

# Some GIT Basics

---

Dr. Uwe Ziegenhagen

September 28, 2022

[www.uweziegenhagen.de](http://www.uweziegenhagen.de)

# Content

---

Introduction

MinGW Basics

Git

# Introduction

---

# Introduction

- “Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. ”
- Version control systems track changes to files and allow you to go back to earlier versions thus creating backups as well.
- Git is not the first or only available version control system (VCS): CVS, Bitbucket, and Subversion are wellknown
- Git was developed by the Linux creator Linus Torvalds to maintain the Linux kernel

# Centralized versus Distributed VCS

- Subversion is a centralized vcs, it uses a central server. Only this server has the full history of all files
- All developers get special snapshots from this server.
- Backing up the server is essential!
- Git is a distributed vcs, so all clients (developers) have the complete repository on their machines.
- I personally used Subversion for a long time (and still use it for some projects) but mostly have migrated to Github.
- Github = a central platform where I can put my projects, but not the “central server” like with Subversion

# Working with Git

In the following we will look at various use cases for working with Git

- Create new repositories<sup>1</sup>
- Add files to the repository
- Making changes to the repository
- 

Remark: Git can be quite complex, but normally you need only a few commands.

---

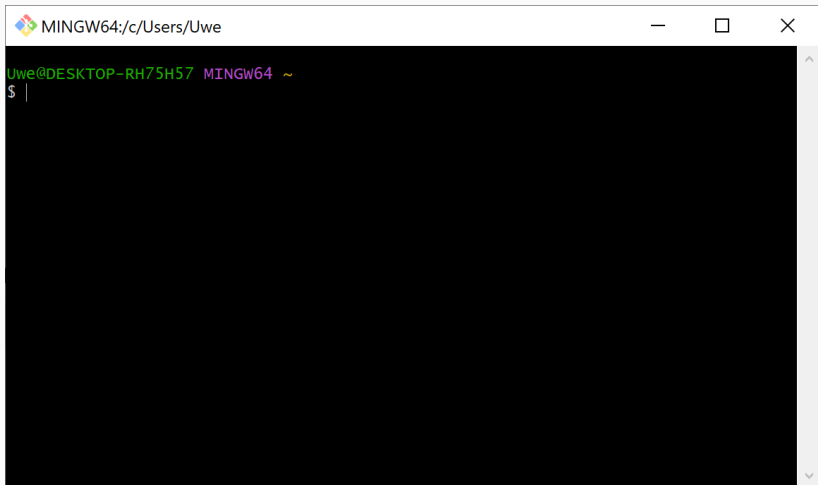
<sup>1</sup>The project structure you manage with Git

# MinGW Basics

---

# Running Git

- You find Git 2.28 on your desktop
- When you start it you land here:

A screenshot of a MINGW64 terminal window. The title bar at the top shows the MINGW64 logo and the path "MINGW64:/c/Users/Uwe". The terminal area has a black background with green text. The prompt "Uwe@DESKTOP-RH75H57 MINGW64 ~" is displayed, followed by a dollar sign "\$" and a vertical cursor bar "|".

```
MINGW64:/c/Users/Uwe
Uwe@DESKTOP-RH75H57 MINGW64 ~
$ |
```



- MinGW = Minimal GNU<sup>2</sup> for Windows
- A shell that ports many Unix/Linux tools to Windows
- This is not Git, Git is just a commandline tool that can be used within MinGW
- It contains a few Linux tools as well
- To move in this MinGW environment you need to use Linux commands

---

<sup>2</sup>“GNU is not Unix” = Open-Source stuff

# Basic MinGW commands

**pwd** In which directory are we?

**ls** List all files and folders

**cd** go to some specific directory

**mkdir** create a new directory

Remarks:

- There are no drive letters in MinGW
- / is the root directory
- Windows drive letters are (invisible) directories below this root directory
- so `cd /c` takes you to the `C:\` directory

# Git

---

# Create new Repositories

Create a directory, change to that directory and init the repository. The directory may already contains some files

```
cd /e # go to the e: drive

mkdir myfirstgitrepo # create empty directory

cd myfirstgitrepo # go to the directory

git init . # create repo (with a 'master' branch)
```

# git status

Use `git status` whenever you want to know something about the current state of the repository

```
Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use
"git add" to track)
```

# Adding files to the Repository 1

```
$ touch README.MD # creates an empty file
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will  
be committed)
```

```
    README.MD
```

```
nothing added to commit but untracked files  
present (use "git add" to track)
```

# Adding files to the Repository 2

```
$ git add README.MD # add file to staging area

$ git add -A # add all files to staging area
# not added to repository

$ git reset # remove everything from the
# staging area

$ git commit -m "My message" # Don't forget!!!
```

# Adding files to the Repository 3

```
$ git commit -m "Initial commit"
```

```
Author identity unknown
```

```
*** Please tell me who you are.
```

Run

```
git config --global user.email "you@examp.de"
```

```
git config --global user.name "Your Name"
```

to set your account's default identity.

Omit `--global` to set the identity only in this repository.

```
fatal: unable to auto-detect email address (got 'Uwe@DESKTOP-RH75H57.(none)')
```



# Adding files to the Repository 4

```
Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ git config --global user.email "ziegenhagen@gmail.com"

Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ git config --global user.name "Uwe Ziegenhagen"

Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ git commit -m "Initial commit"
[master (root-commit) acb9d75] Initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.MD

Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Now we have a file under version control! Yippie!

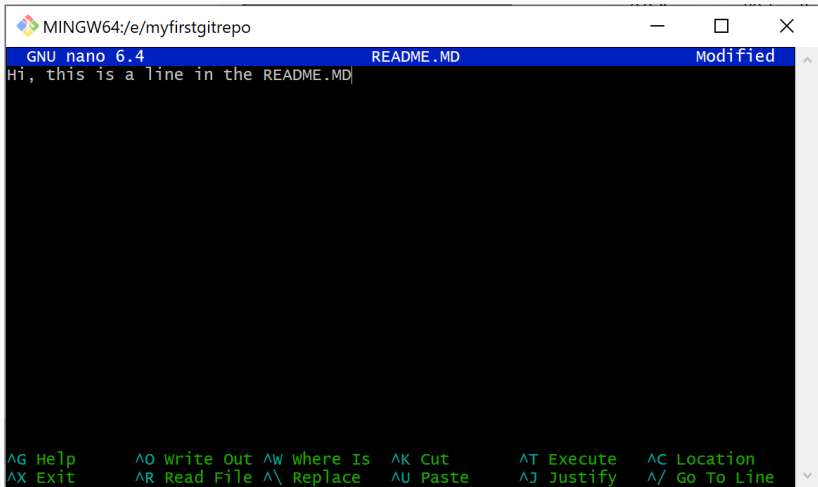
# A Word of Warning!

- When you omit the commit message, Git takes you to vim (VI “improved”) to allow you to enter it
- VIM = very powerful editor with strange user interface
- VIM uses special modes and is (almost) keyboard-only

Let's edit our file and commit it...

You can use e. g. `nano` to edit the file.

# Editing the file with nano README.MD



```
MINGW64:/e/myfirstgitrepo
GNU nano 6.4      README.MD      Modified
Hi, this is a line in the README.MD

^G Help    ^O Write Out ^W Where Is ^K Cut     ^T Execute ^C Location
^X Exit    ^R Read File ^\ Replace  ^U Paste   ^J Justify ^_/ Go To Line
```

- The circumflex means the Ctrl-key
- So Ctrl-O saves the file, Ctrl-X exits nano

# Escape from VIM Hell... – Part 1

```
Uwe@DESKTOP MINGW64 /e/myfirstgitrepo (master)
$ 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:   README.MD

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

Git notices that we changed a file, that is under version control

# Escape from VIM Hell... – Part 2

- We add the file to the commit stage
- you can ignore the LF warning. It just means that nano used Unix-style line endings (`\n`) in the file, for the repository however Windows line endings (`\r\n`)

```
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git add README.MD
warning: in the working copy of 'README.MD', LF will be
replaced by CRLF the next time Git touches it
```

Remarks: `\n` means Line Feed-Character, `\r\n` means Carriage Return + Line Feed. Helpful to know when working with text files.

# Escape from VIM Hell... – Part 3

We briefly check the status

```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.MD
```

and commit it without specifying the message parameter

```
$ git commit
```

Which takes us to eternal pain, the VIM!!!

## Escape from VIM Hell... – Part 4

[illegible]

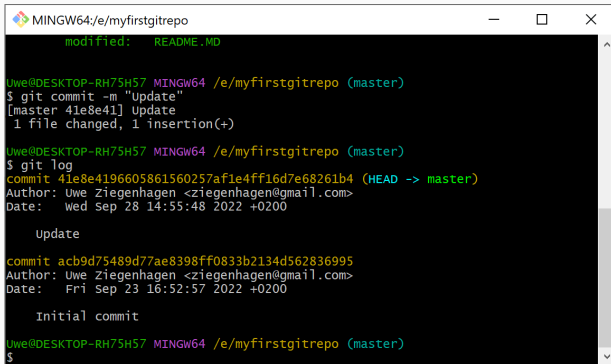
## Escape from VIM Hell... – Part 5

- ESC : q lets you exit without specifying a message, but you do not commit then.
- ESC : q ! lets you exit without specifying a message if you typed anything, but you do not commit then.



# Display the commit history with `git log`

- `git log` for the history of commits
- `git log -p` including the the full diff<sup>3</sup>

A screenshot of a terminal window titled 'MINGW64:/e/myfirstgitrepo'. The terminal shows a sequence of git commands and their outputs. First, 'git commit -m "Update"' is run, resulting in a commit with hash 41e8e41. Then, 'git log' is run, displaying the commit history. The output shows two commits: the first is the 'Update' commit (41e8e41) and the second is the 'Initial commit' (acb9d75). The terminal text is as follows:

```
MINGW64:/e/myfirstgitrepo
modified:  README.MD

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git commit -m "Update"
[master 41e8e41] Update
1 file changed, 1 insertion(+)

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log
commit 41e8e4196605861560257af1e4ff16d7e68261b4 (HEAD -> master)
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 14:55:48 2022 +0200

    Update

commit acb9d75489d77ae8398ff0833b2134d562836995
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Fri Sep 23 16:52:57 2022 +0200

    Initial commit

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$
```

<sup>3</sup>diff = differences between file in the `diff` format

# Example for git log -p

```
MINGW64:/e/myfirstgitrepo
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log -p
commit 41e8e4196605861560257af1e4ff16d7e68261b4 (HEAD -> master)
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 14:55:48 2022 +0200

    Update

diff --git a/README.MD b/README.MD
index e69de29..5f094a6 100644
--- a/README.MD
+++ b/README.MD
@@ -0,0 +1 @@
+Hi, this is a line in the README.MD

commit acb9d75489d77ae8398ff0833b2134d562836995
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Fri Sep 23 16:52:57 2022 +0200

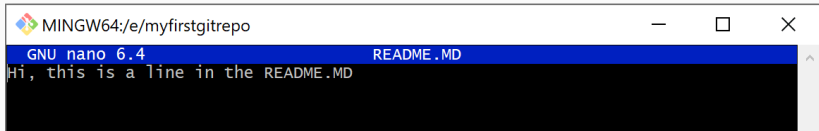
    Initial commit

diff --git a/README.MD b/README.MD
new file mode 100644
index 0000000..e69de29

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ |
```

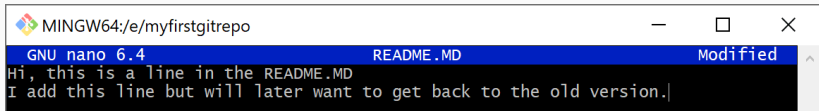
# Get back to earlier versions 1

- Let us assume we have this line in the old file



A screenshot of a terminal window titled "MINGW64:/e/myfirstgitrepo". The window shows the GNU nano 6.4 editor editing the file README.MD. The editor's status bar at the top indicates "GNU nano 6.4" and "README.MD". The main text area contains a single line: "Hi, this is a line in the README.MD".

and have changed it (and added it to the staging area and committed it)



A screenshot of the same terminal window, showing the GNU nano 6.4 editor after modifications. The status bar now includes "Modified" on the right. The main text area contains two lines: "Hi, this is a line in the README.MD" and "I add this line but will later want to get back to the old version.".

# Get back to earlier versions 2

```
MINGW64:/e/myfirstgitrepo
$ git commit -m "Added another line"
[master 11467ef] Added another line
1 file changed, 1 insertion(+)

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log -p
commit 11467efa3aaa09c2b28c1481ee6f32307b7cf867 (HEAD -> master)
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 15:09:00 2022 +0200

    Added another line

diff --git a/README.md b/README.md
index 5f094a6..03c879b 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
-Hi, this is a line in the README.md
+I add this line but will later want to get back to the old version.

commit 41e8e4196605861560257af1e4ff16d7e68261b4
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 14:55:48 2022 +0200

    Update

diff --git a/README.md b/README.md
index e69de29..5f094a6 100644
--- a/README.md
+++ b/README.md
@@ -0,0 +1 @@
+Hi, this is a line in the README.md

commit acb9d75489d77ae8398ff0833b2134d562836995
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Fri Sep 23 16:52:57 2022 +0200

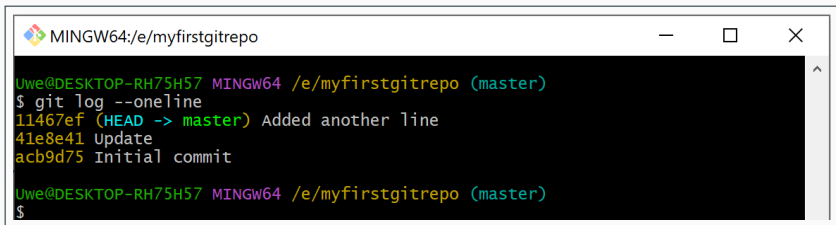
    Initial commit

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..e69de29

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$
```

# Get back to earlier versions 3


To which version you want to go?

A screenshot of a Windows terminal window titled 'MINGW64:/e/myfirstgitrepo'. The terminal shows the output of the command 'git log --oneline'. The output lists three commits: '11467ef (HEAD -> master) Added another line', '41e8e41 update', and 'acb9d75 Initial commit'. The prompt 'Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)' is visible at the top and bottom of the terminal output.

```
MINGW64:/e/myfirstgitrepo
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log --oneline
11467ef (HEAD -> master) Added another line
41e8e41 update
acb9d75 Initial commit
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$
```

- `git checkout 41e8e41 .`
- Do not forget the dot at the end, otherwise you end in detached head state, which you can/need to fix by `git checkout master`

# Get back to earlier versions 4

 MINGW64:/e/myfirstgitrepo

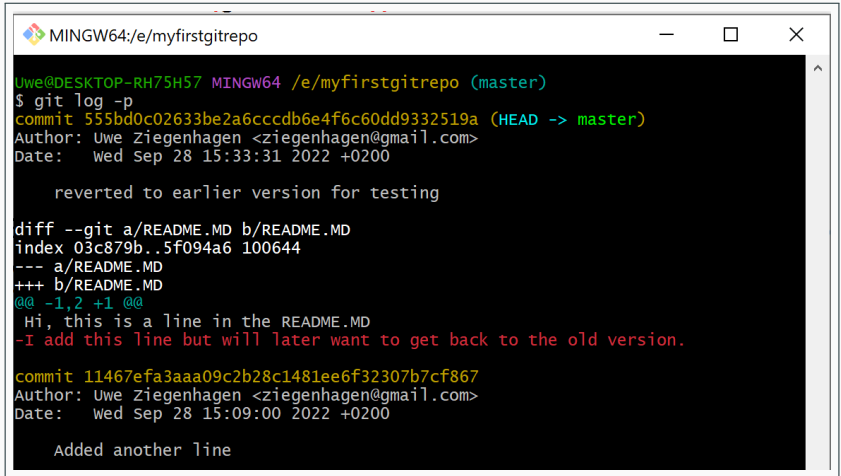
```
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log --oneline
11467ef (HEAD -> master) Added another line
41e8e41 Update
acb9d75 Initial commit
```

```
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git checkout 41e8e41 .
Updated 1 path from 72c45f4
```

```
Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.MD
```

Commit this version as well and note in the message why you reverted!

# Get back to earlier versions 5



```
MINGW64:/e/myfirstgitrepo

Uwe@DESKTOP-RH75H57 MINGW64 /e/myfirstgitrepo (master)
$ git log -p
commit 555bd0c02633be2a6cccd6e4f6c60dd9332519a (HEAD -> master)
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 15:33:31 2022 +0200

    reverted to earlier version for testing

diff --git a/README.MD b/README.MD
index 03c879b..5f094a6 100644
--- a/README.MD
+++ b/README.MD
@@ -1,2 +1 @@
 Hi, this is a line in the README.MD
-I add this line but will later want to get back to the old version.

commit 11467efa3aaa09c2b28c1481ee6f32307b7cf867
Author: Uwe Ziegenhagen <ziegenhagen@gmail.com>
Date:   Wed Sep 28 15:09:00 2022 +0200

    Added another line
```