1

Commit a state of the code base

Branch a reference to a commit; can have a tracked upstream

Tag a reference (standard) or an object (annotated)

HEAD a place where your working directory is now

Git Cheat Sheet

01 Git configuration

git config --global

user.name “Your Name” Set the name that will be attached to your commits and tags.

git config --global

user.email “you@example.

com”

Set the e-mail address that will be attached to your commits

and tags.

git config --global

color.ui auto Enable some colorization of Git output.

02 Starting a project

git init [project name]

Create a new local repository in the current directory. If

[project name] is provided, Git will create a new directory

named [project name] and will initialize a repository inside it.

git clone <project url> Downloads a project with the entire history from the remote

repository.

git rm [file] Remove file from working directory and staging area.

04 Storing your work

git stash Put current changes in your working directory into stash for

later use.

git stash pop Apply stored stash content into working directory, and clear

stash.

git stash drop Delete a specific stash from all your previous stashes.

05 Git branching model

git branch [-a] List all local branches in repository. With -a: show all branches

(with remote).

git branch [branch\_name] Create new branch, referencing the current HEAD.

git rebase [branch\_name]

Apply commits of the current working branch and apply them

to the HEAD of [branch] to make the history of your branch

more linear.

git checkout [-b]

[branch\_name]

Switch working directory to the specified branch. With -b: Git

will create the specified branch if it does not exist.

git merge [branch\_name] Join specified [branch\_name] branch into your current branch

(the one you are on currently).

git branch -d [branch\_

name]

Remove selected branch, if it is already merged into any other.

-D instead of -d forces deletion.

03 Day-to-day work

git status

Displays the status of your working directory. Options include

new, staged, and modified files. It will retrieve branch name,

current commit identifier, and changes pending commit.

git add [file]

Add a file to the staging area. Use. in place of the full file path

to add all changed files from the current directory down into

the directory tree.

git diff [file] Show changes between working directory and staging area.

git diff --staged [file] Shows any changes between the staging area and the

repository.

git checkout -- [file] Discard changes in working directory. This operation is

unrecoverable.

git reset <path>...] Revert some paths in the index (or the whole index) to their

state in HEAD.

git commit Create a new commit from changes added to the staging area.

The commit must have a message!

2

11 Ignoring files

cat <EOF > .gitignore

/logs \*

!logs/.gitkeep

/tmp

\*.swp

EOF

To ignore files, create a .gitignore file in your repository with a line for each pattern. File ignoring will

work for the current and sub directories where .gitignore file is placed. In this example, all files are

ignored in the logs directory (excluding the .gitkeep file), whole tmp directory and all files \*.swp.

10 Git installation

For GNU/Linux distributions, Git should be available in the standard system repository. For

example, in Debian/Ubuntu please type inthe terminal:

sudo apt-get install git

If you need to install Git from source, you can get it from git-scm.com/downloads.

An excellent Git course can be found in the great Pro Git book by Scott Chacon and Ben Straub.

The book is available online for free at git-scm.com/book.

09 Synchronizing repositories

git fetch [remote] Fetch changes from the remote, but not update tracking

branches.

git fetch --prune

[remote]

Delete remote Refs that were removed from the remote

repository.

git pull [remote] Fetch changes from the remote and merge current branch with

its upstream.

git push [--tags]

[remote] Push local changes to the remote. Use --tags to push tags.

git push -u [remote]

[branch]

Push local branch to remote repository. Set its copy as an

upstream.

06 Inspect history

git log [-n count] List commit history of current branch. -n count limits list to last

n commits.

git log --oneline

--graph --decorate

An overview with reference labels and history graph. One

commit per line.

git log ref . List commits that are present on the current branch and not

merged into ref. A ref can be a branch name or a tag name.

git log .ref List commit that are present on ref and not merged into current

branch.

git reflog List operations (e.g. checkouts or commits) made on local

repository.

07 Tagging commits

git tag List all tags.

git tag [name]

[commit sha]

Create a tag reference named name for current commit. Add

commit sha to tag a specific commit instead of current one.

git tag -a [name]

[commit sha] Create a tag object named name for current commit.

git tag -d [name] Remove a tag from local repository.

08 Reverting changes

git reset [--hard]

[target reference]

Switches the current branch to the target reference, leaving

a difference as an uncommitted change. When --hard is used,

all changes are discarded. It's easy to lose uncommitted

changes with --hard.

git revert [commit sha] Create a new commit, reverting changes from the specified

commit. It generates an inversion of changes.