Laborationsrapport

Moment 1 – Versionshantering & GIT
VT2022 DT093 Datateknik GR (B), PHP, 7,5 hp (distans)

Författare: Natalie Salomons Frick, nasa2104@student.miun.se
Termin, år: HT, 2022

1 Sammanfattning

This report answers specific questions regarding the usage of GIT.

Specifically, it contains explanations and usage of the following terms:

- Workspace
- Staging area
- Local repository
- Remote repository
- Branch
- Merge

And commands:

- git config
- git init
- git status
- git add
- git commit
- git push
- git checkout
- git pull
- git merge
- git fetch
- git log

A special case regarding how to ignore files that should not be cataloged/indexed is also included.

2 Innehållsförteckning

3 Frågor

3.1 Förklara följande begrepp:

3.1.1 Workspace

The workspace is the set of directories that hold the files of the product/programs. It is the working area, sometimes called working tree. Consider a library, the workspace would be all the shelves containing the books [files][1].

3.1.2 Staging area

The staging area contains the files being worked on. It is a temporary storage of current files being worked on. So, back to the the library, a specific book(s) is taken down from the shelves over to a table and to be read. These files are assigned out of the workspace and updated. The files in the staging area are the files being worked on. GIT knows which files these are because either they have recently been committed to or have been checked out of the repository [1].

3.1.3 Local repository

The repository is the place which contains all the commits. It like a backup of your files held within the workspace/git directory stored safely away. Once changes have been made to file which the developer is happy with and want to keep, then that file is then saved to the repository. So, if you think of this library we are working in, we have the shelves of books, the tables for working and there is also a room with a door, also having shelves of book, duplicates of some of the books elsewhere in the library. This room is the repository. With every new version of a file, the developer can choose to place a copy in the repository, as a backup [1].

Local repository is the repository located on your computer. Only people with access to the computer has access to the files. Access to the internet is not required.

3.1.4 Remote repository

Remote repository is the repository located in an online- cloud (or server). Can be accessed from any computer. Files can be shared easily for collaboration. Access to the internet is required.

3.1.5 Branch

Branching allows several people to work on the same file at the same time but not overriding the others contribution. When the files are then saved again (committed/pushed) to the remote repository/directory, two versions are saved, along with the information of who made the changes and what changes (via the message provided by the person making the changes). These files then need to be merged, combining all the changes [2].

3.1.6 Merge

Merging files takes several versions of the same files and merge the documents, creating a single document with all the updates and changes within. If there is duplicate code or conflicting code, GIT will warn the user before merging the files. GIT can enable developers to verify changes of each “branch” for potential conflicts [3][2].

3.2 Beskriv följande kommandon (vad de gör, hur de fungerar etcetera):

3.2.1 git config

The command git-config allows the user to get and set options for the repository. With this command the user can search, set, replace, or unset different options. Below is the code with several options that can be used with the command [4].

```
git config     [<file-option>] [--type=<type>] [--fixed-value]
               [--show-origin] [--show-scope] [-z|--null] name
               [value [value-pattern]]
```

3.2.2 git init

The command git-init is used to create an empty GIT repository/directory with subdirectories for objects, heads and tags. It can also be used to reinitialize an existing one. Below is the code with several options that can be used with the command [5].

```
git init [-q | --quiet] [--bare] [--template=<template_directory>]
```

3.2.3 git status

This command allows the user to see the status of the working tree or a summary of files with changes that are ready to (or should) be added. Below is the code with several options that can be used with the command [6].

```
git status [<options>…] [--] [<pathspec>…]
```

3.2.4 git add

git add is the command used to add file contents to the index, i.e., the temporary storage of the staging area. To work on a file, it needs to be moved/added to the staging area. This is how git knows which files are being worked on. Below is the code with several options that can be used with the command. . For example, the command git add –-update will update the repository with only new versions of the files already existing and hop over any new files [7].

```
git add        [--edit | -e] [--update | -u]] [--refresh]
               [--ignore-errors] [--ignore-missing] [--renormalize]
```

3.2.5 git commit

Once the files to be added are specified [git add] the command git commit records these changes to the repository, i.e., saves or uploads new files (if specified) or replaces existing files with newer versions. Below is the code with several options that can be used with the command. If the command has been executed but immediately after it is regretted due to finding a mistake, the commit can be recovered by using the command git reset [8].

```
git commit     [-a | --interactive | --patch] [-s] [-v] [--dry-run]
               [-F <file> | -m <msg>] [--allow-empty-message]
               [--no-verify] [-e] [--author=<author>] [--date=<date>]
```

3.2.6 git push

The command git push updates remote repositories using the local repository. Below is the code with several options that can be used with the command [9].

```
git push [--all | --mirror | --tags] [--follow-tags]
          [--repo=<repository>] [-f | --force] [-d | --delete]
```

3.2.7 git checkout

The command git checkout one can check out, get, or download, files to the workspace and thus updating the existing version with the checked-out version. Below is the code with several options that can be used with the command. If working on a specific branch, then that can be specified in the checkout command [10].

```
git checkout [-q] [-f] [-m] [<branch>]
```

3.2.8 git pull

Two people are working on the same file, thus on different branches. Person 1 wants to get the changes person 2 made in their version of the file. Using the command git pull this is possible. The command gets a specific file from a specific branch (the given parameters) and then merges it with the specified file. Before this command is run, it is recommended to run a git commit command (to save a backup) just in case problems arise and a step

backwards needs to occur. Below is the code with several options that can be used with the command [11].

```
git pull [<options>] [<repository> [<refspec>…]]
```

3.2.9 git merge

The command git merge joins files together into one. It incorporates changes from the specified files/commits into the file in the current branch creating a new file which has all current code from the branches specified, including in the log a description of changes.

If a merge has been run, but conflicts arise the command git merge –abort will try to reconstruct the (committed/backed up) pre-merge state.

Below is the code with several options that can be used with the command [12].

```
git merge [-n] [--stat] [--commit]  [--no-commit] [--[no-]edit]
```

3.2.10 git fetch

git fetch allows downloading of objects and files from other repositories. Below is the code with several options that can be used with the command [13].

```
git fetch [<options>] [<repository> [<refspec>…]]
```

3.2.11 git log

All past changes, updates, commits, and messages are saved in a log following the parent branch. To view the log, use the command git log. The log is shown in reverse chronological order.

For example, the command:

```
$ git log foo bar ^baz
```

Can be phrased “list all commits reachable from foo or bar, but not baz”.

Below is the code with several options that can be used with the command [14].

```
git log [<options>] [<revision range>] [[--] <path>…]
```

3.3 Specialfall: beskriv hur går vi tillväga för att ignorera visa filer eller kataloger att indexeras med Git?

All files that are being worked on have been added to the staging area with the command git add. When changes are complete and there are new versions, these files needed to be added back into the repository. This is done by the command git commit, which basically says, save these versions as backup as a new version.

Git commit command automatically indexes all files in the staging area. To specify which files shall not be indexed or committed to the repository the create a file in the directory and name it ".gitignore". This file will hold all the files or catalogues that should be ignored during indexing. [15]

4 Slutsatser

Git is a useful tool to implement into the coding process. How many times have you worked and updated code in a project and everything is going well, until something happens and now nothing works as it should? And you don't know what happened. All you want to do is take two steps back to where everything was working. But you can't because those files have been overwritten during the saving process. So now you are stuck.

Using git is like using a backup system. All files get backed up periodically and then when you run into that problem, going two steps back is easy because all that is required is pulling those files.

It also helps when collaborating with others. Everyone can work on the same file simultaneously and then these files and changes can then be merged. Git even notes potential conflicts before the merge to be addressed.

It is a great system to incorporate into the project process.

5 Källförteckning

[1] Front End Masters. Working Area, Staging Area, Repository. https://frontendmasters.com/courses/git-in-depth/working-area-staging-area-repository/. Retrieved 2022-1-19

[2] Atlassian Bitbucket Tutorials. Learn Git. https://www.atlassian.com/git/tutorials/what-is-version-control. Retrieved 2022-1-19

[3] GIT. GIT basics Episode 2. https://git-scm.com/video/what-is-git. Watched 2022-1-19

[4] GIT. GIT documentation. "git-config". https://git-scm.com/docs/git-config. Retrieved 2022-1-19

[5] GIT. GIT documentation. "git-init". https://git-scm.com/docs/git-init. Retrieved 2022-1-19

[6] GIT. GIT documentation. "git-status". https://git-scm.com/docs/git-status. Retrieved 2022-1-19

[7] GIT. GIT documentation. "git-add". https://git-scm.com/docs/git-add. Retrieved 2022-1-19

[8] GIT. GIT documentation. "git-commit". https://git-scm.com/docs/git-commit. Retrieved 2022-1-19

[9] GIT. GIT documentation. "git-push". https://git-scm.com/docs/git-push. Retrieved 2022-1-19

[10] GIT. GIT documentation. "git-checkout". https://git-scm.com/docs/git-checkout. Retrieved 2022-1-19

[11] GIT. GIT documentation. "git-pull". https://git-scm.com/docs/git-pull. Retrieved 2022-1-19

[12] GIT. GIT documentation. "git-merge". https://git-scm.com/docs/git-merge. Retrieved 2022-1-19

[13] GIT. GIT documentation. "git-fetch". https://git-scm.com/docs/git-fetch. Retrieved 2022-1-19

[14] GIT. GIT documentation. "git-log". https://git-scm.com/docs/git-log. Retrieved 2022-1-19

[15] MIUN DT093G Datateknik GR (B) Moment 1 Course Video. "Git Version hantering & teoretisk introduktion" @17:00. Watched 2022-1-19 https://www.youtube.com/watch?v=dzY85kqvFaE&t=1257s.