student9@student9vdi MINGW64 ~

$ git

usage: git [-v | --version] [-h | --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>] [--config-env=<name>=<envvar>]

<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.

student9@student9vdi MINGW64 ~

$ git config --user.name"Raj"

error: unknown option `user.nameRaj'

usage: git config [<options>]

Config file location

--global use global config file

--system use system config file

--local use repository config file

--worktree use per-worktree config file

-f, --file <file> use given config file

--blob <blob-id> read config from given blob object

Action

--get get value: name [value-pattern]

--get-all get all values: key [value-pattern]

--get-regexp get values for regexp: name-regex [value-pattern]

--get-urlmatch get value specific for the URL: section[.var] URL

--replace-all replace all matching variables: name value [value-pattern]

--add add a new variable: name value

--unset remove a variable: name [value-pattern]

--unset-all remove all matches: name [value-pattern]

--rename-section rename section: old-name new-name

--remove-section remove a section: name

-l, --list list all

--fixed-value use string equality when comparing values to 'value-pattern'

-e, --edit open an editor

--get-color find the color configured: slot [default]

--get-colorbool find the color setting: slot [stdout-is-tty]

Type

-t, --type <type> value is given this type

--bool value is "true" or "false"

--int value is decimal number

--bool-or-int value is --bool or --int

--bool-or-str value is --bool or string

--path value is a path (file or directory name)

--expiry-date value is an expiry date

Other

-z, --null terminate values with NUL byte

--name-only show variable names only

--includes respect include directives on lookup

--show-origin show origin of config (file, standard input, blob, command line)

--show-scope show scope of config (worktree, local, global, system, command)

--default <value> with --get, use default value when missing entry

student9@student9vdi MINGW64 ~

$ git config --user.email"helloraj@gmail.com"

error: unknown option `user.emailhelloraj@gmail.com'

usage: git config [<options>]

Config file location

--global use global config file

--system use system config file

--local use repository config file

--worktree use per-worktree config file

-f, --file <file> use given config file

--blob <blob-id> read config from given blob object

Action

--get get value: name [value-pattern]

--get-all get all values: key [value-pattern]

--get-regexp get values for regexp: name-regex [value-pattern]

--get-urlmatch get value specific for the URL: section[.var] URL

--replace-all replace all matching variables: name value [value-pattern]

--add add a new variable: name value

--unset remove a variable: name [value-pattern]

--unset-all remove all matches: name [value-pattern]

--rename-section rename section: old-name new-name

--remove-section remove a section: name

-l, --list list all

--fixed-value use string equality when comparing values to 'value-pattern'

-e, --edit open an editor

--get-color find the color configured: slot [default]

--get-colorbool find the color setting: slot [stdout-is-tty]

Type

-t, --type <type> value is given this type

--bool value is "true" or "false"

--int value is decimal number

--bool-or-int value is --bool or --int

--bool-or-str value is --bool or string

--path value is a path (file or directory name)

--expiry-date value is an expiry date

Other

-z, --null terminate values with NUL byte

--name-only show variable names only

--includes respect include directives on lookup

--show-origin show origin of config (file, standard input, blob, command line)

--show-scope show scope of config (worktree, local, global, system, command)

--default <value> with --get, use default value when missing entry

student9@student9vdi MINGW64 ~

$ cd desktop

student9@student9vdi MINGW64 ~/desktop

$ mkdir ctlab

student9@student9vdi MINGW64 ~/desktop

$ touch ctpgml.html

student9@student9vdi MINGW64 ~/desktop

$ cd ctlab

student9@student9vdi MINGW64 ~/desktop/ctlab

$ touch ctpgml.txt

student9@student9vdi MINGW64 ~/desktop/ctlab

$ echo "version control tool"

version control tool

student9@student9vdi MINGW64 ~/desktop/ctlab

$ echo "hello wassssuuuppp"

hello wassssuuuppp

student9@student9vdi MINGW64 ~/desktop/ctlab

$ echo "hello wasssuuuppp">ctpgml.txt

student9@student9vdi MINGW64 ~/desktop/ctlab

$ git init

Initialized empty Git repository in D:/Users/student9/Desktop/ctlab/.git/

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

ctpgml.txt

nothing added to commit but untracked files present (use "git add" to track)

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ git add ctpgml.txt

warning: in the working copy of 'ctpgml.txt', LF will be replaced by CRLF the next time Git touches it

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ git commit -m"wassup"

[master (root-commit) cd0ee4e] wassup

1 file changed, 1 insertion(+)

create mode 100644 ctpgml.txt

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ echo "first program">>ctpgml.txt

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: ctpgml.txt

no changes added to commit (use "git add" and/or "git commit -a")

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$ git log

commit cd0ee4e4cab5d701c307f0b16131c94647725b26 (HEAD -> master)

Author: githubsairaj <79621832+githubsairaj@users.noreply.github.com>

Date: Fri Jan 6 15:17:50 2023 +0530

wassup

student9@student9vdi MINGW64 ~/desktop/ctlab (master)

$