no-verify] [-e] [--author=<author>]
                  [--date=<date>] [--cleanup=<mode>] [--[no-]status]
                  [-i | -o] [--pathspec-from-file=<file> [--pathspec-file-nul]]
                  [-S[<keyid>]] [--] [<pathspec>...]

DESCRIPTION
       Create a new commit containing the current contents of the index and the given log message describing the changes. The new commit is
       usually the tip of the current branch, and the branch is updated to point to it (unless no branch is associated with the working tre
       "detached" as described in git-checkout(1)).

       The content to be committed can be specified in several ways:

        1. by using git-add(1) to incrementally "add" changes to the index before using the commit command (Note: even modified files must

        2. by using git-rm(1) to remove files from the working tree and the index, again before using the commit command;

        3. by listing files as arguments to the commit command (without --interactive or --patch switch), in which case the commit will ign
           the index, and instead record the current content of the listed files (which must already be known to Git);

        4. by using the -a switch with the commit command to automatically "add" changes from all known files (i.e. all files that are alre
           and to automatically "rm" files in the index that have been removed from the working tree, and then perform the actual commit;

        5. by using the --interactive or --patch switches with the commit command to decide one by one which files or hunks should be part
           addition to contents in the index, before finalizing the operation. See the "Interactive Mode" section of git-add(1) to learn ho
           modes.
```

➔ Man git-push



```
GIT-PUSH(1)                                                                              Git Manual

NAME
       git-push - Update remote refs along with associated objects

SYNOPSIS
       git push [--all | --mirror | --tags] [--follow-tags] [--atomic] [-n | --dry-run] [--receive-pack=<git-receive-pack>]
                  [--repo=<repository>] [-f | --force] [-d | --delete] [--prune] [-v | --verbose]
                  [-u | --set-upstream] [-o <string> | --push-option=<string>]
                  [--[no-]signed|--signed=(true|false|if-asked)]
                  [--force-with-lease[=<refname>[:<expect>]]]
                  [--no-verify] [<repository> [<refspec>...]]

DESCRIPTION
       Updates remote refs using local refs, while sending objects necessary to complete the given refs.

       You can make interesting things happen to a repository every time you push into it, by setting up hooks there. See documentation fo

       When the command line does not specify where to push with the <repository> argument, branch.*.remote configuration for the current
       determine where to push. If the configuration is missing, it defaults to origin.

       When the command line does not specify what to push with <refspec>... arguments or --all, --mirror, --tags options, the command fin
       by consulting remote.*.push configuration, and if it is not found, honors push.default configuration to decide what to push (See gi
       meaning of push.default).

       When neither the command-line nor the configuration specify what to push, the default behavior is used, which corresponds to the si
       push.default: the current branch is pushed to the corresponding upstream branch, but as a safety measure, the push is aborted if th
       not have the same name as the local one.

OPTIONS
       <repository>
           The "remote" repository that is destination of a push operation. This parameter can be either a URL (see the section GIT URLS b
           remote (see the section REMOTES below).

       <refspec>...
           Specify what destination ref to update with what source object. The format of a <refspec> parameter is an optional plus +, foll
           <src>, followed by a colon :, followed by the destination ref <dst>.
```

➔ Man git-remote



```
GIT-REMOTE(1)                                                                            Git Manual

NAME
       git-remote - Manage set of tracked repositories

SYNOPSIS
       git remote [-v | --verbose]
       git remote add [-t <branch>] [-m <master>] [-f] [--[no-]tags] [--mirror=<fetch|push>] <name> <url>
       git remote rename <old> <new>
       git remote remove <name>
       git remote set-head <name> (-a | --auto | -d | --delete | <branch>)
       git remote set-branches [--add] <name> <branch>...
       git remote get-url [--push] [--all] <name>
       git remote set-url [--push] <name> <newurl> [<oldurl>]
       git remote set-url --add [--push] <name> <newurl>
       git remote set-url --delete [--push] <name> <url>
       git remote [-v | --verbose] show [-n] <name>...
       git remote prune [-n | --dry-run] <name>...
       git remote [-v | --verbose] update [-p | --prune] [(<group> | <remote>)...]

DESCRIPTION
       Manage the set of repositories ("remotes") whose branches you track.

OPTIONS
       -v, --verbose
           Be a little more verbose and show remote url after name. NOTE: This must be placed between remote and subcommand.

COMMANDS
       With no arguments, shows a list of existing remotes. Several subcommands are available to perform operations on the remotes.

       add
           Adds a remote named <name> for the repository at <url>. The command git fetch <name> can then be used to create and update remote-tracking
           <name>/<branch>.

           With -f option, git fetch <name> is run immediately after the remote information is set up.

           With --tags option, git fetch <name> imports every tag from the remote repository.

           With --no-tags option, git fetch <name> does not import tags from the remote repository.

           By default, only tags on fetched branches are imported (see git-fetch(1)).

           With -t <branch> option, instead of the default glob refspec for the remote to track all branches under the refs/remotes/<name>/ namespace
 Manual page git-remote(1) line 1 (press h for help or q to quit)
```