

# Documentation

# OWAES Git

WORKING WITH GIT COMMANDS



Dellisse Manuel

HOWEST GEERT HOFMAN | THE STUDIOS

# Table of Contents

General Git commands .....	3
Clone one branch of the repository .....	3
Stage/unstage changes .....	3
Stage .....	3
Unstage .....	3
Commit changes .....	3
Push commits to remote .....	3
Fetch commits from remote .....	4
Pull commits from remote .....	4
Get local repository status .....	4
Merge branches .....	4
Log of commits .....	4
Changes from a specific commit .....	4
Current repository changes .....	4
Merge tool (fix conflicts) .....	4
Git scenarios .....	5
Getting started and first commit .....	5
Cloning a single branch .....	5
Make changes to the repository (add/modify/remove) .....	5
Stage changes .....	5
Commit stage .....	5
Push commit to remote .....	7
Update local repository from remote .....	8
Fetch remote repository information and compare with local .....	8
Update local repository with unpacked objects .....	8
Resolving conflicts .....	9
Conflict when pushing commit .....	9
Pulling changes .....	9
Resolve conflict .....	9
File with conflict .....	9
Resolved file .....	10
Git status after resolving conflict .....	10

# General Git commands

## Clone one branch of a repository

```
git clone -b <branch> --single-branch <url.git> <folder>
```

Ex.: git clone -b 0-work --single-branch <https://github.com/owaes/owaes.git>  
var/www/owaes

## Stage/unstage changes

### Stage

Stage all changes WITHOUT ignored files:

```
git add . (dot is required)
```

Stage all changes WITH ignored files:

```
git add . -f (dot is required)
```

Select the files you want to stage:

```
git add <file>
```

Select the ignored files you want to stage:

```
git add <file> -f
```

Stage a deleted files:

```
git rm <file>
```

Stage deleted directories:

```
git rm -r <directory>
```

Note: git rm doesn't only stage files but also deletes files from the repository.  
Use "git checkout" to undo repository changes, recovering deleted files included.  
"git checkout" is only possible on unstaged files.

```
git checkout -- <file>
```

### Unstage

Unstage everything:

```
git reset HEAD
```

Select the files you want to unstage:

`git reset HEAD <file>`

## **Commit changes**

`git commit`

## **Push commits to remote**

`git push`

## **Fetch commits from remote**

`git fetch`

## **Pull commits from remote**

`git pull`

## **Get local repository status**

`git status`

## **Merge branches**

`git merge <branch>` (merge x into current branch)

## **Log of commits**

`git log`

## **Changes from a specific commit**

`git show <commit hash>`

## **Current repository changes**

`git diff`

## **Merge tool (fix conflicts)**

`git mergetool` (with system's default graphical diff tool)

# Git scenarios

## Getting started and first commit

### Cloning a single branch

```
crunchbang@crunchbang:~$ git clone -b Test --single-branch https://github.com/Ma
nuel-Dellisse/owaes.git git/owaes
Cloning into 'git/owaes'...
remote: Counting objects: 2052, done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 2052 (delta 37), reused 0 (delta 0), pack-reused 1982
Receiving objects: 100% (2052/2052), 19.86 MiB | 2.19 MiB/s, done.
Resolving deltas: 100% (1041/1041), done.
```

### Make changes to the repository (add/modify/remove)

```
crunchbang@crunchbang:~$ cd git/owaes
crunchbang@crunchbang:~/git/owaes$ vim demo.php
crunchbang@crunchbang:~/git/owaes$ git status
# On branch Test
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       demo.php
nothing added to commit but untracked files present (use "git add" to track)
```

Git status command can be really helpful to keep tracking of what you're doing.

### Stage changes

```
crunchbang@crunchbang:~/git/owaes$ git add .
crunchbang@crunchbang:~/git/owaes$ git status
# On branch Test
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   demo.php
#
crunchbang@crunchbang:~/git/owaes$ git rm demo.php
rm 'demo.php'
crunchbang@crunchbang:~/git/owaes$ git status
# On branch Test
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    demo.php
#
```

### Commit stage

```
crunchbang@crunchbang:~/git/owaes$ git commit
```

```

GNU nano 2.2.6      File: .git/COMMIT_EDITMSG      Modified
Added demo.php
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Committer: CrunchBang LiveUser <crunchbang@crunchbang.(none)>
#
# On branch Test
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   demo.php
#

```

```

crunchbang@crunchbang:~/git/owaes$ git commit
[Test bdae536] Added demo.php
Committer: CrunchBang LiveUser <crunchbang@crunchbang.(none)>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 3 insertions(+)
create mode 100644 demo.php

```

This should only happen the very first time.

```

crunchbang@crunchbang:~/git/owaes$ git config --global user.name "Manuel-Dellisse"
crunchbang@crunchbang:~/git/owaes$ git config --global user.mail manuel.dellisse@student.howest.be
crunchbang@crunchbang:~/git/owaes$ git commit --amend --reset-author

```

```

GNU nano 2.2.6      File: .git/COMMIT_EDITMSG
Added demo.php
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Committer: Manuel-Dellisse <crunchbang@crunchbang.(none)>
#
# On branch Test
# Your branch is ahead of 'origin/Test' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD^1 <file>..." to unstage)
#
#       new file:   demo.php
#

```

You can also directly change your settings in .gitconfig file.

```
[user]
```

```
name = Manuel-Dellisse
mail = manuel.dellisse@student.howest.be
```

## Push commit to remote

```
crunchbang@crunchbang:~/git/owaes$ git push
Username for 'https://github.com': Manuel-Dellisse
Password for 'https://Manuel-Dellisse@github.com':
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/Manuel-Dellisse/owaes.git
47f453e..84df87a Test -> Test
```

## Update local repository from remote

Fetch remote repository information and compare with local

```
crunchbang@crunchbang:~/git/owaes$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Manuel-Dellisse/owaes
 84df87a..6392c33  Test      -> origin/Test
```

Update local repository with unpacked objects

```
crunchbang@crunchbang:~/git/owaes$ git pull
Updating 84df87a..6392c33
Fast-forward
 demo.php |    2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```



## Resolving conflicts

Conflicts are update problems due changes made to the same file at the same time. For instance if some has commit changes to a file called “demo.php”, and you are also making changes to “demo.php” it will cause a conflict. Conflicts occurs when you pull or merge changes.

### Conflict when pulling a commit

```
crunchbang@crunchbang:~/git/owaes$ git push
Username for 'https://github.com': Manuel-Dellisse
Password for 'https://Manuel-Dellisse@github.com':
To https://github.com/Manuel-Dellisse/owaes.git
 ! [rejected]          Test -> Test (non-fast-forward)
error: failed to push some refs to 'https://github.com/Manuel-Dellisse/owaes.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

This means you don't have the latest version. You can't push anything if you aren't up-to-date. Also fetching before pulling is not required anymore since the push has checked the remote.

### Pulling changes

```
crunchbang@crunchbang:~/git/owaes$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Manuel-Dellisse/owaes
 6392c33..d2bc608  Test      -> origin/Test
Auto-merging demo.php
CONFLICT (content): Merge conflict in demo.php
Automatic merge failed; fix conflicts and then commit the result.
```

### Resolve conflict

Conflicts are manually resolved since automatic merge failed with your editor and commit your change.

### File with conflict

```
<?php
<<<<<< HEAD
Yours { echo "Demo update repository";
        echo "Demo conflict";
        echo "Conflict";
        echo "Conflict4";
=====
Theirs { echo "This is a demo";
         echo "Conflict2";
         echo "Conflict3";
>>>>>> d2bc608f2624a6f32950790898c3ee4687075052
?>
```

## Resolved file

```
<?php
    echo "This is a demo";
    echo "Demo conflict";
    echo "Conflict";
    echo "Conflict2";
    echo "Conflict3";
    echo "Conflict4";
?>
```

Of course most of time it's not that simple, you will have to check Github.com to see their changes because renames can't be shown. In the example "Demo update repository" has been renamed to "This is a demo". The only way to know that is by checking on Github.com. Or you can also make use of tools to help you merge changes.

```
<?php
-   echo "Demo update repository";
+   echo "This is a demo";
+   echo "Conflict2";
+   echo "Conflict3";
?>
```

## Git status after resolving conflict

```
crunchbang@crunchbang:~/git/owaes$ git status
# On branch Test
# Your branch and 'origin/Test' have diverged,
# and have 1 and 1 different commit each, respectively.
#
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#       both modified:      demo.php
#
no changes added to commit (use "git add" and/or "git commit -a")
```

It's always a good thing to use git status when something unusual happens or happened.

When this is done just stage, commit and push changes.



## Merge branch 'Test' of <https://github.com/Manuel-Dellisse/owaes> into ...

Manuel-Dellisse authored 5 minutes ago



## Modification in demo.php

Manuel-Dellisse authored 18 minutes ago



## Demo conflicts

Manuel-Dellisse authored 19 minutes ago

<?php

```
- echo "Demo update repository";  
+ echo "This is a demo";  
  echo "Demo conflict";  
  echo "Conflict";  
+ echo "Conflict2";  
+ echo "Conflict3";  
  echo "Conflict4";
```

?>