Git Cheat Sheet .org

Search

Create a branch and checkout (switch) to it

```
git checkout -b my-new-branch
```

Creates and checks out (switch) to a new branch named "my-new-branch".

Delete branch

```
git branch -d branch-to-delete
```

Deletes the branch called "my-branch-to-delete".

Merge branches

First, checkout the branch you want to merge in, then run:

```
git merge branch-from
```

Merges the commits from "branch-from" into the currently checked out branch.

Finalize the merge commit after resolving conflicts

```
git commit
```

Concludes a merge after fixing conflicts.

Add file or directory to staging

```
# Add single file
git add file.txt
```

Git Cheat Sheet .org

Search

```
git add file1.txt file2.txt

# Add all text files in current dir
git add *.txt

# Add files in "my-dir"
git add my-dir
```

Ignore files and directories

Edit the .gitignore file (create one if it absent)

```
# Ignore all txt files
*.txt

# Track "file1.txt" even if all other txt files are ignored
!file1.txt

# Ignore a file called "credentials" in the current directory
/credentials

# Ignore all files in any directory named "logs"
logs/

# ignore all txt files in the "logs" folder, but not "logs/apa
logs/*.txt

# ignore all ".txt" files in the "logs" directory and any of i
logs/**/*.txt
```

Provide a message with your commit

```
git commit -m "Commit Message"
```

Show the log containing commit history

```
# Show the most recent 5 commits
git log -5

# Show commits on one line
git log --oneline

# Show a patch with changes introduced by each commit
# Limiting the result to the most recent two commits
git log -p -2
```

Show the latest N commits

```
git log -3
```

Show the log, one line for each commit

```
git log --oneline
```

Revert a commit

```
git revert abc123
```

Reverts changes introduced by that commit "abc123" by creatting another commit that is the "opposite" of "abc123".

Change the message and/or files of the last commit

Using the information in your staging area to amend the last commit

```
git commit --amend
```

will be updated. WARNING: use this only for local commits that have not been pushed to a remote. Amending pushed commits will cause problems for your collaborators.

Add remote repositories

```
git remote add my-remote-name https://gitcheatsheet.org/exampl
```

Fetch data from a remote repository

```
# Fetch from "origin" or the configured upstream branch
git fetch

# Fetch all branches from "my-remote-name" remote
git fetch my-remote-name

# Fetch "branch1" from "my-remote-name" remote
git fetch my-remote-name branch1
```

Downloads data from the remote repository into your local repository, without trying to merge anything with your work.

Pull data from a remote repository

```
# Pull from "origin" remote
git pull

# Pull from "my-remote-name" remote
git pull my-remote-name
```

This command will try to fetch and then merge the remote branch into your local branch.

Git Cheat Sheet .org

Search

```
# Push the current branch on the remote repository
git push

# Push the main (a.k.a. master) branch to the main branch on t
git push origin main
```

Switch to a different (existing) branch using "git checkout"

```
git checkout develop
```

Checks out (switch) to an existing branch named "develop".

Create a new branch

```
git branch my-new-branch
```

Creates a branch named "my-new-branch".

Mark the file as resolved after a merge conflict

```
git add myFile.txt
```

Marks the "myFile.txt" as resolved after a merge conflict.

Abort a merge

```
git merge --abort
```

Cancels the merge process and tries to go back to the state before the merge.

Git Cheat Sheet .org

Search

```
git branch
```

List the available branches with details about the upstream branch and last commit message

```
git branch -vv
```

Using "-v" instead of "-vv" shows less information.

Show the status of the files in the current branch

```
git status
```

Commit changes to the local repository

```
git commit
```

This will open your registerd text editor to allow writing a commit message. Once the editor is closed, the commit will be performed.

Untrack files AND remove them from working tree

```
git rm file1.txt file2.txt
```

Untrack files from staging area, without removing them from the working tree

```
git rm --cached file1.txt file2.txt
```

Git Cheat Sheet .org

Search

Show the commits that affect a specific file or directory

```
git log -- path/to/file
```

Unstage file using "git reset"

```
git reset HEAD fileToUnstage.txt
```

Unstages the file without changing the file contents. It will still be seen as modified, but not staged for commit. Note: this is an alternative to the "git restore" command, which was introduced in Git version 2.23.0.

Unstage file using "git restore"

The "restore" command was introduced in Git version 2.23.0

```
git restore --staged fileToUnstage.txt
```

Unstages the file without changing the file contents. It will still be seen as modified, but not staged for commit.

Revert file using "git checkout --"

```
git checkout -- fileToRevert.txt
```

Replaces the file in the working directory by the latest staged or commited file. WARNING: any local changes made to the file are lost. Git replaces the file with the last staged or committed version.

List the configured remotes

Search

Show the configured remotes together with their URLs

```
git remote -v
```

Remove a remote

```
git remote remove remote-name
```

Removes "remote-name", together with all remote-tracking branches and configuration settings related to that remote.

Resolve conflicts visually using a merge tool

```
git mergetool
```

Opens up the configured merge tool for resolving conflicts. Additionally, depending on the tool you used, you might need to mark the files as resolved using "git add".

Rebase the current branch

```
# Regular rebase
git rebase other-branch

# Interactive rebase
git rebase -i other-branch
```

Rewrites current branch's history so that it has all the commits of "other-branch" and then reapplies the changes in commits that were made in the current branch before the rebase. WARNING: since rebasing rewrites history, you should only use it on local branches.

Git Cheat Sheet .org

Search

Show the changes of both staged and unstaged files since the last commit

```
git diff HEAD
```

Shows what changed since the last commit (both staged and unstaged files). Note: you can use "--cached" instead of "--staged". They mean the same thing.

Show the changes of files that are staged

```
git diff --staged
```

Shows what changed since the last commit and is staged for commit. Note: you can use "--cached" instead of "--staged". They mean the same thing.

Reset the staging area to a specific commit

```
git reset abc123
```

Goes back to commit "abc123" by resetting the staging area to match it, preserving the changed files in the working directory. WARNING: This deletes commits subsequent to "abc123".

Create a new local repository

```
git init
```

This will create a .git directory.

Clone an existing (remote) repository in the current directory

This will copy the remote repository into your current working directory.

Show the current branch name and other information

```
git status
```

Show whether the current branch is up-to-date, ahead or behind the remote branch

```
git status
```

Show the changes of files that are not yet staged for commit

```
git diff
```

Shows what changed since the last commit but is not yet staged.

Show a patch with changes introduced by each commit

```
git log -p -2
```

Note: Using -2 to limit the result to the most recent two commits.

Show the log as a graph

```
git log --oneline --graph
```

The --oneline option is added for readability.

Filter the log entries by author name

```
git log --author='John'
```

Note: the command will display any author that contains the string John, e.g. "John Doe" or "Johnny".

Filter the log entries by committer name

```
git log --committer='John'
```

Note: the command will display any committer that contains the string John, e.g. "John Doe" or "Johnny".

Filter the log entries by date range

Displaying commits made between 2021-01-01 (inclusive) and 2021-02-01 (exclusive):

```
# Using before and after
git log --after="2021-01-01" --before="2021-02-01"

# Using since and until
git log --since="2021-01-01" --until="2021-02-01"
```

Note: The "--after" and "--since" results INCLUDE the specified date (>=), while the "--before" and "--until" EXCLUDE the specified date (<).

Filter the log entries by commit message containing a string

Searching for the text "hello" in commit messages:

```
git log --grep="hello" -i
```

Git Cheat Sheet .org

Search

Reset the working directory to the state of a specific commit

```
git reset --hard abc123
```

Resets the working directory to the state of commit "abc123". WARNING: This deletes uncommitted changes and also deletes commits subsequent to "abc123".

Rename a remote

```
git remote rename old-remote-name new-remote-name
```

Renames the "old-remote-name" into "new-remote-name".

Edit the global configuration

```
git config --global --edit
```

Show the currently configured email address

```
git config user.email
```

From the documentation: "Options --system, --global, --local, --worktree and --file <filename> can be used to tell the command to read from only that location".

Show the email address configured for a specific location (e.g. worktree, local, global, system)

```
# Worktree
git config --worktree user.email
git config --local user.email

# Global
git config --global user.email

# System
git config --system user.email
```

The precedence is: worktree, local, global, system.

Show the currently configured user name

```
git config user.name
```

From the documentation: "Options --system, --global, --local, --worktree and --file <filename> can be used to tell the command to read from only that location".

Show the user name configured for a specific location (e.g. worktree, local, global, system)

```
# Worktree
git config --worktree user.name

# Local (current repository)
git config --local user.name

# Global
git config --global user.name

# System
git config --system user.name
```

The precedence is: worktree, local, global, system.

Set the user name for all repositories

Git Cheat Sheet .org

Search

Set the email address for all repositories

```
git config --global user.email johndoe@example.com
```

Track new or modified files

```
# Add single file
git add file.txt

# Add multiple files
git add file1.txt file2.txt

# Add all text files in current dir
git add *.txt

# Add files in "my-dir"
git add my-dir
```

Add all files to staging

```
# Stage new, modified and deleted files
git add .

# Stage new and modified, ignore deleted files
git add --ignore-removal .

# Stage modified and deleted files, ignore new files
git add -u
```

Inspect a remote

```
git remote show origin
```

Git Cheat Sheet .org

Search

Edit the local configuration

Inside the repository that you want to configure, run:

```
git config --local --edit
```

Set up the default text editor

```
git config --global core.editor "'C:/path/to/executable' -para
```

Set the user name for the current repository

Inside the repository that you want to configure, run:

```
git config --local user.name "John Doe"
```

The --local parameter is optional, as it is the default.

Set the email address for the current repository

Inside the repository that you want to configure, run:

```
git config --local user.email johndoe@example.com
```

The --local parameter is optional, as it is the default.

Cherry-pick commits

use the "-x" option to automatically append a "cherry picked from commit" to the commit message, specifying which commit has been picked.

Push files to stash

```
# Stash local modifications
git stash push -m "My Stash Message"

# Include untracked files
git stash push -u -m "Including untracked files"

# Stash only specified files
git stash push -u -m "Stashing specific files" -- file1.txt f
```

Moves the local modifications into a new stash entry. Using "-u" includes untracked files. The message provided with "-m" is optional.

List the stash entries

```
git stash list
```

Displays the list of stash entries.

Apply a stash entry to the current working tree

```
# Apply the LATEST stash entry (index 0)
git stash apply

# Apply SPECIFIC stash entry (index 1)
git stash apply stash@{1}
```

Pop a stash entry and apply its contents

```
git stash pop


# Pop a SPECIFIC stash entry (index 1)
git stash pop stash@{1}
```

Drop a stash entry from the stash list

```
# Drop the LATEST stash entry (index 0)
git stash drop

# Drop a SPECIFIC stash entry (index 1)
git stash drop stash@{1}
```

Clear all the stash entries

```
git stash clear
```

Edit the system configuration

```
git config --system --edit
```

List all the configured variables

```
git config --list
```

From the documentation: "Options --system, --global, --local, --worktree and --file <filename> can be used to tell the command to read from only that location".

Move files

```
git mv someFile.txt newFile.txt
```

Revert a file using "git restore"

```
git restore fileToRevert.txt
```

Replaces the file in the working directory by the latest staged or commited file. WARNING: any local changes made to the file are lost. Git replaces the file with the last staged or committed version. Note: the "restore" command was introduced in Git version 2.23.0.

Show the file modifications saved in the stash

```
# Show files in the LATEST stash entry (index 0), IGNORING unt
git stash show

# Show files in the LATEST stash entry (index 0), INCLUDING ur
git stash show --include-untracked

# Show files in SPECIFIC stash entry (index 1)
git stash show --include-untracked stash@{1}

# Show ONLY UNTRACKED files in stash entry (index 1)
git stash show --only-untracked stash@{1}

# Show ONLY UNTRACKED files in stash entry (index 1)
### Compatible with older versions of Git
git show stash@{1}^3:
```

Note: older versions of Git do not support the --include-untracked option.

Associate Notepad++ as the default editor

```
git config --global core.editor "'C:/Program Files (x86)/Note
```

Search

Associate VisualStudio Code as the default text editor

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

Associate TextMate as the default editor

```
git config --global core.editor "mate -w"
```

Add a repository inside another repository (using subtrees)

```
git subtree add --prefix my-nested-repo https://gitcheatsheet.
```

Creates a "clone" of the "main" branch of the remote repository into a local directory called "my-nested-repo", squashing the history of the "cloned" repository. Note: instead of specifying the url directly, you can use a remote name, if configured.

Pull a subtree

```
git subtree pull --prefix my-nested-repo https://gitcheatsheet
```

Pulls the "main" branch of the repository specified in the url into a local directory called "my-nested-repo", squashing the history of the pulled repository. Note: instead of specifying the url directly, you can use a remote name, if configured.

Push a subtree

```
git subtree push --prefix my-nested-repo https://gitcheatsheet
```

specified repository url. Note: instead of specifying the url directly, you can use a remote name, if configured.

Associate Atom as the default text editor

```
git config --global core.editor "atom --wait"
```

Associate Sublime Text as the default editor

```
git config --global core.editor "'C:/Program Files (x86)/subl
```