Git CheatSheet

Create repository

```
# Create an empty local repository
git init
```

```
# Create a local copy from an existing repository
git clone https://gitlab.cern.ch/project/repo.git
```

Update

```
# Get updates but don’t apply them
git fetch
```

```
# Get updates and apply them
git pull
```

Commit

```
# Add a file to the next commit (staging area)
git add path/to/file
```

```
# commit the changes
git commit
# Then, an editor should open and ask you to give a commit message
```

```
# Add all modified files, commit them and give a commit message
git commit -am “my message”
```

```
# Update last commit
git commit --amend
```

Note: new files must be added via the explicit git add command.

```
# See the current state of the working dir and staging area
git status
```

Publish the code

```
# Publish the code on the remote
git push
```

Help on commands

```
# show help on the given command (what it does, available options…)
git command --help
# Example:
git push –help
```

Manage branches

```
# Create a local branch
git branch branch_name
# Create a new local branch and checkout on it
git checkout -b branch_name
```

```
# Checkout on branch master
git checkout master
```

```
# Checkout on previous branch
git checkout -
```

```
# Remove a local branch
git branch -D branch_name
```

```
# Publish a branch on remote
git push -u origin branch_name
```

```
# List local branches
git branch
# List remote branches
git branch -r
```

History

```
# Show differences between working dir and staging area
git diff [path/to/file]
# Show differences between 2 commits
git diff 3036e6 79ea1a7e
```

```
# Show the logs
git log
# Show the logs for all branches, with a graph to see the branches
git log --all --graph
```

Search in sources

```
# Grep in all files tracked by Git, from the current directory
# Better to start it form the root of the repo
# Options -in does a case insensitive search and gives the line number
git grep -in “search pattern”
```

```
# List all files tracked by git
git ls-files
```

```
# Find the location of a file, knowing its name partially
git ls-files | grep -i “partial name”
```

Play with commits

```
# Copy a commit on top of our current branch
git cherry-pick 3036e6
```

```
# Change the staging area with the given commit
git checkout 3036e6
```

```
# Change the staging area with the previous commit
git checkout HEAD~1
```

```
# Reset the current branch to a commit / branch
git reset 3036e6|branch_name
```

```
# Reset the current branch to a commit / branch + reset the working dir
git reset --hard 3036e6|branch_name
```

Warning: with this last command, you lose all your changes!

Update a branch with master workflow

```
# First we update our local master
git checkout master
git pull
# Then we checkout on the branch we want to update
git checkout branch_name
git pull
# We create a tmp branch as a save state of our current branch
git branch tmp
# Now we start the rebase
git rebase master
# If conflicts, you’ll have to solve them here, for each rebased commit
# NEVER EVER pull HERE!!!
# We push the result
git push --force-with-lease
git branch -D tmp
```

Rename and remove files

```
# Rename: ensure that Git keeps the track of your file
git mv filename newfilename
```

```
# Remove
git rm filename
```

Reset local repo to a known state

```
# Soft reset: only changes the staging area and keep all your changes
# in the working dir
# No loss here
# Replace <REFERENCE> with either a commit hash, a tag or a branch name
git reset <REFERENCE>
```

```
# Hard reset: changes both staging area and working dir!
# You may lose all your changes staged or unstaged!
# Replace <REFERENCE> with either a commit hash, a tag or a branch name
git reset --hard <REFERENCE>
```

Note: reset is used to change the position of a branch pointer, but you need to be careful as you may lose data.