# Distributed Version Control with Git\_ Mastering the Git command line - .pdf

* working tree - The current collection of files is called the working tree.
* git add . – adding to the staging area
* git commit – commit the changes in staging area
* git log - - HEAD # seeing the git log for the HEAD file
* git log HEAD - - # seeing the git log for the HEAD reference
* git log HEAD # if there is no HEAD file, the second command becomes
* Commit
  + This commit object is addressable via a hash (SHA-1 checksum).
  + Commit ID
  + Tree Object ID > snopshot of the filesystem
  + Author
  + Committer
  + Commit Message

# Terminology’

## git config

* Git allows you to store user settings in the .gitconfig file located in the user home directory. This is also called the global Git configuration.
* You can also store repository specific settings in the .git/config file of a repository. Use the -- local or use no flag at all. If neither the --system not the --global parameter is used, the setting is specific for the current Git repository.
* Avoid merge commits for pulling
* Excercise
  + # configure the user which will be used by Git
  + # this should be not an acronym but your full name
  + git config --global user.name "Firstname Lastname"
  + # configure the email address
  + git config --global user.email [your.email@example.org](mailto:your.email@example.org)

# setup vim as default editor for Git (Linux)

* git config --global core.editor vim

#To query your Git settings, execute the following command:

git config --list

#If you want to query the global settings you can use the following command.

git config --global --list

## Branch

* Selecting a branch in Git terminology is called to checkout a branch.
* HEAD - HEAD is a symbolic reference most often pointing to the currently checked out branch. Sometimes the HEAD points directly to a commit object, this is called detached HEAD mode

## Staging / Index

* Index is an alternative term for the staging area.

## Revision

## staging area

## Tag

## URL

* Git distinguishes between fetchurl for getting new data from other repositories and pushurl for pushing data to another repository

## working tree

### File Status

* untracked:
* tracked:
* staged:
* dirty / modified > the file has changed but the change is not staged

## Commit

* A commit reference can be a simple reference (simple ref), in this case it points directly to a commit. This is the case for a commit hash or a tag. A commit reference can also be symbolic reference (symbolic ref, symref). In this case it points to another reference (either simple or symbolic). For example HEAD is a symbolic ref for a branch, if it points to a branch. HEAD points to the branch pointer and the branch pointer points to a commit.
* You can use ^ (caret) and ~ (tilde) to reference predecessor commit objects from other references. You can also combine the ^ and ~ operators.
* The Git terminology is parent for ^ and ancestor for ~.
* Commit ranges with the double dot operator
* Commit ranges with the triple dot operator (either one commits from different branches)

## Ignoring

### files and tracking empty directories

* # ignore all bin directories
* # matches "bin" in any subfolder
* bin/
* # ignore all target directories
* target/
* # ignore all files ending with ~
* \*~
* You can create the .gitignore file in the root directory of the working tree to make it specific for the Git repository.
* The .gitignore file tells Git to ignore the specified files in Git commands. You can still add ignored files to the staging area of the Git repository by using the --force parameter, i.e. with the git add --force [paths] command.
* It is good practice to commit the local .gitignore file into the Git repository so that everyone who clones this repository have it.

### Global (cross-repository) .gitignore settings

# Create a ~/.gitignore in your user directory

cd ~/

touch .gitignore

# Exclude bin and .metadata directories

echo "bin" >> .gitignore

echo ".metadata" >> .gitignore

echo "\*~" >> .gitignore

echo "target/" >> .gitignore

# for Mac

echo ".DS\_Store" >> .gitignore

echo ".\_\*" >> .gitignore

# Configure Git to use this file

# as global .gitignore

git config --global core.excludesfile ~/.gitignore

### a

## A