**Basic principle**

The basic principle of Git Hub is “branching”. Branch is nothing else than a local copy of the main code, which you modify and then merge with the main (master) branch. Git Hub checks if the code can be merged, preventing the errors, when for example a number of people working simultaneously on the same project.   
  
**Basic terminology**

repository project  
branch your copy of the code. Github supports multiple branches.   
pull download / get updates from the online master branch

commit a kind of “confirm the changes” made to the code. Each “commit” must have a commentary, describing what changes have been made.

push upload your code to the online repository.

pull request a request to the repository owner to merge your branch (copy) with the master copy. Github offers a discussion board for each pull request  
merge final merging of your branch (copy) with the master branch.

**Basic tools**

GitHub online repository (remote)  
Git Shell (local)  
Git Desktop (local)

Git Hub Shell is a command line, which is installed automatically together with GitHub Desktop. It is more handy for programmers. For non-programmers GitHub Desktop is intended.

**Github workflow**

|  |
| --- |
| **GIT Desktop** |
|  |
| 1. Create an account on Git Hub 2. Login in GitHub online, browse for “phoenix-mossimo” and fork the repository (project) “BTS-Manual”.  3. Download and install Github Desktop (the download in BBAW network is problematic; I use FU VPN and “Run as administrator. |
| 4. Login in Github Desktop with the User account created in Github Online.  5.In Github Desktop “Options” type in “C:\Users\YOUR\_USERNAME\Documents\GitHub” in “Clone path”  6. Click on "+" sign (left top corner), click on "Clone". On the left panel your Github account name should be displayed and on the right the “forked” repository “BTS-Manual”. Select "BTS-Manual" and click “Clone”.  7. Confirm “C:\Users\YOUR\_USERNAME\Documents\GitHub” as a destination directory -> The repository is saved, the name is displayed on the menu on the left side of GitHub desktop. |
| 8. Create a new brunch in Github Desktop (ex. Max\_1) – the new brunch is now the current working branch. You can close Github desktop now. |
| 9. Start “Oxygen XML Editor”. Browse to “C:\Users\YOUR\_USERNAME\Documents\GitHub\BTS-Manual” and open “BTS.ditamap“. Confirm “Open with “Dita Maps manager”.  10. Work with the files. Save the modified / new files. |
| 11. At the end of the day open Github desktop and select “BTS-Manual” repository. On the central panel the modified files and the changes will be displayed.  12. Write a "Summary" and "Description" - and click on "Commit to ..." - the files will be committed to **your local** branch. “Summary” of the “Commit” should be the date. “Description” should be the description of the major changes made during the day.  31. At the end of the week - click on “Create pull request” – pull request will be sent to the server.  -> Admin will merge the pull request with the master branch. |
|  |
| ATTENTION: Do not switch to Master Branch on Github Desktop. –  -Local “Master” branch may contain the outdated data in case Admin did not perform “merge” yet.  -even if it is up-to date, it should be “synched” with the remote “master”.  - if it was synched, there is no way to synch it with the current working brunch – a new work brunch should be created.  -> theoretically every day should start with updating the local master branch and creating a work branch. The day should finish with the “commit” and “pull request”, whereas the branch is deleted by the admin on the server. |
|  |

**Basic workflow – short**

|  |  |
| --- | --- |
| **Step** | **Explanation** |
| 1. Start Guthub, switch to “master” branch, click “sync”. | The updated files are downloaded from the remote “master” onto the local “master” branch. Local “master” branch now has the up-to-date files with changes from other users |
| 2. Create a new branch. | The copy of the master branch is created under the given name. You will be working with the files of this branch, which later on will be “merged” back with the remote master. |
| 3. Now open “Oxygen” and work with the files. | In case you forgot to do steps 1 and 2 and started working with Oxygen files first, after opening Github in the tab “changes” you will see all files, where changes have been made. Just create a copy of the current branch. If is preferable to create a separate branch for each work segment (ex. daily work) |
| 4. Create a “Commit” with summary and description | “Commits” changes to the local branch |
| 5. Create a “pull request” | Pull request sends your updated files to the remote server with a plea to merge them with master |
| 6. Go to the online repository, open the tab “Pull requests” and click merge | The files will be merged with the “master” branch |

**GUT Shell workflow**

|  |
| --- |
| **GIT Shell** |
| **STEP 1: create a repository** |
| 1. Create an account and a new repository (project) on Git Hub. The address of you repository will be smth like: <https://github.com/phoenix-mossimo/BTS-Manual.git>.  2. Install GitHub Desktop application. Git Shell comes with it. Provide your Git Hub account data (login and password) and Git Hub local path (the place where the projects will be locally stored, ex. :\Users\Max\Documents\GitHub) |
| ***git clone / git pull*** |
| 1. Start the git shell and type in in the command line: "git clone https://github.com/phoenix-mossimo/BTS-Manual.git" (the last is the http address of your project) -> as the result the project and all containing files will be copied to the local Github directory \*C:\Users\Max\Documents\GitHub).  2. If the project existed already, you can download the files, which were created in the online repository, to the local PC. For that in Git Shell go to the repository directory (cd...) and type in "git pull". |
| **STEP 2: make a new branch for your code** |
| ***git branch / git checkout*** |
| 1. "git branch feature1" : creates the copy of the master branch called "feature1"  2. "git checkout feature 1": switches to "feature1" branch. From now on all changes made to files will be made in this branch. Branch master will remain unaffected. |
| **STEP 3: Code a new feature, commit it, check “mergability”** |
| ***git status / git add -a (or “.”) / git commit*** |
| 1. Copy-paste new files / modify existing files in the repository. Check the contents of the repository with Git Shell: go into it with "cd bts-manual" and type "git status". The new files are displayed in red - they are not yet added to the repository.  Remember: you are working on your own “feature 1” branch.  2. Now **Add** the file with "git add index.html". If you do "git status", you will see that the color of the file changed to green. "git add -A" will add all files.  3. Now **Commit** files, describing the changes: "git commit -m "I added an index.html file”. |
|  |
| ***git merge*** |
| 1. "git merge master": checks the "mergability" of "feature" branch with "master branch". If there is a conflict - Git shell will display it. Moreover in the file the lines, which affected the conflict, will be underlined too. |
| **STEP 4: Push files up, make a pull request** |
| ***git push ( git push –set-upstream origin branchname)*** |
| 1. Now **Push** files onto the repository with "git push" -> The file is uploaded to the online repository. "git push --set-upstream origin feature1" will create a new branch “feature1” in the online repository.  2. Online click on "Compare & pull request" -> will display if two branches can be merged.  3. Click “Create new pull request” (or smth similar).The new pull request will be generated (message: phoenix-mossimo wants to merge 1 commit into "master" from "max\_local").  4. If more changes necessary, in the "Pull requests" tab / "Conversation" tab, write a comment. |
| **STEP 5: merge with master branch** |
| 1. Click "Merge pull request" & "Confirm merge" & delete "Feature 1 branch" |
|  |
| **Additional** |
| - ls : display the contents of the directory  - "esc": wq  - if changes were done to the file online , which I also changed locally, git will produce the following messages:  a) file changed locally but not commited: error, local changes will be overwritten, commit your changes  b) file committed locally: "merge conflict". PLUS in the local file the lines, which produced the conflict, will be displayed". |