Contents

Chapter 1

github basics

1.1 Initialize a repository

Go to the folder where you want to create a repository. Then use *git init* command:

```
git init
```

The repository is initialized and the hidden folder .git is created.

1.2 Check the status of a repository

```
git status
```

It reveals whether there are non-tracked files. i.e. files which need to be added to the repository and other information about the repository (for examples files to commit).

1.3 Add files to the repository

```
git add <filename>
git add –A .
```

The last command adds all files and folders below the current one. It is possible to use wild-cards, but in those cases remember to use the single quotes. For example:

```
git add '*.txt'
```

1.4 Committing changes

Once added a file goes into the staging area. It means it is added but has not been committed yet. To commit changes:

```
git commit -m "message goes here"
```

This command will commit all the changes to all files.

1.5 Viewing logs

To see a log of all changes there is a dedicated command:

```
git log
git log --summary
```

1.6 Pushing local repository to remote server

To push the local repository to the github server we need to add a remote repository:

```
git remote add origin <url of the remote repository>
```

origin is the name usually used for the remote repository, but another name can be used too. So now my remote repository is called origin and the main branch of my local repository is *master*.

1.7 Push to remote repository

Once the remote repository is ready, we shall push the local changes to the remote repository. Using the -u option allows to remember the configuration *push* has to use:

```
git push -u origin master
```

1.8 Pull from remote repository

Other people have done some work and they have pushed their changes to the remote repository. I can pull those changes to my local repository.

```
git pull origin master
```

1.9 Check differences

To check the differences between the current status (after a pull) and my last commit I can use the *diff* command.

```
git diff HEAD
```

HEAD is a pointer that holds your position within all your different commits. By default *HEAD* points to your most recent commit, so it can be used as a quick way to reference that commit.
To show the differences in staged files (i.e. files not committed yet):

```
git diff --staged
```

1.10 Remove staged files

To remove a staged file use the *reset* command:

```
git reset <filename>
```

1.11 Rewind to a commit point

To "rewind" everything back to the point where a certain commit was made use the *checkout* command:

```
git checkout -- <filename>
```

1.12 Branches

A branch is a copy of an existing repository. I can work on a branch, make commits and then merge the branch back into the master repository.
To create a branch use the *branch* command.

```
git branch <branchname>
git checkout <branchname> --> switches to new branch
git checkout -b <branchname> --> creates a new branch and switches to it
```

Once a branch is ready and commit, it can be pushed to the remote repository too.

1.13 Remove files

To remove files from a repository use the *rm* command:

```
git rm <filename>
```

The *rm* command, not only removes the files from the disk, it also stages the removal of the files from the repository.
To recursively remove all folders and files from the given directory us the *-r* option:

```
git rm -r <foldername>
```

If a file is removed with the standard *rm* command (I mean the OS command), then either *git rm* must be used later or use:

```
git commit -a
```

The *-a* option auto-removes deleted files.

1.14 Merging a branch

To merge a branch with the master. First switch to the master:

```
git checkout master
```

then merge

```
git merge <branchname>
```

If you're hosting your repo on GitHub, you can do something called a pull request. A pull request allows the boss of the project to look through your changes and make comments before deciding to merge in the change.

1.15 Remove a branch

After a branch has been merged, the branch can be removed:

```
git branch -d <branchname>
```

note that *-d* works only on branches which have been merged.
To delete a branch which has not been merged use *-d* and *-f* options together or just the *-D* option.

1.16 Reset a branch

If you get the error

error: Your local changes to the following files would be overwritten by merge

when using git pull, then try to use

```
git reset --hard
```

That should fix the problem.