1. *Create the repository*
2. *Copy the repos address*
3. *Open the folder and right click and choose Git bash here.*
4. *In Git bash type:*

```
$ git init
$ git clone <repos adress>
```

5. *Close the bash, and inside the cloned folder reopen git bash.*
6. *Move files/folders to the Cloned folder*
4. *In Git bash type:*

```
$ git init
$ git add .
$ git commit –m “message”
$ git push
```

Deleting all files or some files in repos:

```
cd /tmp
git clone /your/local/rep  # make a temp copy
cd rep
git rm -r *                # delete everything
cp -r /your/local/rep/* .  # get only the files you want
git add *                  # add them again
git status                 # everything but those copied will be removed
git commit -a -m 'deleting stuff'
cd /your/local/rep
git pull /tmp/rep          # now everything else has been removed
```

När man skapar ny branch:

```
$ git branch work
$ git push --set-upstream origin work
```

När man ska uppdaterar master från en annan branch (t.ex. work):
Man byter först till master:

```
$ git checkout master
```

Sedan mergar:

```
$ git merge work
```

When you probably want to create a local branch, which is tracking the remote branch:

```
$ git branch test origin/test
```

To delete a local branch

```
$ git branch -d the_local_branch
```

To see all remote and local branch:

```
$ git branch -a
```

To see all changed file before staging (before running $ git add .)

```
$ git diff
```

To see all changed file After staging (after running $ git add .)

```
$ git diff --staged
```

```
$ git diff --cached
```

To Unstaging a file

```
$ git reset HEAD <Filename.xx>
```

The file will be modified but not unstaged.

To Unmodifying a modified file

```
$ git checkout -- <Filename.xx>
```

To repeat last commit command:

```
$ git commit --amend
```

To get data from your remote projects:

```
$ git fetch [remote-name]
```

To see the commit history:

```
$ git log --pretty=oneline
```

```
$ git log --since=2.weeks
```

```
$ git log --pretty=format:"%h %s" –graph
```

```
$ git log --oneline --decorate
```

To see branches:

```
$ git branch
```

To see last commit on each branch:

```
$ git branch -v
```

```
$ git branch -vv
```

To see which branches are already merged into the pointer branch RESPECTIVE not merged branches:

```
$ git branch --merged
```

```
$ git branch --no-merged
```

To listing Tags:

```
$ git tag
```

```
$ git tag –l ’v1.8.5*’
```

To create an Annotated Tag:

```
$ git tag –a V.1.4 –m ’my version 1.4’
```

To see tag data info:

```
$ git show v.1.4
```

To share and transfare tags to remote servers (git push will not do it by default):

```
$ git push origin V.1.4
```

For single tag and

```
$ git push origin --tags
```

For all tags that are not already there to the remote server.

To have clear history compare with merge:

```
$ git rebase <branch_name>
```

```
H:\>c:
C:\>cd Users/34958
C:\Users\34958>cd desktop

C:\Users\34958\Desktop>dir
 Volume in drive C is PwC_OS
 Volume Serial Number is FCD3-FD31

 Directory of C:\Users\34958\Desktop

2018-08-23  15:28    <DIR>          .
2018-08-23  15:28    <DIR>          ..
2018-08-23  16:05    <DIR>          New folder
               0 File(s)              0 bytes
               3 Dir(s)  386 716 590 080 bytes free


C:\Users\34958\Desktop>cd new folder

C:\Users\34958\Desktop\New folder>cd uppgift1

C:\Users\34958\Desktop\New folder\Uppgift1>git branch kia

C:\Users\34958\Desktop\New folder\Uppgift1>git add 4.txt
fatal: pathspec '4.txt' did not match any files

C:\Users\34958\Desktop\New folder\Uppgift1>git create 4.txt
git: 'create' is not a git command. See 'git --help'.

The most similar command is
        reset


C:\Users\34958\Desktop\New folder\Uppgift1>git add 3.docx


C:\Users\34958\Desktop\New folder\Uppgift1>git commit -m "nw"
[master b22fc23] nw
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 3.docx

C:\Users\34958\Desktop\New folder\Uppgift1>git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads.
```

```
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 221 bytes | 55.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/KiarashBeh/Uppgift1
   3e61507..b22fc23  master -> master

C:\Users\34958\Desktop\New folder\Uppgift1>git checkout -b kia
fatal: A branch named 'kia' already exists.

C:\Users\34958\Desktop\New folder\Uppgift1>git checkout kia
Switched to branch 'kia'
D       1.docx

C:\Users\34958\Desktop\New folder\Uppgift1>git push
fatal: The current branch kia has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin kia


C:\Users\34958\Desktop\New folder\Uppgift1>git add 1.pptx

C:\Users\34958\Desktop\New folder\Uppgift1>git commit -m "test"
[kia bda298c] test
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.pptx

C:\Users\34958\Desktop\New folder\Uppgift1>git push
fatal: The current branch kia has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin kia


C:\Users\34958\Desktop\New folder\Uppgift1>git push --set-upstream origin kia
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 226 bytes | 56.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/KiarashBeh/Uppgift1
 * [new branch]      kia -> kia
Branch 'kia' set up to track remote branch 'kia' from 'origin'.
```

C:\Users\34958\Desktop\New folder\Uppgift1>