git

basic workflow

```
# initialize an empty repository
> git init

# get the current status of every file present in working directory
> git status

# get short status
> git status -s

# note: git status -s command returns a status with two characters
# 1st character: shows the status of file with respect to the staging area
# 2nd character: shows the status of file with respect to the working area

# ??: untracked file (the repository does not recognize the file or does
not have any version of this file created yet)
# A : the changes are present in the staging area and will be added to the
repository after committing
#  M: the changes are present in the working directory and the file is
modified in the working directory
# M : the changes are moved to the staging area
# UU: the file has conflicts

# add the file/files to staging area
> git add <file name>
> git add .

# create a version of all the files present in staging area
> git commit -m <message>

# get the list of all commits
> git log

# get the list of logs in one line with color and graph
> git log --oneline --graph --color

# get the difference between the latest version and the previous version
> git diff

# get the last version from repository and replace it with current version
# note: even if the file is deleted from the working directory, you can
still bring it (last version of it) back from the repository
> git checkout <file name>

# soft reset:
# - move all the changes from staging area to the working directory
```

```
# - no changes will be lost in the process
> git reset

# hard reset:
# - remove all the changes from staging area or working directory
# - get last version of all the files and replace with current version
(irrespective of their location)
# - note: please execute this command on your own risk
> git reset --hard

# remove all the metadata or repository
# note: this command will remove all the history
> rm -rf .git
```

branches

- reference to a commit in the history

```
# get the list of branches
# note: the branch which has * in front of the name, is the current branch
> git branch

# to find out the current branch, you can use command git status and look
for the first line to get the branch name

# git uses HEAD to find the current branch
# HEAD: reference to the current branch

# create a new branch
# note: this command does not switch to the newly created branch
> git branch <branch name>

# switch to another branch
> git checkout <branch name>

# create a new branch and switch to it immediately
> git checkout -b <branch name>

# merge a branch in another branch
# (merging a feature branch)

# example: merging a branch named prime-number in main branch (source)
# step1: checkout the source branch (the branch in which you want to merge
another branch)
> git checkout main

# step2: merge the changes from another branch to the source branch
# > git merge <branch name>
> git merge prime-number
```

```
# delete a branch
# note: you CAN NOT delete the current branch
> git branch -d <branch name>

# conflict scenario
# - when a file gets modified by two branches on the same line(s), git
does not understand which changes to keep and which ones to remove, this
scenario is called as a conflict scenario
# - conflict scenario CAN NOT be resolved automatically
# - conflict scenario MUST be resolved manually
# - conflicted file(s) will have a conflict marker (>>>>>> or <<<<<<<)
# - resolving a conflict means, removing the conflict markers
# - once the conflicts are resolved, committing a new version is mandatory
```

shared repository

```
# types
# - local
#   - present on local machine
# - shared
#   - present on the server
#   - also known as remote repository
#   - to create a shared repository
#     - create a project or repository on shared repository server
#     - connect the local repository to the remote one
#   - e.g. GitHub, GitLab, BitBucket

# get the remote repository url
> git remote -v

# add the remote repository
# > git remote add <server alias> <repository url>
> git remote add origin <repository>

# send all the local changes to the remote repository
> git push <alias> <branch name>

# pull all the changes from remote to local repository
> git pull <alias> <branch name>

# remove the remote repository from local one
> git remote remove <alias>
```