git

basic workflow

```
# initialize an empty repository
> git init

# get the current status of the repository
> git status

# get the short status of the repository
> git status -s

# statuses
# note:
# - there will be two characters in the status
# - first character: shows the status of the file against the staging area
# - second character: shows the status of the file against the working
# ??: no version is created in the repository yet (untracked)
# A : the changes to the file are present in the staging area and will be
added to the repository
#  M: the file is modified and present only in working directory
# M : the file is present in the staging area
#
#

# add the changes to the staging area
# > git add <file name(s)>

# add changes to all the files in the current directory
> git add .

# commit the changes to the repository
# > git commit -m <commit message>

# get the difference between the working directory version and the last
version present in the repository
# +: the line added in the current version
# -: the line removed from the previous version
> git diff

# get the list of commits
> git log

# get the list of commits with
# - oneline: get only one line description of the log
# - graph: show the graph of the repository
# - color: show the colors while rendering the graph
> git log --oneline --graph --color
```

```
# remove all the changes made to the file (from working directory)
# > git checkout <file name>

# move all the changes from staging area to the working directory
> git reset

# remove all the changes from staging area and working directory
# note: please execute this command on your own risk
> git reset --hard
```

git stash

```
# move the changes from working directory to the stash area
> git stash

# move the changes from working directory to the stash area with a message
# > git stash save <message>
> git stash save "first-implementation"

# get the list of changes added to the stash area
> git stash list

# remove the last changes from stash area and apply them on working
directory
# note: pop = apply + drop
> git stash pop

# apply the specific change to the working directory and remove the entry
from stash area
# note: pop = apply + drop
# > git stash pop <stash entry id>

# apply the changes from stash area to the working directory
# note: the changes will NOT be removed from stash area
# > git stash apply <stash entry id>

# remove the changes from stash area
# > git stash drop <stash entry id>

# remove everything from stash area
> git stash clear
```

branching

```
# get the list of branches
# note: the branch which has star in front of it, is the current branch.
Which means, the changes will be committed to the current branch
# note: git internally maintain a pointer named as HEAD to point to the
current branch
> git branch

# find the current branch
# option1: get the list of branches and find the star
# option2: fire the command git status
# option3: fire the git log command

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

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

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

# merge a branch
# step1: go the branch in which you want to merge another branch
# > git checkout <branch name>
# step2: merge another branch
# > git merge <branch name>

# rename current branch
# > git branch -M <new branch name>

# delete a branch
# > git branch -d <branch name>

# conflict scenario
# - when two branches modify same file on same line number(s), while
merging them, git goes in conflict scenario and marks the file as
conflicted file with conflict marker (>>>>>>>)
# - conflict MUST be resolved manually (explicitly)
# - once the conflict is resolved, developer MUST commit the changes
explicitly
```

remote

- the repository on local machine is called as local repository
- the repository on remote server (shared server) (e.g. GitHub, GitLab etc.) is called as remote repository

```
# get the remote repository url
> git remote

# link the remote repository with local one
# > git remote add <remote repo alias> <remote repo url>

# push the changes from local repo to remote repo
# > git push <alias> <branch name>
> git push origin main
```