**Level 1: init**

A new directory, git\_hug, has been created; initialize an empty repository in it.

cd git\_hug/

git init

**Level 2: config**

Set up your git name and email, this is important so that your commits can be identified.

git config ––global user.name „Tobias Kerst“

git config ––global user.email [moc.erocsbot@tsrek.saibot](mailto:moc.erocsbot@tsrek.saibot)

**Level 3: add**

here is a file in your folder called README, you should add it to your staging area Note: You start each level with a new repo. Don’t look for files from the previous one.

git add README

**Level 4: commit**

The README file has been added to your staging area, now commit it.

git commit -m ‚Initial commit‘

**Level 5: clone**

Clone the repository at <https://github.com/Gazler/cloneme>.

git clone <https://github.com/Gazler/cloneme>

**Level 6: clone\_to\_folder**

Clone the repository at <https://github.com/Gazler/cloneme> to my\_cloned\_repo.

git clone https://github.com/Gazler/cloneme my\_cloned\_repo

**Level 7: ignore**

The text editor ‘vim’ creates files ending in .swp (swap files) for all files that are currently open.We don’t want them creeping into the repository.Make this repository ignore .swp files.

vim .gitignore

Insert: \*.swp

**LeveL 8: include**

Notice a few files with the ‘.a’ extension.We want git to ignore all but the ‘lib.a’ file.

vim .gitignore

\*.a

!lib.a

**Level 9: status**

There are some files in this repository, one of the files is untracked, which file is it?

git status

You now have to simply insert the name of the untracked file: database.yml

**Level 10: number\_of\_files\_commited**

There are some files in this repository, how many of the files will be committed?

git status

Look, at those files, that are ready to be committed. Those are rubyfile1.rb and rubyfile4.rb

**Level 11: rm**

A file has been removed from the working tree, however the file was not removed from the repository.Find out what this file was and remove it.

git status

git rm deleteme.rb

**Level 12: rm\_cached**

A file has accidentally been added to your staging area, find out which file and remove it from the staging area.NOTE Do not remove the file from the file system, only from git.

git status

git rm ––cached deleteme.rb

**Level 13: stash**

You’ve made some changes and want to work on them later. You should save them, but don’t commit them.

git stash

**Level 14: rename**

We have a file called oldfile.txt. We want to rename it to newfile.txt and stage this change.

git mv oldfile.txt newfile.txt

**Level 15: restructure**

You added some files to your repository, but now realize that your project needs to be restructured.Make a new folder named src, and move all of the .html files into this folder.

mkdir src

git mv \*.html src/

**Level 16: log**

You will be asked for the hash of most recent commit.You will need to investigate the logs of the repository for this.

git log ––oneline

Enter the hash of the last commit that will be printed. For a nicer output, I chose the option ––oneline.

**Level 17: tag**

We have a git repo and we want to tag the current commit with new\_tag.

git tag new\_tag

**Level 18: push\_tags**

There are tags in the repository that aren’t pushed into remote repository. Push them now.

git push ––tags

**Level 19: commit\_amend**

The README file has been committed, but it looks like the file forgotten\_file.rb was missing from the commit.Add the file and amend your previous commit to include it.

git add forgotten\_file.rb

git commit ––amend

**Level 20: commit\_in\_future**

Commit your changes with the future date (e.g. tomorrow).

git commit ––date=25.10.2015T22:23:23

In addition, the date part is accepted in the following formats: YYYY.MM.DD, MM/DD/YYYY and DD.MM.YYYY.

**Level 21: reset**

There are two files to be committed.The goal was to add each file as a separate commit, however both were added by accident.Unstage the file to\_commit\_second.rb using the reset command (don’t commit anything).

git reset HEAD to\_commit\_second.rb

**Level 22: reset\_soft**

You committed too soon. Now you want to undo the last commit, while keeping the index.

git reset ––soft HEAD^

**Level 23: checkout\_file**

A file has been modified, but you don’t want to keep the modification.Checkout the config.rb file from the last commit.

git checkout –– config.rb

**Level 24: remote**

This project has a remote repository.Identify it.

git remote

**Level 25: remote\_url**

The remote repositories have a url associated to them.Please enter the url of remote\_location.

git remote -v

Copy the url of the remote repo

**Level 26: pull**

You need to pull changes from your origin repository.

git pull origin master

**Level 27: remote\_add**

Add a remote repository called origin with the url <https://github.com/githug/githug>

git remote add origin <https://github.com/githug/githug>

**Level 28: push**

Your local master branch has diverged from the remote origin/master branch. Rebase your commit onto origin/master and push it to remote.

git rebase origin/master master

git push origin master

**Level 29: diff**

There have been modifications to the app.rb file since your last commit.Find out which line has changed.

git diff

Explanation: You get the output, starting at line 23, which displays “erb :success”. The changes three lines below, so the changes are in line 26.

**Level 30: blame**

Someone has put a password inside the file config.rb find out who it was.

git blame `config.rb`

**Level 31: branch**

You want to work on a piece of code that has the potential to break things, create the branch test\_code.

git branch test\_code

**Level 32: checkout**

Create and switch to a new branch called my\_branch.You will need to create a branch like you did in the previous level.

git checkout -b my\_branch

**Level 33: checkout\_tag**

You need to fix a bug in the version 1.2 of your app. Checkout the tag v1.2.

git checkout v1.2

**Level 34: checkout\_tag\_over\_branch**

You need to fix a bug in the version 1.2 of your app. Checkout the tag v1.2 (Note: There is also a branch named v1.2).

git checkout tags/v1.2

**Level 35: branch\_at**

You forgot to branch at the previous commit and made a commit on top of it. Create branch test\_branch at the commit before the last.

git branch test\_branch HEAD^

**Level 36: delete\_branch**

You have created too many branches for your project. There is an old branch in your repo called ‘delete\_me’, you should delete it.

git branch -d delete\_me

**Level 37: push\_branch**

You’ve made some changes to a local branch and want to share it, but aren’t yet ready to merge it with the ‘master’ branch.Push only ‘test\_branch’ to the remote repository

git push origin test\_branch

**Level 38: merge**

We have a file in the branch ‘feature’; Let’s merge it to the master branch.

git branch

git merge feature

I like check which branch I’m on by using git branch before I merge changes is. This helped me a lot in the past and I recommend you to do the same, in case you merge into the wrong branch.

**Level 39: fetch**

Looks like a new branch was pushed into our remote repository. Get the changes without merging them with the local repository.

git fetch origin

**Level 40: repack**

Optimise how your repository is packaged ensuring that redundant packs are removed.

git repack -d