# Cloud Native Computing

**University**

of Applied Sciences

**Würzburg-Schweinfurt**

**Faculty of Computer Science and Business Information Systems**

27.06.2022

**Exercise: Basic Git and Github**

Download Git to your Linux environment. Please use this Link

**https://git-scm.com/downloads**

----------------------------------------------------------------------Pause------------------------------------------------------------------

Check that Git is installed using this command

**git --version**

Setup some global configuration variables. This will tell other people, who checking the code in and out etc. This name will be added to your check-ins and code changes

**git config --global user.name "<your-name>"**

**git config --global user.email "<your-e-mail>"**

**git config --list**

Create your exercise folder to store your git repository

**mkdir my-first-git-repo**

**cd my-first-git-repo**

Run this to create repository. .git folder will be created.

**git init**

Check the status of your git repository

**git status**

Create a file called “first-file.txt” and write and text “first line” in it

**touch first-file.txt**

**echo “first-line” >> first-file.txt**

Check the status of your git repository again. What do you see?

**git status**

Move our file to staging area

**git add first-file.txt**

Check the status of your git repository again. What do you see?

**git status**

Commit your change to local repository

**git commit -m "my first commit"**

Check the status of your git repository again. What do you see?

**git status**

----------------------------------------------------------------------Pause------------------------------------------------------------------

Now we will push our change to remote repository, but we need one first. In this exercise, we will use Github. Please create a Github account with the following link. If you already have one, you can skip this part.

**https://github.com/signup?ref\_cta=Sign+up&ref\_loc=header+logged+out&ref\_page=%2F&source=header-home**

----------------------------------------------------------------------Pause------------------------------------------------------------------

On the Github UI, create a new repository with the name “my-first-git-repo” as a public repository and don’t add Readme.md file.

----------------------------------------------------------------------Pause------------------------------------------------------------------

Check on which branch we are currently on

**git branch -a**

Back to your terminal, create a main branch for pushing your commit to Github repository

**git branch -M main**

Add Github remote repository to local alias ”origin”

**git remote add origin https://github.com/<your-github-id>/my-first-git-repo.git**

Create Github access token

* Go to
  + Settings -> Developer settings -> Personal access tokens -> Generate new token
* Check on “repo”
* Copy the token (It will be displayed only once)

Push your commit to Github repository

* You will be asked for username (E-Mail) and password (Token)

**git push -u origin main**

Now you should see “first-file.txt” in your my-first-git-repo Github repository

You can check that your local git repository is now connected to remote Github repository

**git remote -v**

We can clone our file back to our local machine, but in another folder. Create a new folder and run this command to clone your Github repo to local

**cd ..**

**mkdir my-clone-git-repo**

**cd my-clone-git-repo**

**git clone**  [**https://github.com/<your-github-id>/my-first-git-repo.git**](mailto:git@github.com:%3cyour-github-id%3e/my-first-repo.git) **.**

Check the cloned repo

**ls -la**

In your clone repository folder, check the branch there

**git branch -a**

In your clone repository local, add a new line to “first-file.txt”

**echo “second-line” >> first-file.txt**

Then add the change to staging and commit it to local repo

**git add first-file.txt**

**git commit -m "commit from cloned repo"**

Push your change from cloned repository to remote Github repository. This will work since it was pushed from the same machine

**git push -u origin main**

Now you should see your update in Github repository

Go back to my-first-git-repo. You can also pull the change from Github. You should see the second line appear there

**git pull origin main**

Check commit number. You will see git number there

**git log**

You can also get all commit number from Github

Check what changes/details has been made in the system particular this commit ID

**git show c9a78f**

You can see the difference between two commits with this command. Get commit numbers from command above. (you don’t need to copy the whole commit number)

**git diff <commit-nr> <commit-nr>**

**git diff a6259b1 388bf85**

----------------------------------------------------------------------Pause------------------------------------------------------------------

Git introduces. gitignore file, which will contain filename that will not be added to the repository

**touch ignore-this-file.txt**

**touch .gitignore**

**echo "ignore-this-file.txt" >> .gitignore**

Add and commit every files in the folder

**git add .**

**git commit -m "try to commit file in gitignore"**

Push your commit to Github repository

**git push -u origin main**

Check your Github repository. You will see only the .gitignore file there, but ignore-this-file.txt will not be there.

If you accidentally add wrong file to staging area, you can undo that using git reset command

**touch should-not-be-added.txt**

**git add should-not-be-added.txt**

**git status**

**git reset should-not-be-added.txt**

**git status**

You can see which files are now in commit stage

**git ls-files**

----------------------------------------------------------------------Pause------------------------------------------------------------------

Since Git support non-linear development of software, we will see concept of branching.

We create a new development branch

**git branch development**

Check how many branches are available. You will see that you are now on main branch

**git branch -a**

Switch to development branch

**git checkout development**

Check again on which branch you are

**git branch -a**

You will see that all commited files are also on development branch

**git ls-files**

Try to create a file on development branch

**touch development-file.txt**

Add and commit this file

**git add development-file.txt**

**git commit -m "commit development file"**

Check commited file and file in the folder

**git ls-files**

**ls -la**

Change back to main branch

**git branch main**

Check commited file and file in the folder. Now you won’t see development-file.txt neither in commited file nor in folder

**git ls-files**

**ls -la**

You can also push your change in the branch to Github by specifying name of the branch

**git push -u origin development**

If you want to merge your change from development to main branch. You have to be on master branch first. It is also the best practice to pull the change from main branch on remote repository first, because someone may add some change in between.

**git checkout main**

**git pull origin main**

**git merge development**

show the branch that we have merge to master so far

**git branch --merge**

Then at last you can push your merge to main branch in repo

**git push origin main**

Now on Github main branch will have development-file.txt inside

You can also delete development branch from **remote** repository

**git push origin --delete development**

Check available branch

**git branch -a**

You can also delete development branch from **local** repository

**git branch --delete development**

If you have committed some bad code and want to rollback. You can rollback to specific commit number.

**git log**

**git reset <commit-number>**