Computer Language

Git

global settings

- get all global settings
  - the settings will be read from <home directory>/.gitconfig

```
> git config --global --list
```

- modify the global settings

```
> git config --global user.name <user name>
> git config --global user.email <email>
> git config --global core.editor <vim>
```

basic operations

- initialize the git repository

```
> git init
```

  - creates a new directory named .git

- check the status of repository

```
> git status
> git status -s
```

  - statuses
    - **??**
      - the file is untracked file
      - the file is not yet tracked
      - untracked: the file is not present (there is no version of it created yet) in the .git directory
    - **A**
      - A -> Add to the repository
      - this will appear only for those file which were not yet tracked (new files)
      - the file is added to the commit area
      - once the changes are committed, the new version of this file will get created
    - **[M]**

        - M -> modified
        - this status will appear only for those files which have been already tracked
        - the file is changes since the last commit
        - the file is not yet added to the commit area
    - **[M]**
        - M -> modified
        - this status will appear only for those files which have been already tracked
        - the file is added to the commit area and ready to create a new version
- add a file to commit (logical) area

```
# add only selected file
> git add <file names>

# add all the files in the current directory
> git add .
```

- create a version of selected files

```
> git commit -m <commit message>
```

- get the logs from the current repository

```
> git log
```

- get the differences from the last commit

```
# difference between the last committed version for a selected file
> git diff <file name>

# show the differences for every file since the last committed version
> git diff
```

stash commands

- these commands are used to move to stash area (logical area)
- move the changes to stash area

```
> git stash
```

- list the versions pushed to the stash area

```
> git stash list
```

- get the specific version from stash keeping the changes in the stash area

```
> git stash apply <version number>
```

- get the last version by keeping the changes in the stash area

```
> git stash apply
```

- get the specific version from stash removing the changes in the stash area

```
> git stash pop <version number>
```

- get the last version by removing the changes in the stash area
  - the pop works with last in first out strategy

```
> git stash pop
```

- clear all the changes pushed to the stash area

```
> git stash clear
```

resetting the changes

- discard the changes and get the last version of the file

```
# it will bring the last version of index.html from repository
> git checkout index.html
```

- soft reset: used to remove the changes from staging area

```
> git reset
```

- hard reset: get the latest versions of every file and replace with the current versions
  - used to discard the changes after moving to the staging area
  - git reset --hard = git reset + git checkout

```
> git reset --hard
```

git branching

- get the list of branches

```
# the branch having star is the current branch
> git branch
```

- get the current branch

```
> git status
```

- create a new branch

```
> git branch <branch name>
```

- switch to another branch

```
> git checkout <branch name>
```

- create and switch to the new branch with a single command

```
> git checkout -b <branch name>
```

- delete a branch

```
> git branch -d <branch name>
```

- get the logs

```
# --online: summary in only one line
# --decorate: show the HEAD pointer
# --graph: show the graph of branch
> git log --oneline --decorate --graph
```

- merge one branch with another

```
> git merge <branch name>
```

  - notes:
    - switch to the branch in which you want to merge the changes
    - then use the merge command

```
# we want to merge the changes from feature-branch to master
# checkout the master first
> git checkout master

# merge the changes from feature-branch
> git merge feature-branch
```

remote/shared repositories

- register yourself on GitLab/GitHub (shared repository server)
- create a project on shared repository server
- add the current repository from your machine to the remote server

```
> git remote add origin <remote repository server url>
```

- get the remote server url

```
> git remote -v
```

- push the changes from your machine to the remote server

```
> git push origin <branch name>
```

```
# push the master branch changes to the remote server
> git push origin master
```