**My very own Git Notes**

1. Install git
2. Open cmd
3. Check if it works by typing   
   git
4. Type   
   git config  
   (there is a global config for this computer and there can be project based configs)
5. Type   
   git config --list or git config -l
6. To change email or user name:  
   git config –-global user.name = "Alexandra Rollins"  
   git config –-global user.email = [alexandra.rollins@bwv-aachen.de](mailto:alexandra.rollins@bwv-aachen.de)
7. Inside project folder:   
   git init
8. To show status of files: git status
9. To stage: git add   
   git add . #adds all files  
   to unstage: git rm   
     
   #Adding a readme file:  
   git add README.md
10. git commit -m „initial commit”



git remote add origin [git@github.com:schulefant/dp01-strategy.git](mailto:git@github.com:schulefant/dp01-strategy.git)

git commit -m “initial commit”

## -u sets up the default for pushing   
## origin is the github location specified by the git remote add command  
## master is the master branch on the origin

git push -u origin master

## using the default settings indicated on the first push:

git push

git log –all –decorate –oneline –graph

alias <aliasName>='<long command>'  
Bsp:  
alias graph='git log --all --decorate --oneline --graph'

Working with SSH keys in github

1. Open Git Bash.
2. Paste the text below, substituting in your GitHub email address.

$ ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"

This creates a new ssh key, using the provided email as a label.

> Generating public/private rsa key pair.

1. When you're prompted to "Enter a file in which to save the key," press Enter. This accepts the default file location.

> Enter a file in which to save the key (/c/Users/*you*/.ssh/id\_rsa):*[Press enter]*

1. At the prompt, type a secure passphrase. For more information, see ["Working with SSH key passphrases"](https://help.github.com/en/articles/working-with-ssh-key-passphrases).
2. > Enter passphrase (empty for no passphrase): *[Type a passphrase]*

> Enter same passphrase again: *[Type passphrase again]*

Start ssh agent

# start the ssh-agent in the background

$ eval $(ssh-agent -s)

> Agent pid 59566

Add key to ssh agent:

$ ssh-add ~/.ssh/id\_rsa

**Adding a new SSH key to your GitHub account**

# Open rsa.pub in an editor and copy it to clipboard

# online go to your user system settings and add your key with add new key

**Testing a connection:**

1. Open Git Bash.
2. Enter the following:

ssh -T git@github.com

1. You may see a warning like this:

> The authenticity of host 'github.com (IP ADDRESS)' can't be established.

> RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.

> Are you sure you want to continue connecting (yes/no)?

or like this:

> The authenticity of host 'github.com (IP ADDRESS)' can't be established.

> RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

> Are you sure you **want** to continue connecting (yes/no)?

1. Verify that the fingerprint in the message you see matches messag in step 2, then type   
   yes:

> Hi *username*! You've successfully authenticated, but GitHub does not

> provide shell access.

**branching and merging:**

git branch shows which branch we are currently on

git branch -a shows all branches and highlights branch we are currently on

git branch -r shows only the remote branches

git branch <nameOfBranch> creates a new branch

git checkout <nameOfBranch> moves HEAD-pointer to the branch

git checkout <hashOfCommit> moves HEAD-pointer to the commit (headless state)

git diff <normMaster>..<nameOfBranchToMerge> shows the differences between the 2 branches

git log --oneline shows all commits in a compact form

git log --graph shows branching and merging

git branch –merged shows which branches were merged

git branch -d <nameOfBranchToBeDeleted> deletes a branch🡪 has safety net, if it has not been merged

git branch -D <nameOfBranchToBeDeleted> forces a deletion of a branch, without safety net

git merge <branchToMergeIntoCurrent> **fast merge** integrates the changes of the branch into current

git merge <branchToMergeIntoCurrent> **3way merge** integrates the changes of the branch into current

git stash saving changes without having to commit them before   
 changing a branch

git stash save “stash message” stashing with a message

git stash list shows the stashes

git stash list -p shows stashes with stages of change

git stash apply applies changes until most recent stash to branch

git stash apply <label> applies changes until specified stash-label (stash@{1}) to branch

git stash pop removes the last stash without applying it

**working with remotes (github):**

git fetch origin/master downloads the master branch from the remote repo

git merge origin/master integrates the remote into our local repo

git pull origin/master fetch + merge in one step

***forking 🡪***  makes a copy of an online repo in your own online account

git remote add <localName> <addressOfOtherRepoThatWeForked>  
Bsp:  
git remote add upstream [betty@github.com:betty-remote/repoName.git](mailto:betty@github.com:betty-remote/repoName.git)

git fetch upstream to have it on the local pc

***pull request /merge request 🡪*** asks the owner of the repo we forked from to fetch and merge my   
 contribution. Normally, we create a branch that we offer

**You can add a subrepository by typing:**

git modules integrate a repository in a current state, but don't update it when the remote repo gets updated

git subtrees integrate a repository and keep it synchronized with remote repo

git subtree add --prefix {local directory being pulled into} {remote repo URL} {remote branch} --squash`

Bsp

git subtree add --prefix subtreeDirectory https://github.com/username/project.git master --squash

--The End--