# Git Tutorial

Kristof Werling

April 2022

# What is this training about

- Focus is on Git commands as used from the command line
- There are many different UI-clients available. By knowing the command line it will be easy to use anyone of them.

# Installation

# How to install Git locally on your computer

- Browse to: https://git-scm.com/downloads
- Follow the instructions for your OS
- Make sure that the GIT commands can be used from the command line

# Set up

# What to configure with a newly installed Git client

- There are two configuration steps absolutely necessary:
  - Configuration of the user name
  - Configuration of the user e-mail address
- If solely working with the command line configuring the editor to use is also beneficial

# User name and e-mail address

- User name

  git config --global user.name “<<Real name of user>>“

- E-Mail address

  git config --global user.email “somebody@computer.com“

- Editor

  git config –global core.editor “code --wait“

# Local, global, system

There are three levels on which the configuration can be made:

| Level | |
|---|---|
| system | Valid for all of the computer |
| global | Valid for the User |
| local | Valid for the repository |

This means, that one can use one e-mail address on the user level, but another one in one specific repository.

# Initialize a local repository

# Create a local repository

- To create a local repository one needs to run the following command in a directory:

```
git init
```

- During creation Git creates a (hidden) directory by the name of “.git “
- Now you have an an empty repository

# Repositories in Git

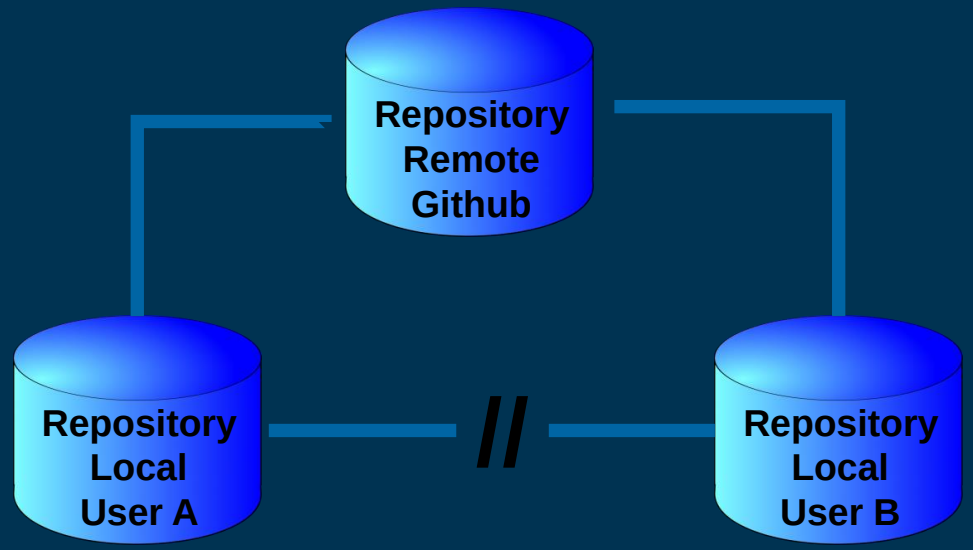
# Relationship between repositories

- Each user has a full local copy of the repository they work with
- There is no special repository, like a master or central repository
- Often, the remote repository (on GitHub) is declared to be the master every user is syncing with
- If repositories are synced then all of them hold the same information

# Relationship between repositories

**Usually it works like this:**

- If all three repositories are synced then they contain the same information – all of the history of the repository
- The remote repository is considered to be the Master by agreement. In theory, any other repository could be in that role, too
- User A and User B will sync with and from the remote repository and not have any internaction beween them directly.



# Exercises with local repositories

# Preparation

## Requirements for the exercises

- Have Git installed on your computer in the way that it can be used from the command line
- Have your name and e-mail address configured, preferably on the global level.
- Create a directory named “TestRep01“ any location you like (and have write access to)
- Go into the directory and initialize a git repository.

# Preparation

## Create a file

- In the TestRep01 directory:
  - Create a text file with the ending of the name t1.txt
  - Write the following text into the file:

    This is the initial text I am going to check into the Git repository

# Committing a file to the repository

**Steps to take to commit content to a repository**

1. First the modified, added, or deleted files need to be added to the Staging area.

   This can be done  multiple times, if needed

   As long as the files are still in the Staging area they can be removed from there if wished.

2. If satisfied the files in the Staging area can be committed into the repository.

# Committing a file to the repository

**Commands to commit content to a repository**

1. Adding files into Staging:

   git add –all

   (There are more specific options, but we will go with –all for the time being)

2. Finally commit the files in Staging into the repository:

   git commit –a –m “<<Comment>>“

```
Git tutorial > dir
 Volume in drive E is Daten
 Volume Serial Number is 91CD-DE50

 Directory of E:\Data\Dev\HHZ_Java_04_2022\GIT\TestRep01

15.05.2022  20:41    <DIR>          .
15.05.2022  20:41    <DIR>          ..
15.05.2022  20:41                68 t1.txt
               1 File(s)             68 bytes
               2 Dir(s)  231.920.046.080 bytes free

Git tutorial > git add --all

Git tutorial > git commit -a -m "Initial checking"
[master (root-commit) 36dca86] Initial checking
 1 file changed, 1 insertion(+)
 create mode 100644 t1.txt

Git tutorial >
```

# Committing a file to the repository

**Commands to commit content to a repository**

- If ever a files was added to Staging and schould not be contained in the commit use the git reset command:

  git reset <<file name>>

- Git status helps to understand what Staging looks like and what will be committed

- **CAREFUL**: git restore eliminates the changes in the file and restores the version of it in the repository

```
Git tutorial > notepad t1.txt

Git tutorial > git add --all

Git tutorial > git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   t1.txt


Git tutorial > git reset t1.txt
Unstaged changes after reset:
M       t1.txt

Git tutorial > git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   t1.txt

no changes added to commit (use "git add" and/or "git commit -a")

Git tutorial > git restore t1.txt

Git tutorial > git status
On branch master
nothing to commit, working tree clean
```

# Committing a file to the repository

**Commands to commit content to a repository**

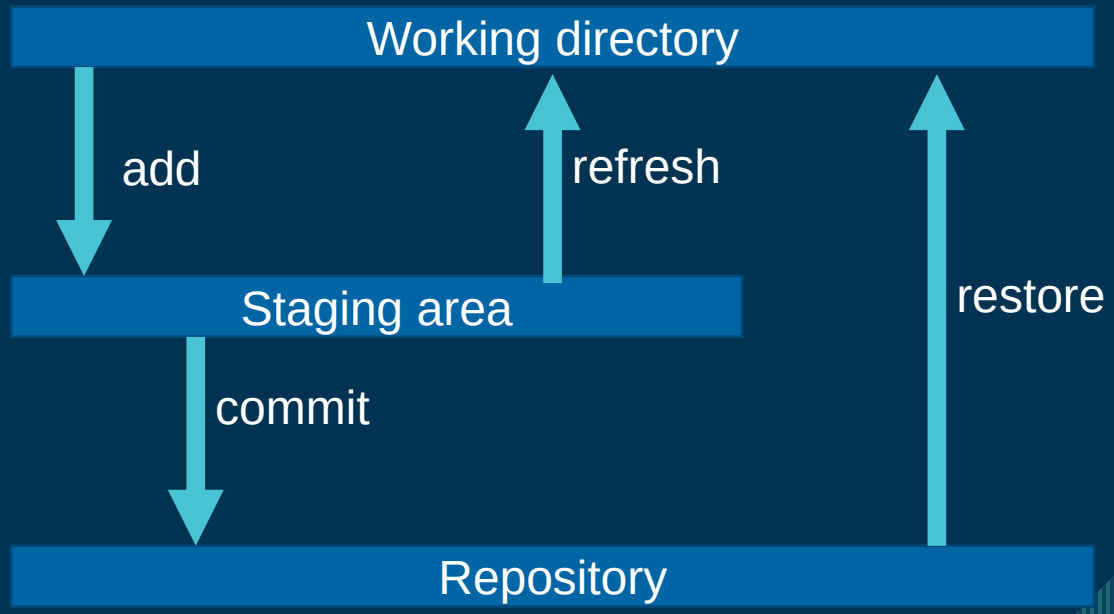
- Modify t1.txt: Change the word initial to modified.
- Run git status
- Run git add –all
- Run git status
- Run git commit
- Run git status

```
Git tutorial > git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   t1.txt

no changes added to commit (use "git add" and/or "git commit -a")

Git tutorial > git add --all

Git tutorial > git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   t1.txt


Git tutorial > git commit -a -m "Second commit"
[master af95de2] Second commit
 1 file changed, 1 insertion(+), 1 deletion(-)

Git tutorial > git status
On branch master
nothing to commit, working tree clean

Git tutorial >
```

# Committing a file to the repository

## Committing content

- Committing happens in 2 phases
- As long as the content is in staging, then it can be taken from the commit
- If changes to a local file should be reverted then the last committed version can be restored



# Committing a file to the repository

## Preventing the commit of files -- .gitignore

- The .gitignore file in the working directory allows for specifying the files, which never should be checked in ( such as the class files resulting from the java source files)

1. Create a .gitignore file with this content:

   *.xls*

2. Create an Excel file in the working directory

3. Run git add and then git status

```
Git tutorial > dir
 Volume in drive E is Daten
 Volume Serial Number is 91CD-DE50

 Directory of E:\Data\Dev\HHZ_Java_04_2022\GIT\TestRep01

15.05.2022  21:26    <DIR>          .
15.05.2022  21:26    <DIR>          ..
15.05.2022  21:23                 6 .gitignore
15.05.2022  21:22                 0 MyCalc.xlsx
15.05.2022  21:26                69 t1.txt
               3 File(s)             75 bytes
               2 Dir(s)  231.919.849.472 bytes free

Git tutorial > git add --all

Git tutorial > git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore


Git tutorial >
```

# Committing a file to the repository

## What changed?

- How to see what was changed between the files in the working directory and the repository?

  Git diff

1. Modify the text in t1.txt.
2. Run git diff

```
Git tutorial > git diff
diff --git a/t1.txt b/t1.txt
index 7f4009d..c8389c5 100644
--- a/t1.txt
+++ b/t1.txt
@@ -1 +1 @@
-This is the modified text I am going to check into the Git repository
\ No newline at end of file
+This is the again modified text I am going to check into the Git repository
\ No newline at end of file
```

# Working with branches

## Create a branch and switch to it

- To create a brach of the actual repository use

  git branch <<name of branch>>

1. Run git branch –a
2. Create a branch called B1
3. Run git branch -a

```
Git tutorial > git status
On branch master
nothing to commit, working tree clean

Git tutorial > git branch -a
* master

Git tutorial > git branch B1

Git tutorial > git branch -a
  B1
* master

Git tutorial >
```

# Working with branches

## Create a branch and switch to it

- In order to switch to branch B1 use the git checkout command:

  git checkout –b <<branch name>>

1. Run git branch –a
2. Run  git checkout –b B1
3. Run git branch -a

```
Git tutorial > git branch -a
  B1
* master

Git tutorial > git checkout -b B!
Switched to a new branch 'B!'

Git tutorial > git branch -a
* B!
  B1
  master

Git tutorial >
```

# Working with branches

## Working in a branch

1. Make sure yóu are in branch B1
2. Modify the t1.txt file and save it
3. Commit the changes to the repository
4. Run git status
5. Change to the master branch
6. Check out the content of t1.txt

```
Git tutorial > git status
On branch B1
nothing to commit, working tree clean

Git tutorial > @rem       Modified t1.txt and saved it

Git tutorial > git commit -a -m "First commit in branch B1"
[B1 0ca08ee] First commit in branch B1
 1 file changed, 1 insertion(+), 1 deletion(-)

Git tutorial > git status
On branch B1
nothing to commit, working tree clean

Git tutorial > type t1.txt
This is the again modified text I am going to check into the Git repository
Git tutorial > git checkout master
Switched to branch 'master'

Git tutorial > type t1.txt
This is the modified text I am going to check into the Git repository
Git tutorial >
```

# Working with branches

## Merging a branch

- Getting the changes of a branch into the master: git merge

1. Run git branch –a
2. Run git merge B1
3. Run git branch –a
4. Verify the content of t1.txt

```
Git tutorial > git branch -a
  B!
  B1
* master

Git tutorial > git merge B1
Updating d817a91..0ca08ee
Fast-forward
 t1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Git tutorial > git branch -a
  B!
  B1
* master

Git tutorial > type t1.txt
This is the again modified text I am going to check into the Git repository
Git tutorial >
```

# Working with branches

## Merge conflicts

- A conflict occurs if a file was modified in a branch and the master
- Modify t1.txt ín the master branch and commit
- Modify t1.txt ín the B1 branch and commit
- Go back to the master branch
- Run git merge B1

```
Git tutorial > git status
On branch master
nothing to commit, working tree clean

Git tutorial > git commit -a -m "Modified t1 in Master"
[master 3080499] Modified t1 in Master
 1 file changed, 1 insertion(+), 1 deletion(-)

Git tutorial > git checkout B1
Switched to branch 'B1'

Git tutorial > git commit -a -m "Modified t1 in B1"
[B1 9be969d] Modified t1 in B1
 1 file changed, 1 insertion(+), 1 deletion(-)

Git tutorial > git checkout master
Switched to branch 'master'

Git tutorial > git merge B1
Auto-merging t1.txt
CONFLICT (content): Merge conflict in t1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

# Working with branches

## Resolving merge conflicts

- The conflicts are shown in the file (in text files) and it is on the user to decide which version to use.
- Once resolved the file needs saving and then the usual git add and git commit.

```
<<<<<<< HEAD
This is the again MASTER modified text I am going to check into the Git repository
=======
This is the again B1 modified text I am going to check into the Git repository
>>>>>>> B1
```

# More info

## Git show

- The git show command provides more info than the git log:

  git show

```
Git tutorial > git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   t1.txt

no changes added to commit (use "git add" and/or "git commit -a")

Git tutorial > git log
commit 308049947893625225eb32611f361c4c32391379 (HEAD -> master)
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 22:08:04 2022 +0200

    Modified t1 in Master

commit 0ca08ee5bfb9b0b9b4000fde2f053680c83649b1
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 21:57:43 2022 +0200

    First commit in branch B1

commit d817a916ef6132bba69b06130c55d3285dbe1016 (B!)
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 21:47:18 2022 +0200

    Added -gitignore file

commit af95de2acc30edc70e7c7b644e9d6d1b4bdb6609
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 21:06:47 2022 +0200

    Second commit

commit 36dca86a6cddbbbc6d831dc2729ad7329e6a57b3
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 20:51:38 2022 +0200

    Initial checking

Git tutorial > git show
commit 308049947893625225eb32611f361c4c32391379 (HEAD -> master)
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 22:08:04 2022 +0200

    Modified t1 in Master

diff --git a/t1.txt b/t1.txt
index c8389c5..05a56f3 100644
--- a/t1.txt
+++ b/t1.txt
@@ -1 +1 @@
-This is the again modified text I am going to check into the Git repository
\ No newline at end of file
+This is the again MASTER modified text I am going to check into the Git repository
\ No newline at end of file
```

# More info

## Git show

- Even more details per commit can be retrieved with

  git show <<hash of commit>>

```
Git tutorial > git show d817a916ef6132bba69b06130c55d3285dbe1016
commit d817a916ef6132bba69b06130c55d3285dbe1016 (B!)
Author: kwerling <kwerling@gmail.com>
Date:   Sun May 15 21:47:18 2022 +0200

    Added -gitignore file

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..d35a3e4
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+*.xls*
\ No newline at end of file
```