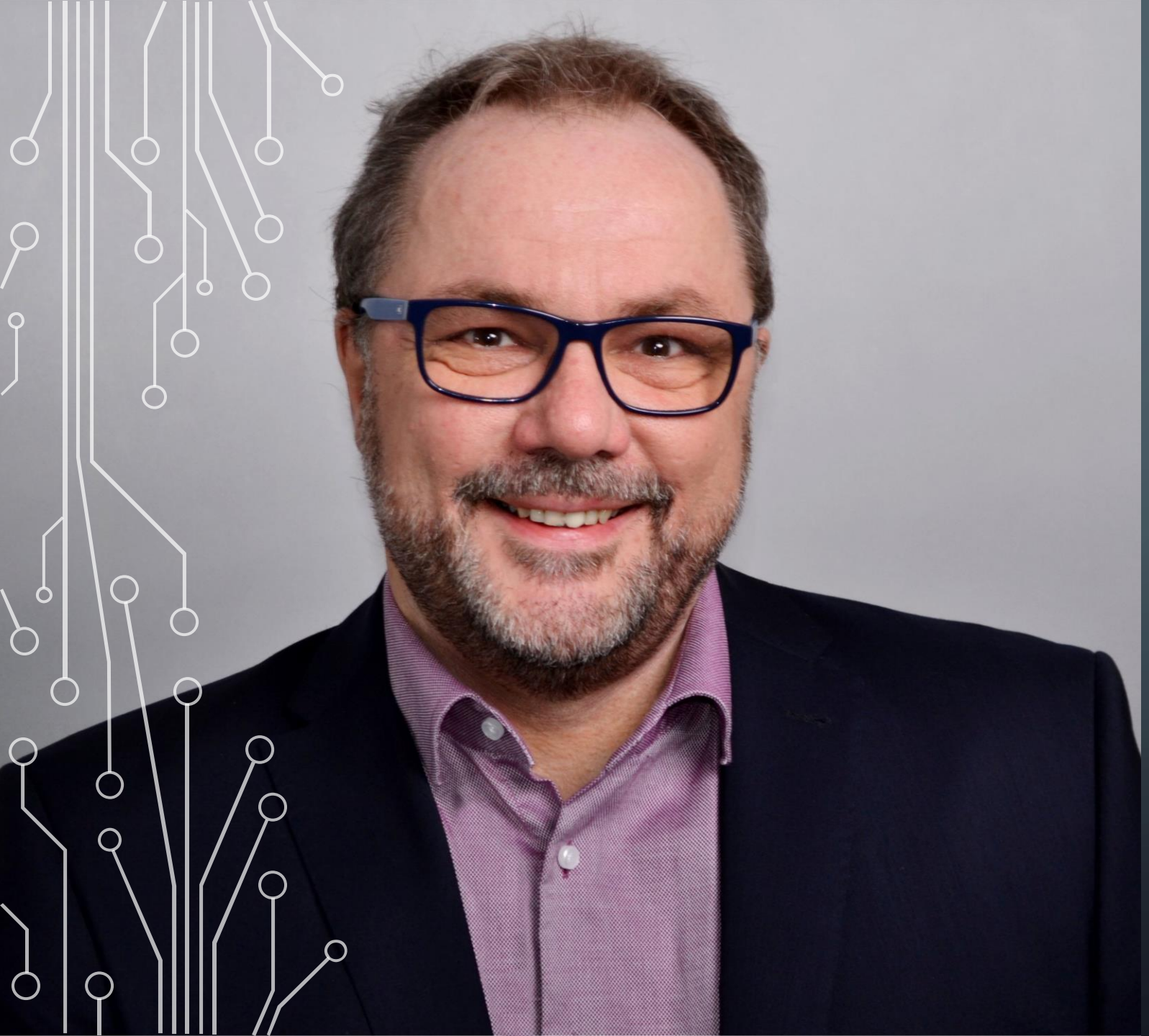
GIT TRAINING FOR IT PROFESSIONALS

KRISTOF WERLING
KWERLINGIT GMBH
1 SEPT 2023

WHO AM I

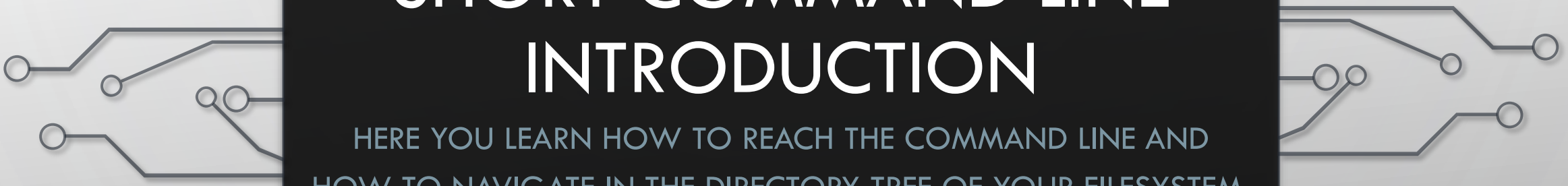
- Kristof Werling
- 31 years of experience at HP: DevOp, Developer, Architect
- Started KwerlingIT GmbH
- Focus: IT / Cyber Security
- Security Audits, IT consulting, Trainings, Software creation

KwerlingIT GmbH
kristof.werling@kwerlingit.com
https://www.linkedin.com/in/kristof-werling/
https://www.kwerlingit.com

TABLE OF CONTENT

DISCLAIMER

- The presentation was created on a MS-Windows-based Computer
- Screen-shots are made, and explanation are based in the MS-Windows environment
- Wherever the author is aware of a significant difference between MS-Windows and Macintosh operating environments the slides will point out these differences.
- The author uses Github.com for all demonstration purposes



SHORT COMMAND LINE INTRODUCTION

HERE YOU LEARN HOW TO REACH THE COMMAND LINE AND HOW TO NAVIGATE IN THE DIRECTORY TREE OF YOUR FILESYSTEM.

ACCESSING THE COMMAND LINE

MS-WINDOWS

1. Open the "Start" menu.
2. Type "cmd" or "Command Prompt" in the search bar and press Enter.
3. This opens the Command Prompt window in your user directory.

MACINTOSH

1. Open the "Applications" folder.
2. Go to the "Utilities" folder.
3. Launch the "Terminal" application.
4. This provides the shell command line in your home directory.

SHOW CONTENT OF A DIRECTORY

MS-WINDOWS

- Type “dir“ and press Enter

MACINTOSH

- Type “ls“ and press Enter
- In order to get the same level of detail as with the “dir“ command type “ls -l“

SHOW CONTENT OF A DIRECTORY

MS-WINDOWS

- There are two special directory names: "." and ".."
- They exist in every directory and are needed for the OS to finds its way in the filetree.
- How to make use of them is shown in the explanation for relative paths.

MACINTOSH

- Same as for MS-Windows

SHOW CONTENT OF A DIRECTORY

MS-WINDOWS

```
GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Users\Kwerling

03.09.2023  00:26    <DIR>          .
09.01.2023  18:37    <DIR>          ..
13.01.2023  02:12    <DIR>          .android
28.08.2023  16:30                53 .git-for-windows-updater
27.01.2023  13:50                59 .gitconfig
13.01.2023  02:12    <DIR>          .gradle
13.01.2023  02:15    <DIR>          .m2
03.09.2023  00:26                14 .minttyrc
09.01.2023  20:24    <DIR>          .vscode
09.01.2023  12:11    <DIR>          3D Objects
09.01.2023  20:47    <DIR>          AndroidStudioProjects
09.01.2023  18:55    <DIR>          Contacts
14.01.2023  22:35    <DIR>          Data
14.01.2023  22:35    <DIR>          Dev
14.01.2023  22:56    <DIR>          Documents
02.09.2023  23:02    <DIR>          Downloads
09.01.2023  18:55    <DIR>          Favorites
09.01.2023  18:55    <DIR>          Links
09.01.2023  18:55    <DIR>          Music
09.01.2023  18:57    <DIR>          OneDrive
30.08.2023  23:42    <DIR>          OneDrive - KwerlingIT GmbH
```

MACINTOSH

```
GIT Training > ls
'3D Objects'/
 AndroidStudioProjects/
 Anwendungsdaten@
 AppData/
 Contacts/
 Cookies@
 Data/
 Dev/
 Documents/
 Downloads/
 Druckumgebung@
'Eigene Dateien'@
 Favorites/
 IntelGraphicsProfiles/
 Links/
'Lokale Einstellungen'@
 Music/
 NTUSER.DAT
 NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf
```

WHERE AM I IN THE DIRECTORY TREE

MS-WINDOWS

- Type "cd" and press Enter
- In MS-Windows the directory names are separated by a backslash ("\")
- In MS-Windows Drives exist. Each drive has a single letter followed by a colon as name ("c:", "f:"). Each drive has its own root directory.

MACINTOSH

- Type "pwd" and press Enter
- In Unix and macOS the directory names are separated by a slash ("/")
- Unix and macOS only have one root directory and the complete directory tree us reachable from there.

WHERE AM I IN THE DIRECTORY TREE

MS-WINDOWS

```
GIT Training > cd
C:\Users\Kwerling
```

MACINTOSH

```
GIT Training > pwd
/c/Users/Kwerling
```

MOVING IN THE FILESYSTEM

ABSOLUTE PATH

- An absolute path specifies the exact location of a file or directory from the file system's root directory (the top-level directory).
- It begins with the root directory's name (e.g., C:\ in Windows or / in Unix-like systems) and provides a complete path to the target file or directory.
- Absolute paths are not dependent on the current working directory and can be used to locate a file or directory from anywhere in the file system.
- Examples of absolute paths:
  - **Windows**: C:\Users\YourUsername\Documents\file.txt
  - **macOS/Linux**: /home/YourUsername/Documents/file.txt

MOVING IN THE FILESYSTEM

RELATIVE PATH

- A relative path starts either with a directory name or one or two dots ("." or ".."). "." (dot) represents the current directory, and ".." (dot-dot) represents the parent directory.
- A relative path specifies the location of a file or directory with respect to the current working directory.
- Examples (in Unix notation with "/"):
  - **Reports/file1.txt:** file "file1.txt" in the subdirectory " **Reports** " of the current directory
  - **../ Reports /file1.txt:** file "file1.txt" in the subdirectory " **Reports** " of the parent directory

MOVING IN THE FILESYSTEM

MS-WINDOWS

- Type "cd <directory>", where <directory> is the directory, you want to change to, and then press Enter.
- Directories can be named absolute or relative.

MACINTOSH

- Same as for MS-Windows

CREATING NEW DIRECTORIES

MS-WINDOWS

- Type "mkdir <directory>", where <directory> is the directory you want to create, and then press Enter.
- Directories can be named absolute or relative.
- "mkdir" can be abbreviated as "md"

MACINTOSH

- Same as for MS-Windows, but the abbreviation of "md does not exist.

REMOVE / DELETE A DIRECTORIES

MS-WINDOWS

- Type "rmdir <directory>", where <directory> is the directory, you want to delete, and then press Enter.
- Directories can be named absolute or relative.
- The directory must be empty.
- "mkdir" can be abbreviated as "rd"

MACINTOSH

- Same as for MS-Windows, but the abbreviation of "rd does not exist.

EXERCISES

MS-WINDOWS

- Open a new Command Prompt window
- Switch to the "AppData\Local\Temp" directory
- The content of the directory will look different for each user, as it is used by the OS and many applications

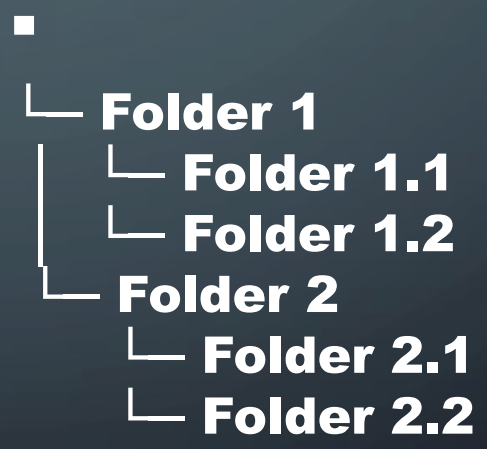
MACINTOSH

- Open a new Terminal window
- Switch to the "/tmp" directory
- The content of the directory will look different for each user, as it is used by the OS and many applications

EXERCISES

MS-WINDOWS AND MACINTOSH

- Build the directories and subdirectories as shown on the right-hand-side.



EXERCISES

MS-WINDOWS AND MACINTOSH

- Switch to directory "Folder 2"
- Delete directory "Folder 2.2"
- Goto to the parent directory
- Delete "Folder 2" --- What happens and how to delete the folder?
- Do the same for "Folder 1"
- Try to use absolute and relative path names for the directories you work with

INTRODUCTION INTO GIT

HOW TO WORK WITH GITHUB, GITLAB, BITBUCKET AND MANY OTHER PROVIDERS OF GIT REPOSITORIES

TECHNICAL ENVIRONMENT

- Git Repository Service Provider: Github
- IDE: IntelliJ
- Command line tools: standard git programs

TABLE OF CONTENT

WHY A VCS?

- Data Protection
  (prevent data loss / corruption)

- Collaboration
  (Multiple users can work on the same project in a controlled way)

- Auditing
  (tracking of changes on each file)

BASIC GIT CONCEPTS

- Repository (local or remote): Store for your files and their version history
- Commit: A new version of files stored to the repository version history.
- Branch: A forked specific version of the files in the repository
- Merge: Combining changes of one branch into another

INSTALLATION OF GIT

MS-WINDOWS

- Windows requires an installation

MACINTOSH

- There is a fair chance, that GIT is already installed on your machine.
- Verify it by opening a Terminal and type "git –version" followed by Enter
- If it is not installed see next slide

INSTALLATION OF GIT

- Goto: https://git-scm.com/downloads
- Depending on your OS follow the instructions on the page.
- Make sure that the git-command can be used from the command line
- There are also GUI Clients available from that page. No need to install any of them.

LOCAL CONFIGURATION

- It is necessary to configure your real name and email address:

- On the command line enter this:

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

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

LOCAL CONFIGURATION

- There are 3 levels of configuration hierarchy:

Level	Explanation
system	Taken, when there is no global or local configuration
global	Taken, when there is no local configuration
local	Taken, if it exists

LOCAL CONFIGURATION

- If wished, a standard editor (or IDE) can be configured for viewing or editing text and source files:

```
git config --global core.editor "idea"
```

CREATION OF A REPOSITORY

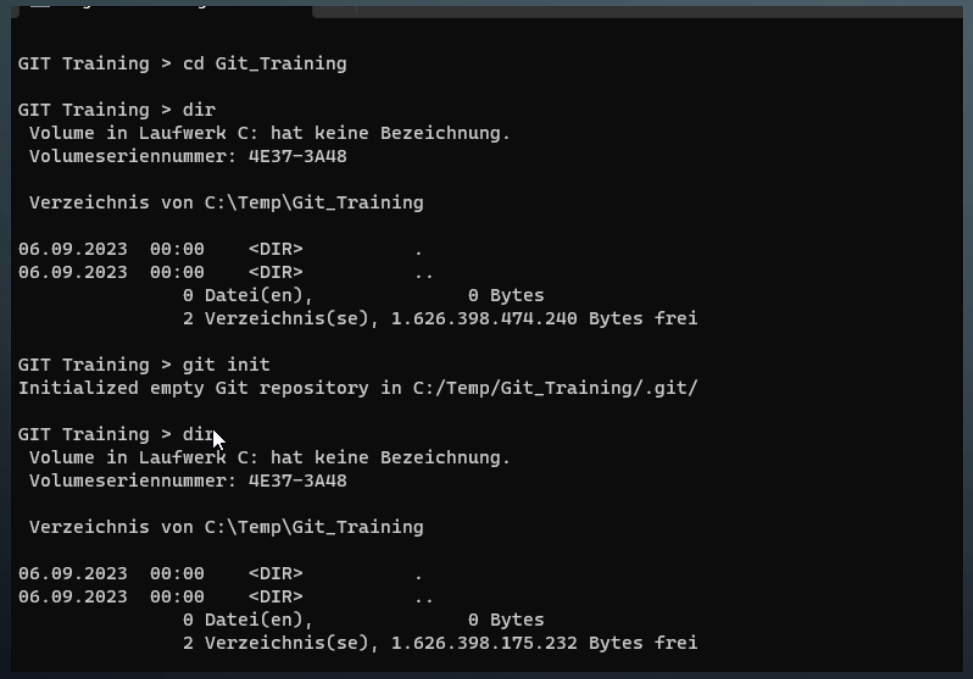
- "git init" create a new git repository in the directory it is executed in
- If there are files or directories in this directory, then these will not be added to the repository automatically.
- After execution there will be a new directory by the name of ".git". It stores the content of the repository as well as the needed information for the repository management.

EXCERCISE: CREATE A NEW LOCAL REPOSITORY

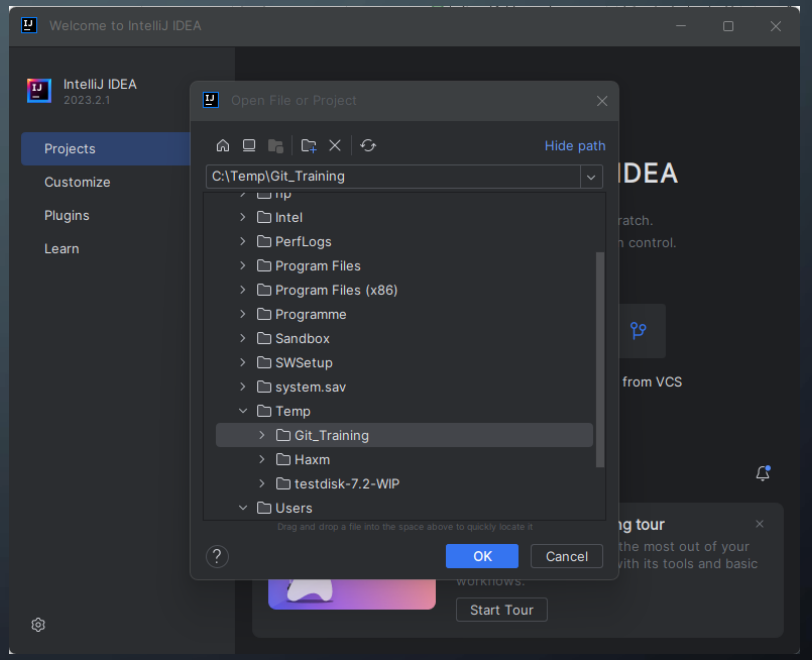
- Open a Terminal window
- Goto an empty directory
- Verify that it is empty
- Run the "git init" command
- Verify that the ".git" directory exists
- Have a look into that directory

EXCERCISE: CREATE A NEW LOCAL REPOSITORY

COMMAND PROMPT



INTELLIJ

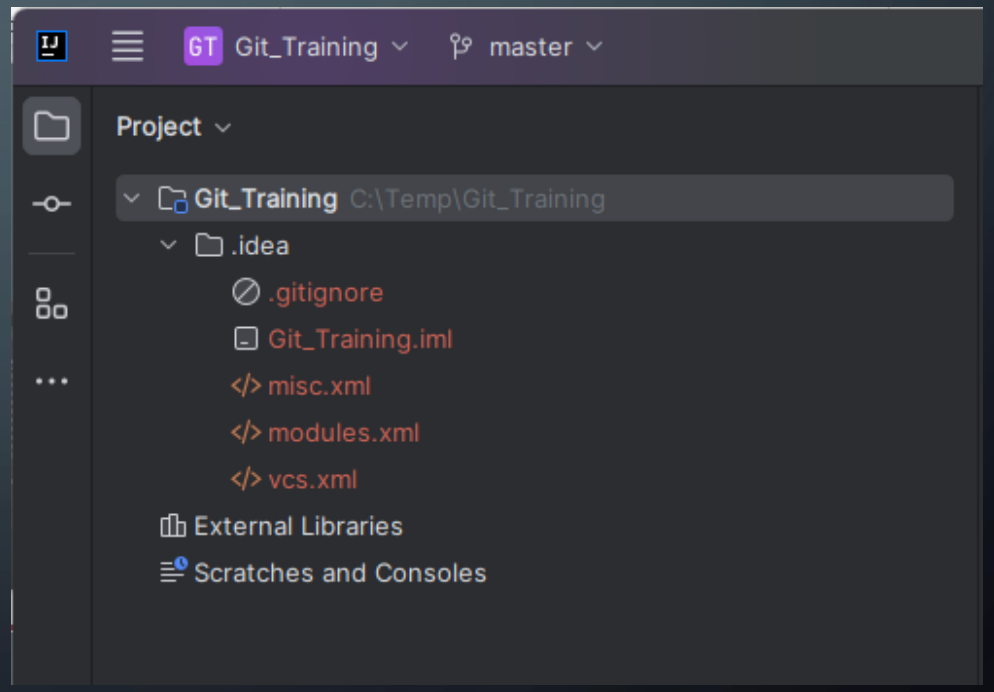


EXCERCISE: CREATE A NEW LOCAL REPOSITORY

COMMAND PROMPT

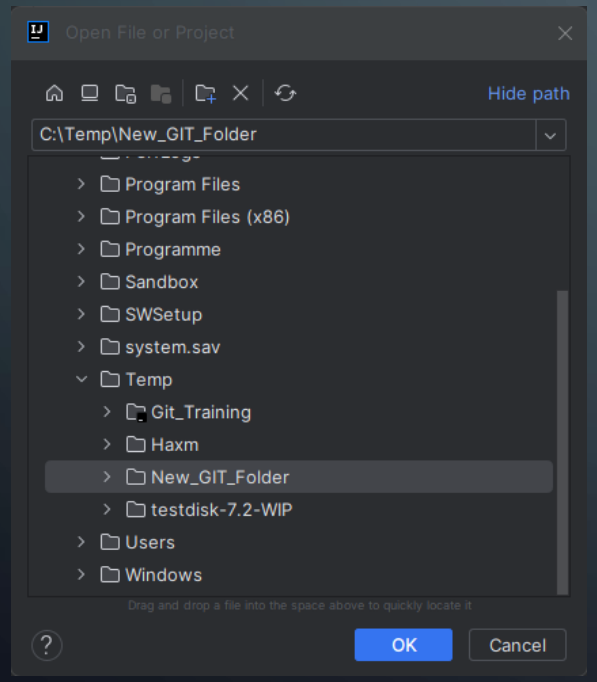
```
GIT Training > dir /AH
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

06.09.2023  00:00    <DIR>          .git
               0 Datei(en),              0 Bytes
               1 Verzeichnis(se), 1.626.402.197.504 Bytes frei

GIT Training > dir .git
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training\.git

06.09.2023  00:00    <DIR>          ..
06.09.2023  00:00               130 config
06.09.2023  00:00                73 description
06.09.2023  00:00                23 HEAD
06.09.2023  00:00    <DIR>          hooks
06.09.2023  00:00    <DIR>          info
06.09.2023  00:00    <DIR>          objects
06.09.2023  00:00    <DIR>          refs
               3 Datei(en),            226 Bytes
               5 Verzeichnis(se), 1.626.402.078.720 Bytes frei
```

INTELLIJ

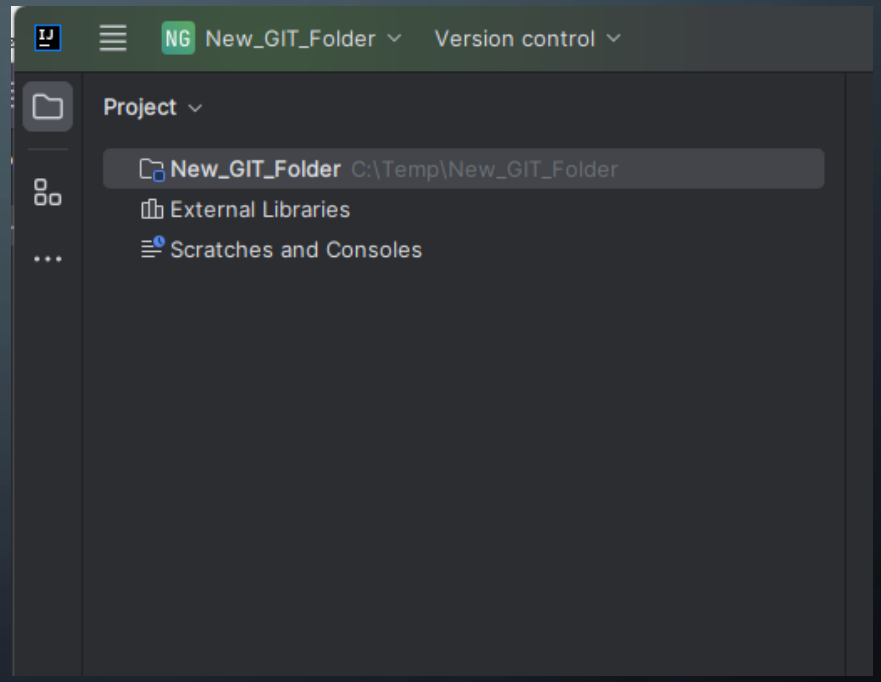


EXCERCISE: CREATE A NEW LOCAL REPOSITORY

INTELLIJ

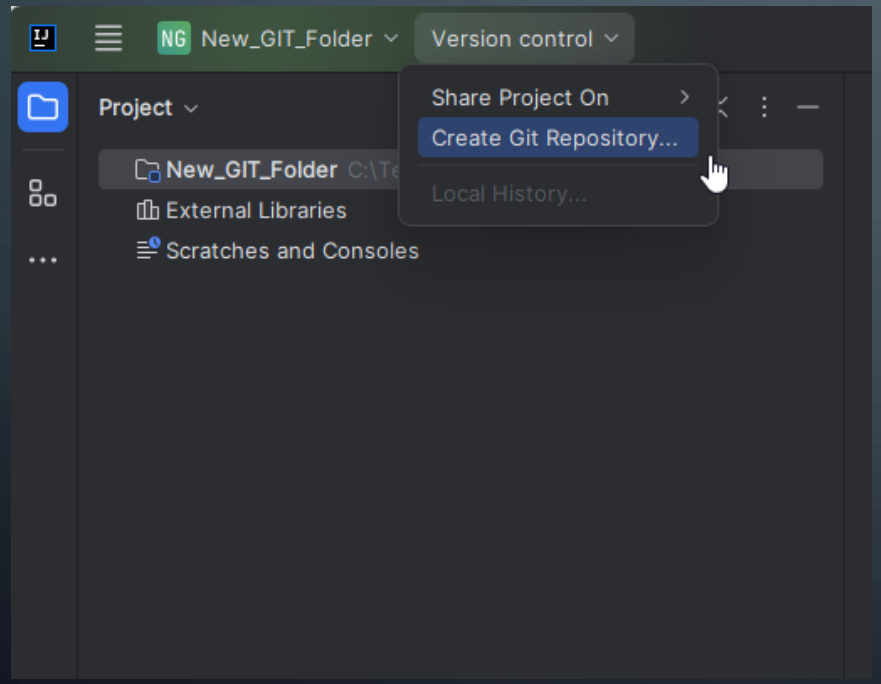


INTELLIJ

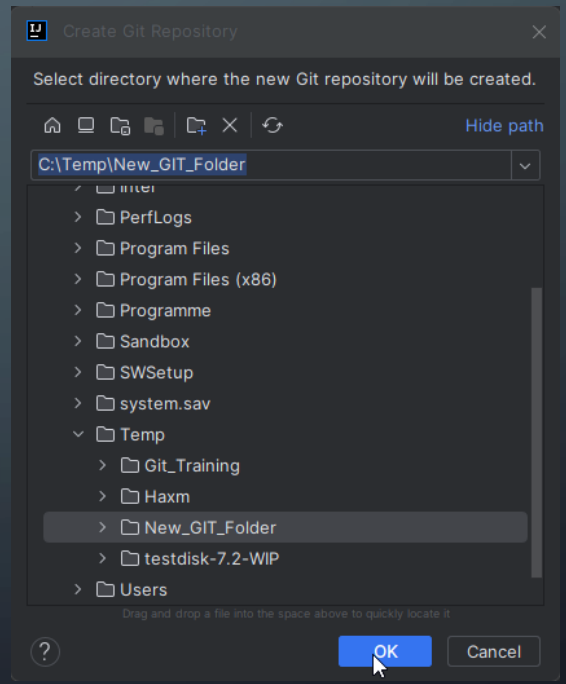


EXCERCISE: CREATE A NEW LOCAL REPOSITORY

INTELLIJ

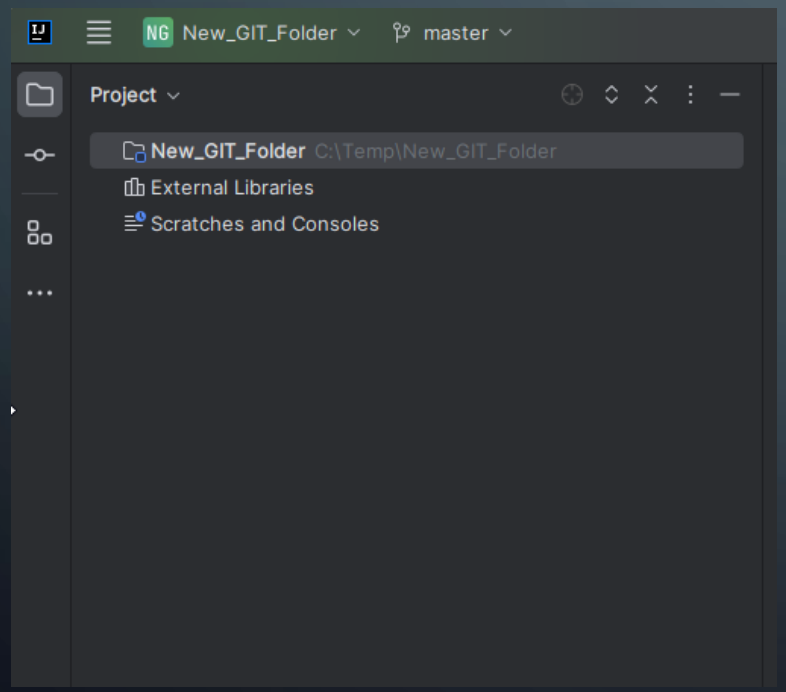


INTELLIJ



EXCERCISE: CREATE A NEW LOCAL REPOSITORY

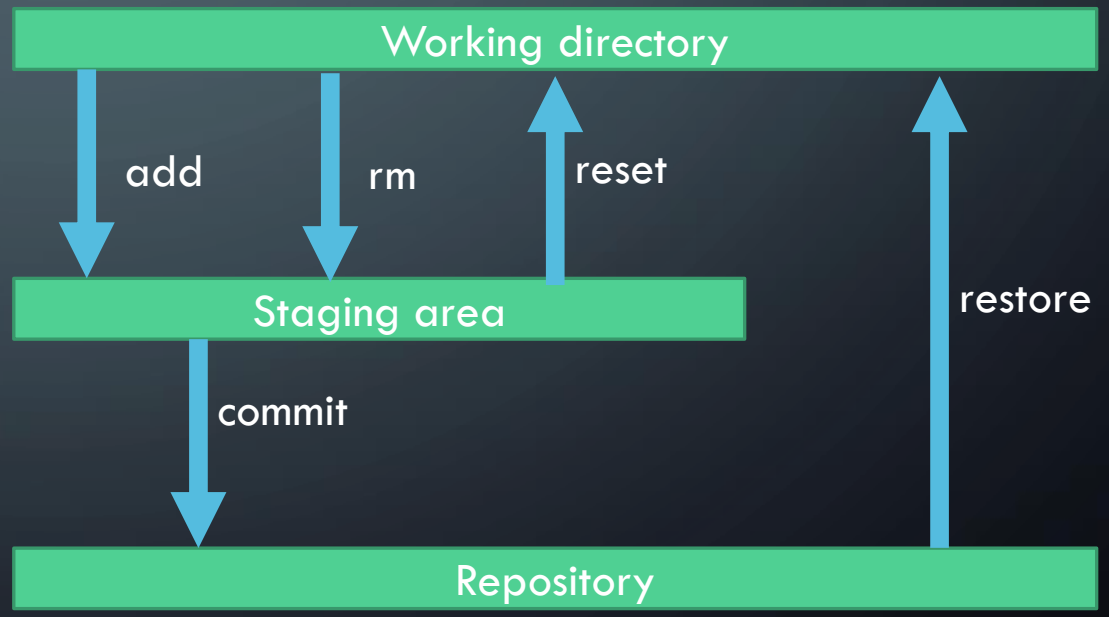
INTELLIJ



ADDING FILES / DIRECTORIES TO A REPOSITORY

- Adding files / directories to a repository is a 2 phase process:

1. Adding to a staging area
2. Committing to the repository



ADDING FILES / DIRECTORIES TO A REPOSITORY

- Here are the command you'll need in that context:

Command	Explanation
git status	Shows changes since last commit and the content of the staging area
git add <<file/dir . --all>>	Adds to the staging area if content is newer than last commit
git commit –m "Commit comment"	Adds files from staging into the repository
git reset HEAD <<file/dir name>>	Remove file/dir from staging
git rm <<file>> | -r <<dir>>	Remove file/dir and add it to staging
git restore <<file/dir>>	Restore file/dir from the repository
git log	Shows the commit history

ADDING FILES / DIRECTORIES TO A REPOSITORY

What	.gitignore	.git/info/exclude
Location	Every directory in the project can have one, but mostly found in the root directory.	Only one file on the above shown location in the project.
Function	Contains the information on what to ignore for /exclude from commits as of the directory it is in.	Contains the information on what to ingnore for / exclude from commits for the project.
Format	• # Start a comment line • * Stands for any character but a slash • / Separates directories • <<file/dir>> Matches the file or dir, absolute or relative • **/ Matches any directory + subdirectories • /** Matches all files and sub-dirs in a dir • [a-zA-Z0-9] Matches a range(s) of characters	

ADDING FILES / DIRECTORIES TO A REPOSITORY

Pattern in .gitignore	Explanation
node_modules	Ignore the node_modules dir (or file)
node_modules/	Ignore the content of the dir, but not the dir itself
# All bin files	Comment, has no consequence
/*/temp	Ignore the temp dir in each of the subdirs of the dir the .gitignore is in (1 level)
/**/readme	Ignore the file readme in any of the subdirs of the dir the .gitignore file is in
d/**/k	Matches d/k, d/x/k, d/x/y/k, and so on

EXERCISE: ADDING TO THE REPOSITORY

(TRY TO DO THE EXERCISE ON THE COMMAND LINE AND, IF POSSIBLE, IN INTELLIJ)

- Goto an empty directory, where you created a new GIT repositor and which you already opened as a project in IntelliJ.
- Look at the GIT status
- You should see an .idea directory, which is not comitted yet
- Add the directory to staging
- Look at the GIT status
- Remove the directory and its files from staging again
- Look at the GIT status

EXERCISE: ADDING TO THE REPOSITORY

(TRY TO DO THE EXERCISE ON THE COMMAND LINE AND, IF POSSIBLE, IN INTELLIJ)

- Make sure that the .idea directory is never considered for staging again
- Look at the GIT status
- Use IntelliJ to create file1.txt with this content:
  This is the first file I am going to check in
- Save the file
- Look at the GIT status
- Add the file to staging
- Look at the GIT status

EXERCISE: ADDING TO THE REPOSITORY

(TRY TO DO THE EXERCISE ON THE COMMAND LINE AND, IF POSSIBLE, IN INTELLIJ)

- Commit to the repository
- Look at the GIT status
- Add a second line to file1.txt and save it:

  Here is a modification
- Look at the GIT status
- Add all modified files to staging and commit them to the repository
- Look at the GIT status

EXERCISE: ADDING TO THE REPOSITORY

(TRY TO DO THE EXERCISE ON THE COMMAND LINE AND, IF POSSIBLE, IN INTELLIJ)

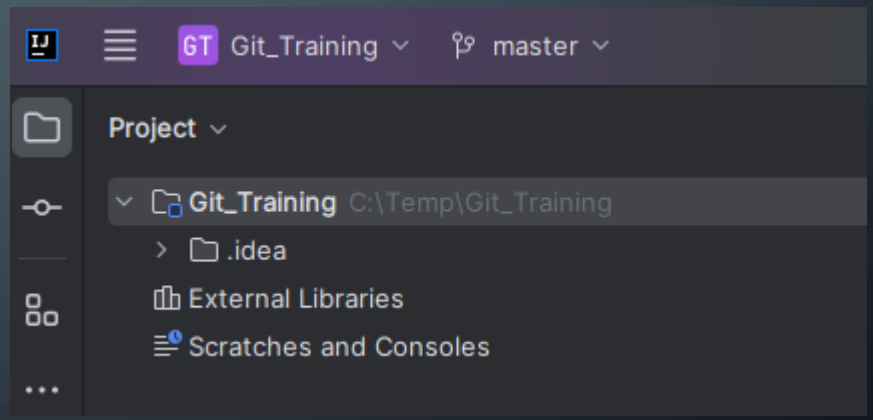
- Use the OS GUI to delete file1.txt from the directory
- Look at the GIT status
- Get the latest version of the file from the repository
- Look at the GIT status
- Have a look at the commit history

EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

06.09.2023  00:07    <DIR>          .
06.09.2023  00:43    <DIR>          ..
06.09.2023  00:07    <DIR>          .idea
               0 Datei(en),              0 Bytes
               3 Verzeichnis(se), 1.620.425.322.496 Bytes frei
```

INTELLIJ

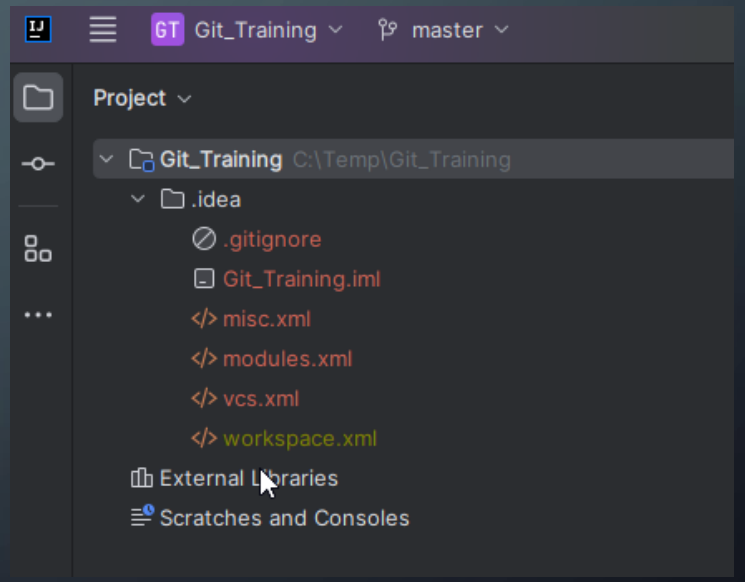


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/

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

INTELLIJ

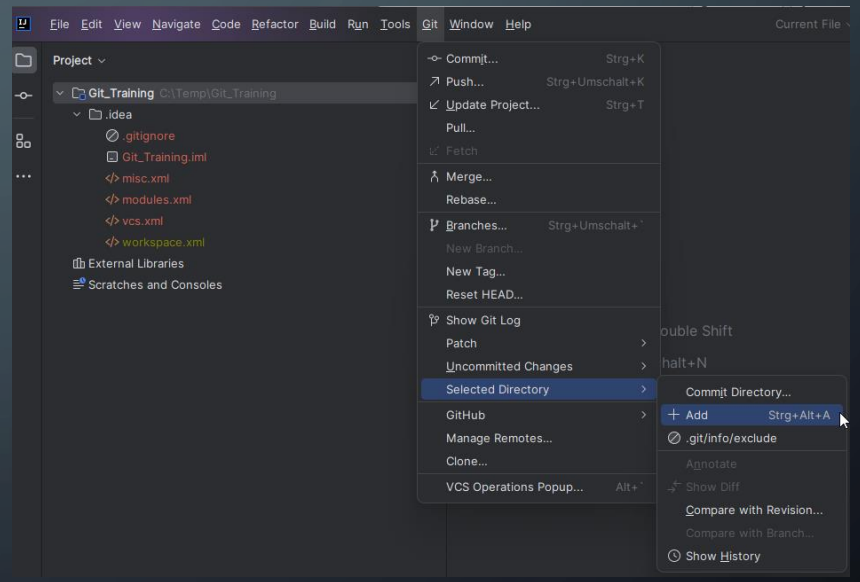


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git add .idea

GIT Training > git add .

GIT Training > git add --all
warning: in the working copy of '.idea/misc.xml', LF will
be replaced by CRLF the next time Git touches it
```

INTELLIJ

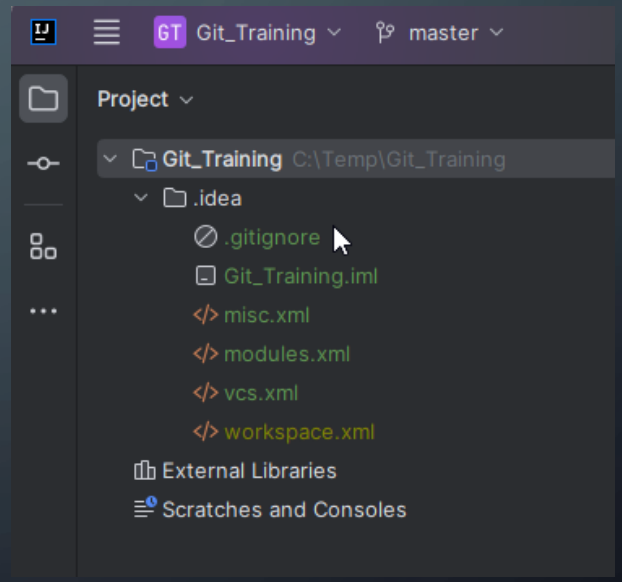


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .idea/.gitignore
        new file:   .idea/Git_Training.iml
        new file:   .idea/misc.xml
        new file:   .idea/modules.xml
        new file:   .idea/vcs.xml
```

INTELLIJ

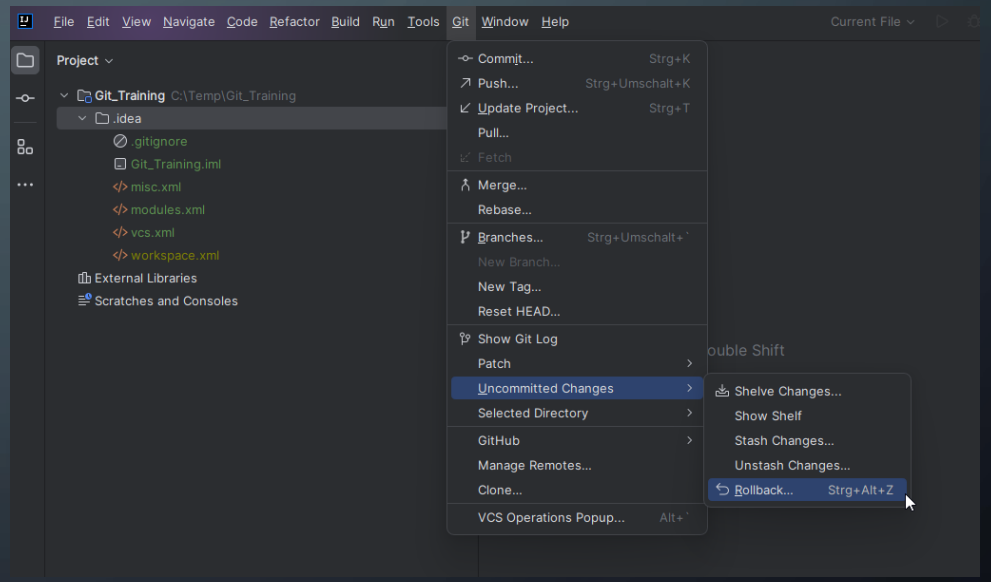


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git reset .idea
```

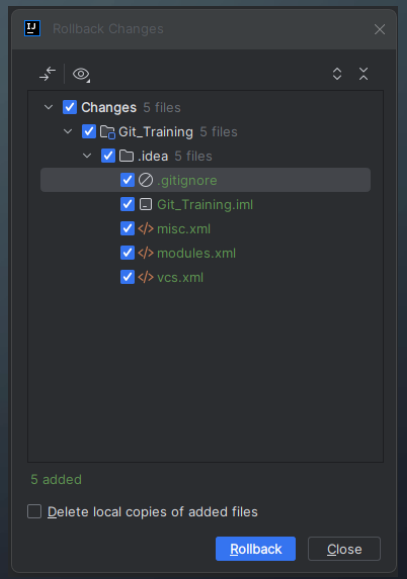
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

INTELLIJ

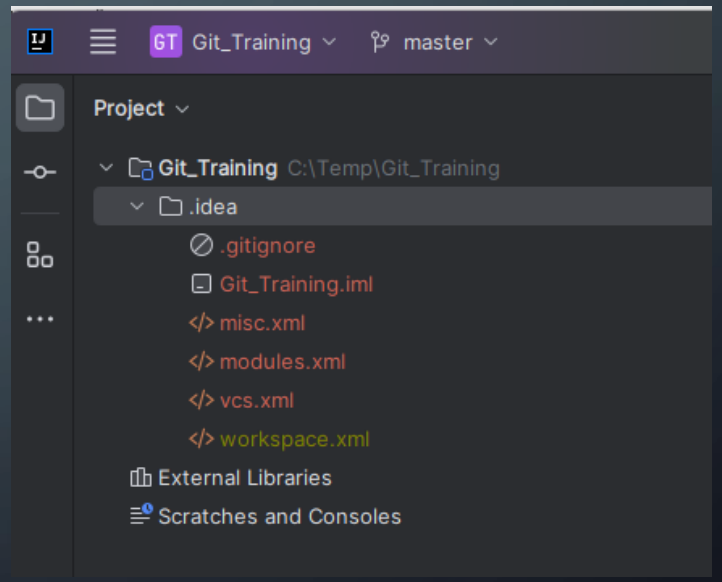


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/

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

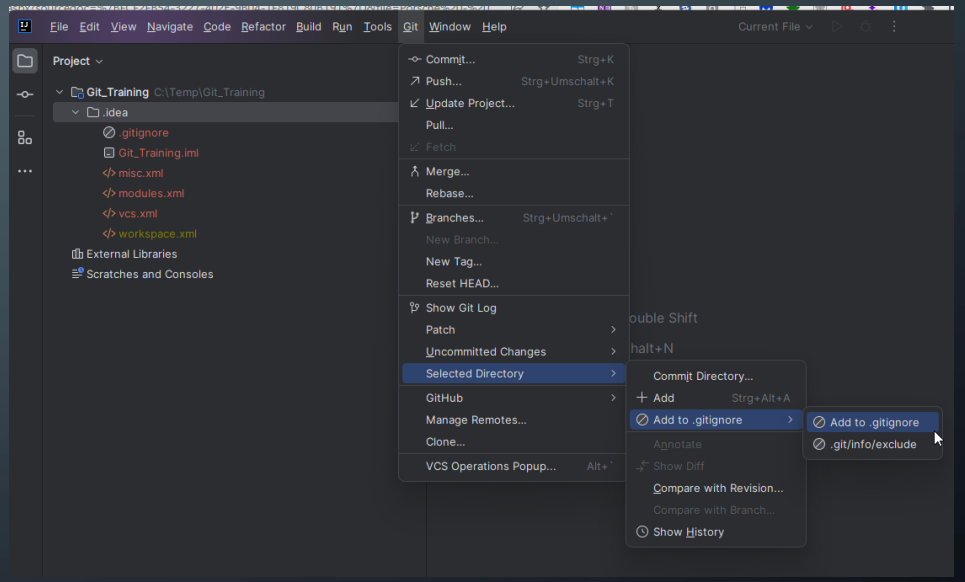
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

INTELLIJ

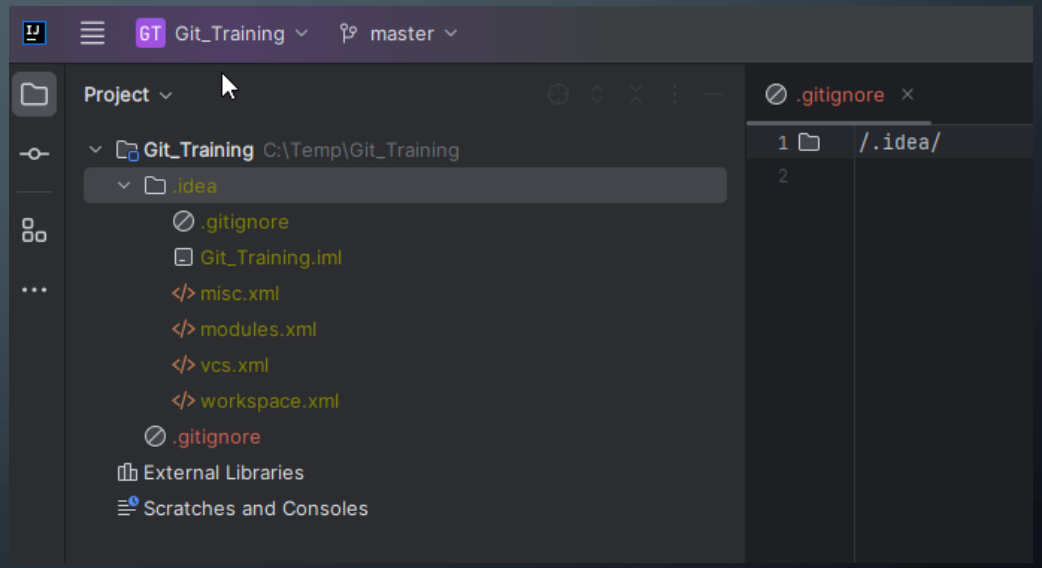


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

07.09.2023  23:37    <DIR>          .
06.09.2023  00:43    <DIR>          ..
07.09.2023  23:37                 9 .gitignore
07.09.2023  23:29    <DIR>          .idea
               1 Datei(en),              9 Bytes
               3 Verzeichnis(se), 1.620.403.814.400 Bytes frei

GIT Training > git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

INTELLIJ

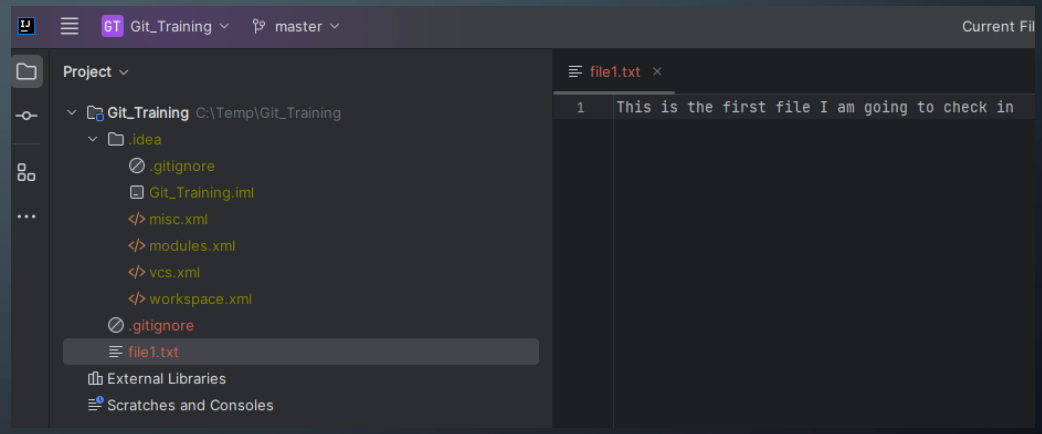


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

07.09.2023  23:42    <DIR>          .
06.09.2023  00:43    <DIR>          ..
07.09.2023  23:37                 9 .gitignore
07.09.2023  23:29    <DIR>          .idea
07.09.2023  23:42                45 file1.txt
               2 Datei(en),             54 Bytes
               3 Verzeichnis(se), 1.620.402.503.680 Bytes frei

GIT Training > git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file1.txt

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

INTELLIJ

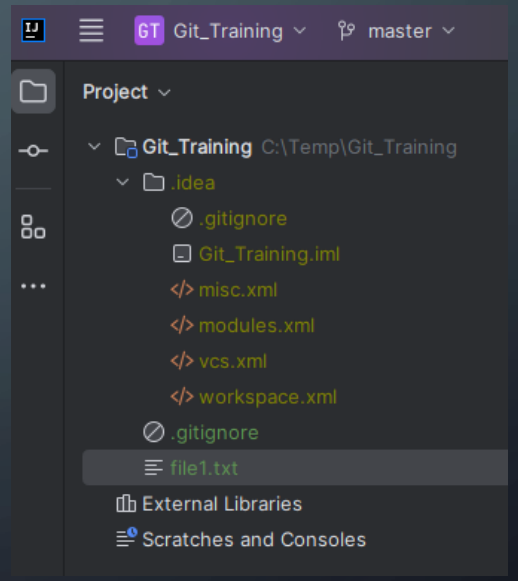


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

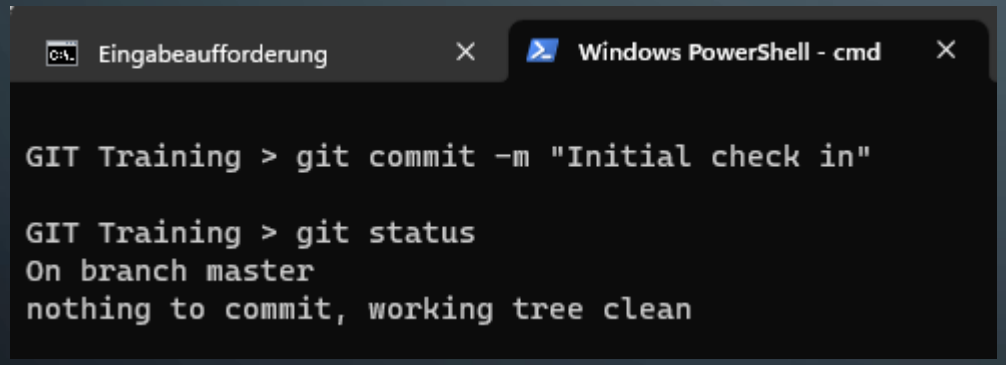
```
GIT Training > git add --all

GIT Training > git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   file1.txt
```

INTELLIJ

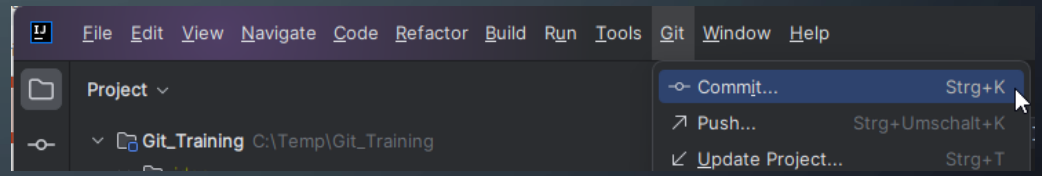


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT



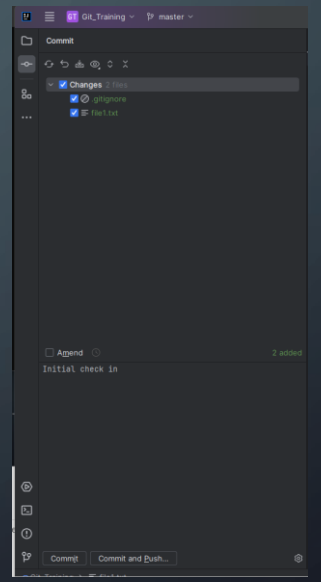
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

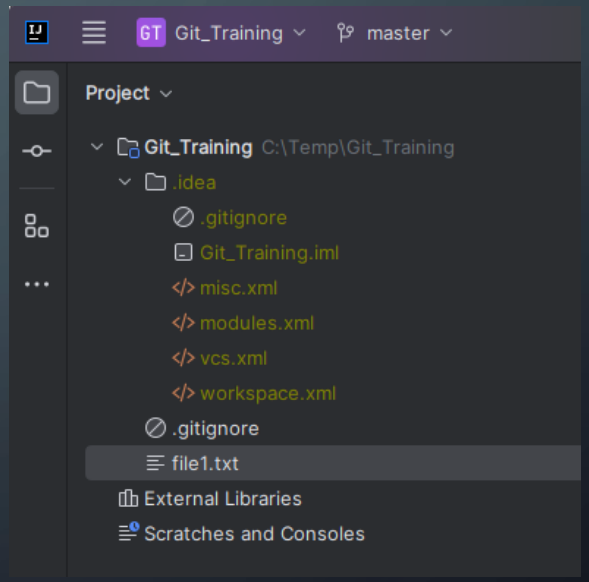
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

INTELLIJ

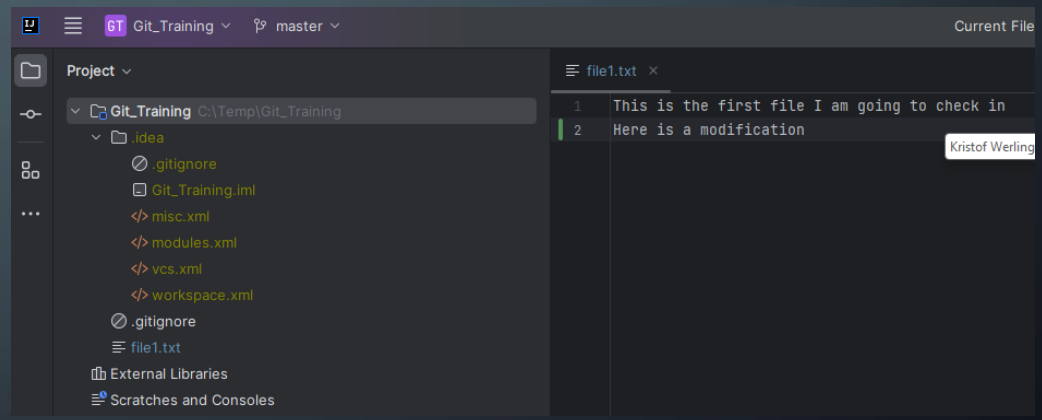


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > 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:   file1.txt

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

INTELLIJ

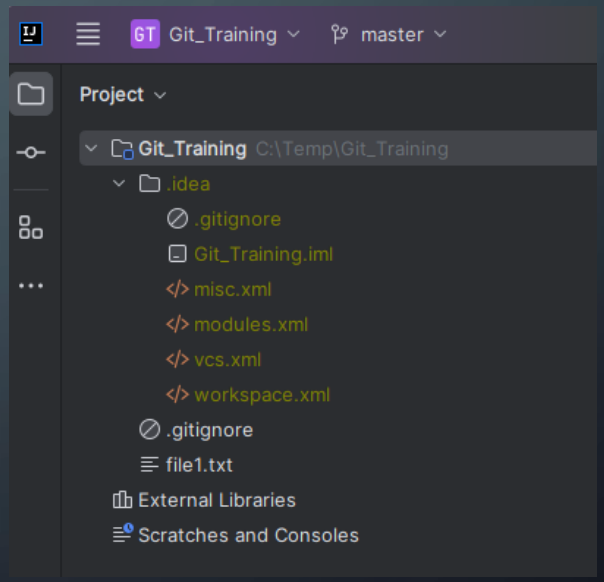


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git add --all

GIT Training > git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1.txt


GIT Training > git commit -m "Added a second line to file1.txt"
[master 1a2cf6d] Added a second line to file1.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

GIT Training > git status
On branch master
nothing to commit, working tree clean
```

INTELLIJ

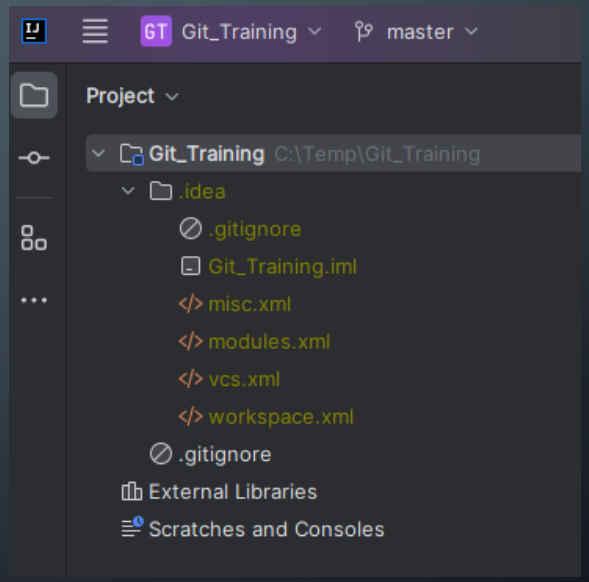


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > del file1.txt

GIT Training > git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file1.txt

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

GIT Training > git add --all

GIT Training > git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file1.txt
```

INTELLIJ

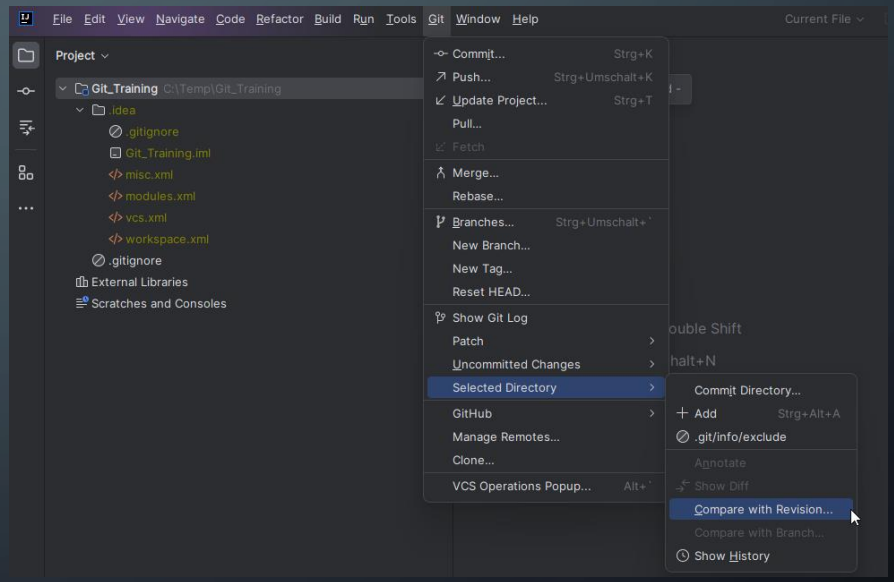


EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

```
GIT Training > git restore file1.txt

GIT Training > git status
On branch master
nothing to commit, working tree clean

GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

08.09.2023  00:07    <DIR>          .
06.09.2023  00:43    <DIR>          ..
07.09.2023  23:37                 9 .gitignore
08.09.2023  00:00    <DIR>          .idea
08.09.2023  00:07                69 file1.txt
               2 Datei(en),             78 Bytes
               3 Verzeichnis(se), 1.620.389.797.888 Bytes frei
```

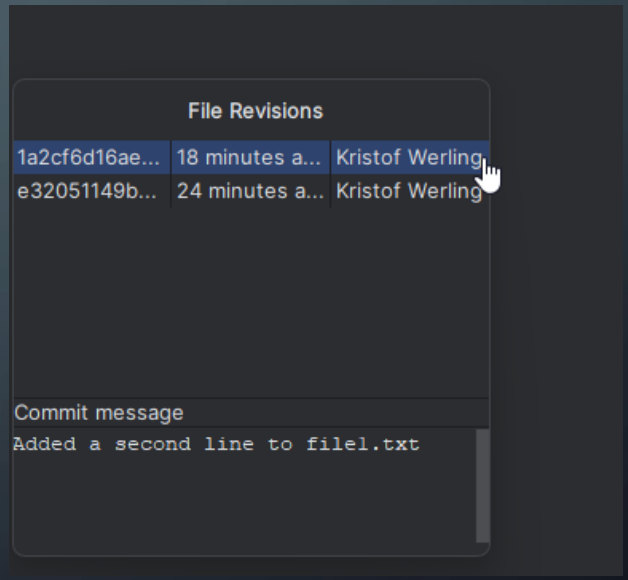
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

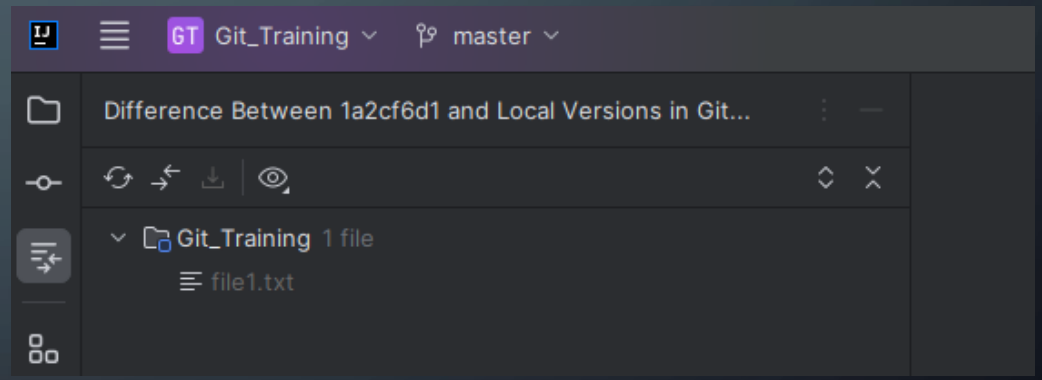
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

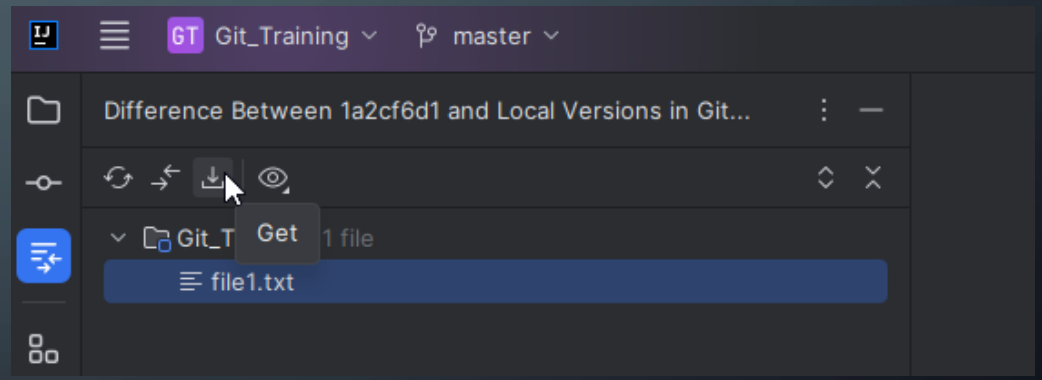
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

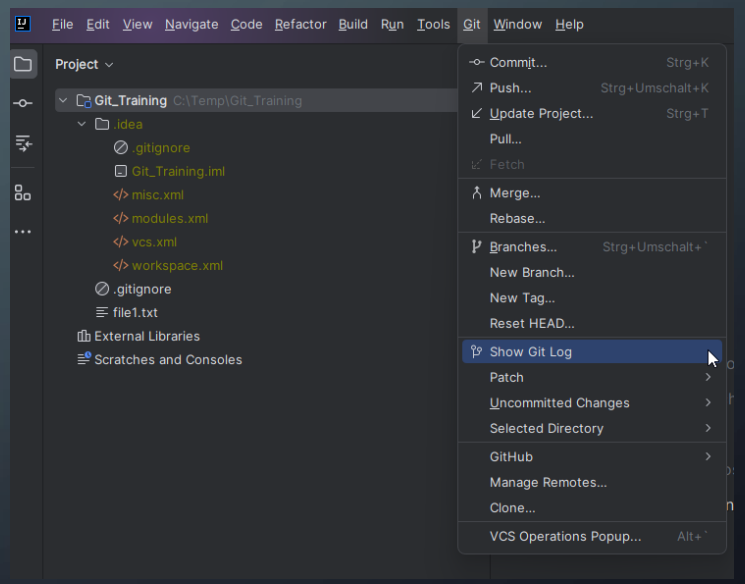
INTELLIJ



EXERCISE: ADDING TO THE REPOSITORY

COMMAND PROMPT

INTELLIJ

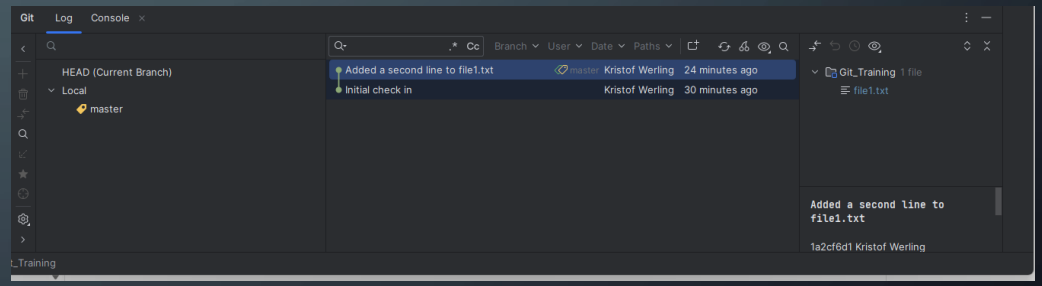


EXERCISE: ADDING TO THE REPOSITORY

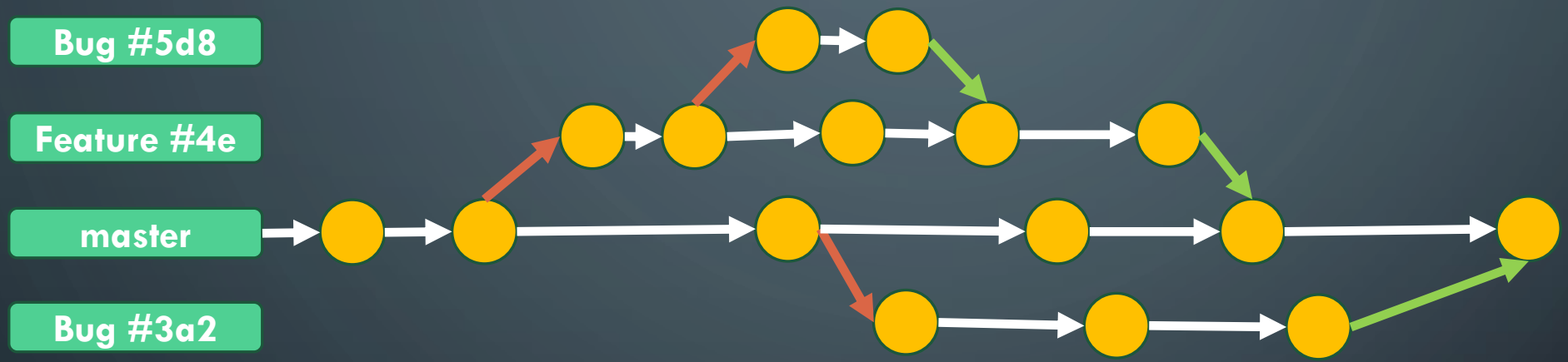
COMMAND PROMPT

```
GIT Training > git log
commit 1a2cf6d16ae94bd00a6dc9b2429bd73d077d23be (HEAD -> master)
Author: Kristof Werling <kwerling@gmail.com>
Date:   Thu Sep 7 23:57:00 2023 +0200

    Added a second line to file1.txt

commit e32051149b71207410f5c4f07b01053b233618db
Author: Kristof Werling <kwerling@gmail.com>
Date:   Thu Sep 7 23:50:59 2023 +0200

    Initial check in
```

INTELLIJ



WORKING WITH BRANCHES



Legend	master	Branch name			Same branch
					New branch
		Commit			Merge

WORKING WITH BRANCHES

- Here are the command you'll need in that context:

Command	Explanation
git branch <<branch>>	Creates a new branch
git switch <<branch>>	Switch to another branch
git merge <<branch>>	Merges the <<branch>> branch into the active one
git mergetool	UI to resolve merge conflicts
git stash	Saves changes to prevent a commit
git stash apply	Re-apply the saved changes
git stash list | drop	List all saved changes | removes a saved change
git branch –[dD] <<branch>>	Deletes branch (-D) or only when merged (-d)

WORKING WITH BRANCHES

- Merge conflicts:
  - They occur when merging overlapping changes from two branches
  - One can resolve them manually (with an editor) or use the "git mergetool" command, IntelliJ, or one of the many other available tool
  - Once resolved the modified file needs to be committed to the repository
  - The modifications to the file(s) only happen in the actual branch.

EXERCISE: WORKING WITH BRANCHES

- Create a branch called "feature"
- Switch to it
- Create a file, called "mergefile.txt" and enter the following text:

  This is a change to the feature branch
- Commit the changes to the repository
- Switch back to the master branch

EXERCISE: WORKING WITH BRANCHES

- Create a file, called "mergefile.txt" and enter the following text:

  This is a change to the master branch

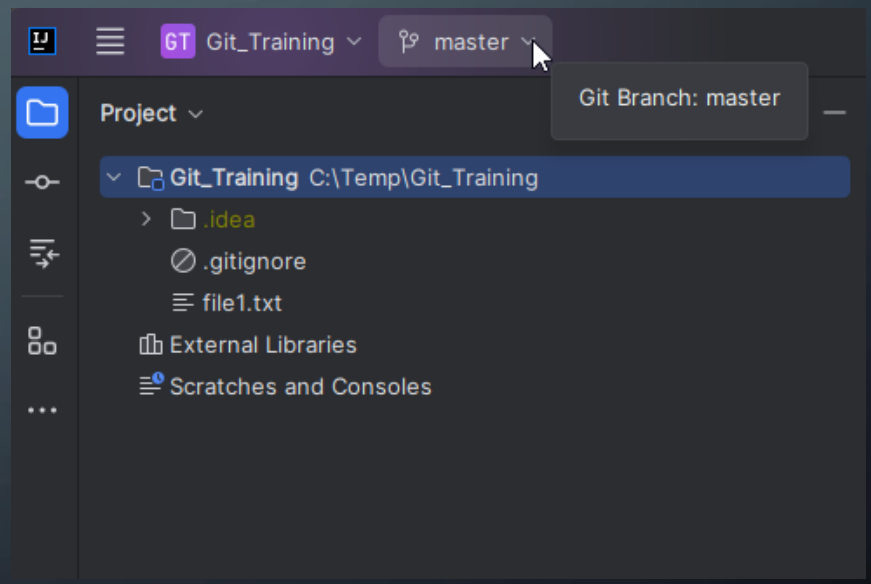
- Commit the changes to the repository
- Merge the "feature" branch into the "master" branch
- Resolve the merge conflict.
- Delete the "feature" branch

EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 4E37-3A48

 Verzeichnis von C:\Temp\Git_Training

08.09.2023  00:16    <DIR>          .
06.09.2023  00:43    <DIR>          ..
07.09.2023  23:37                 9 .gitignore
08.09.2023  00:16    <DIR>          .idea
08.09.2023  00:16                69 file1.txt
               2 Datei(en),             78 Bytes
               3 Verzeichnis(se), 1.605.952.233.472 Bytes frei

GIT Training > git status
On branch master
nothing to commit, working tree clean

GIT Training > git branch feature

GIT Training > git status
On branch master
nothing to commit, working tree clean

GIT Training > git switch feature
Switched to branch 'feature'
```

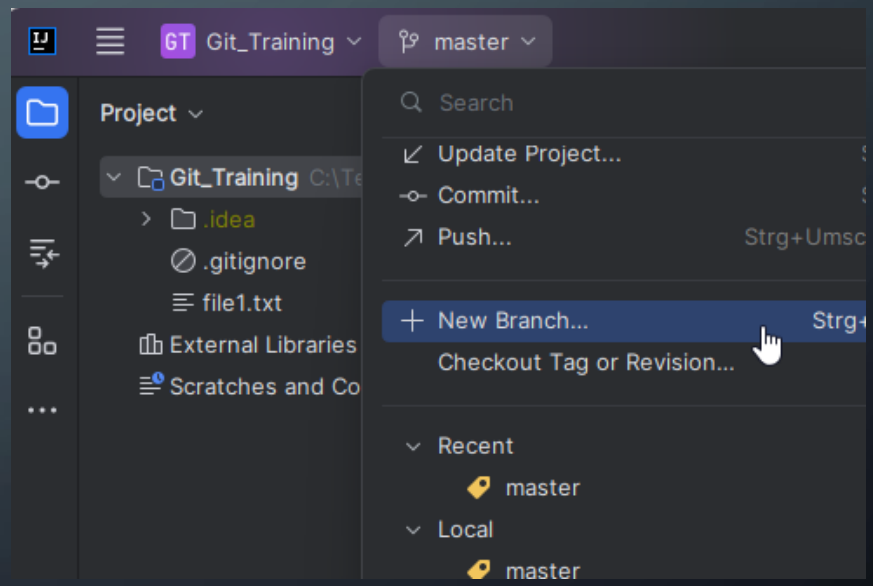
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

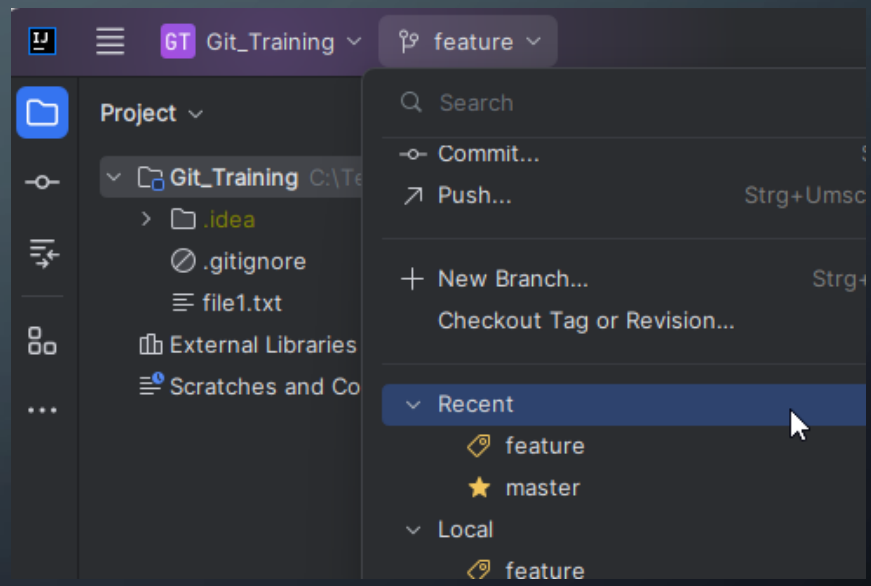
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

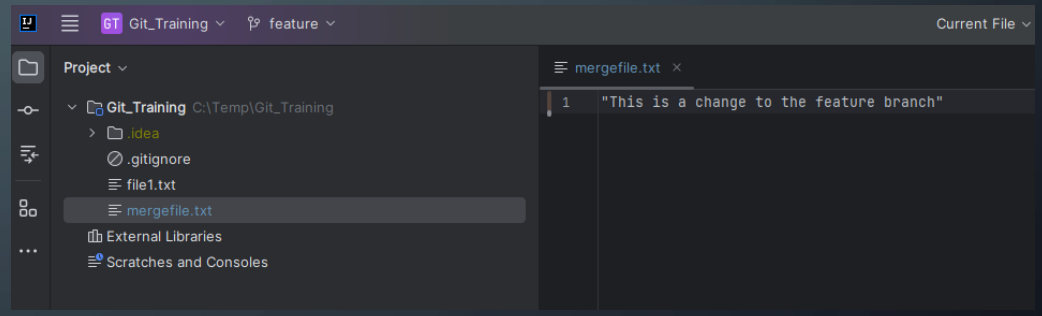
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

INTELLIJ

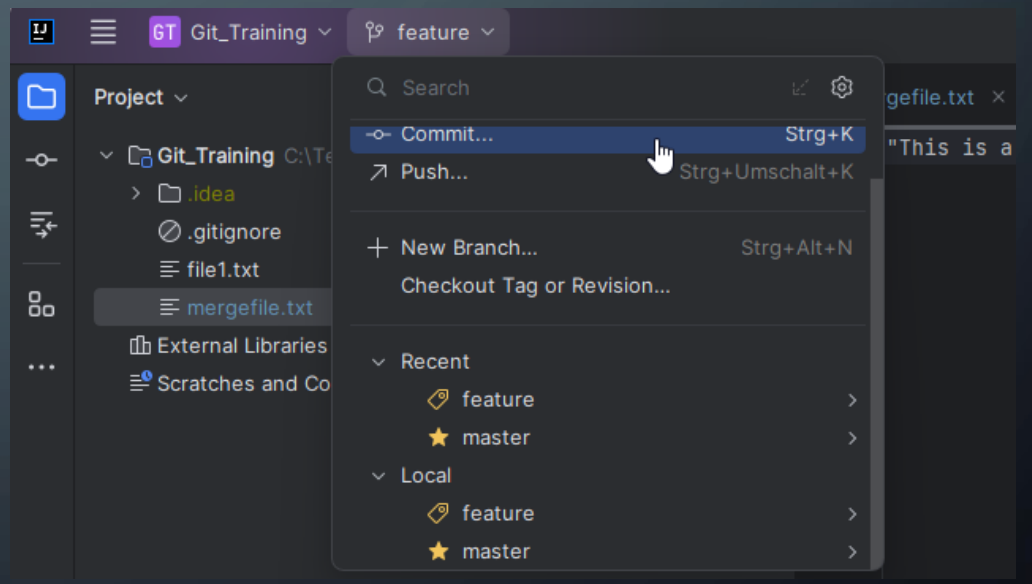


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > echo "This is a change to the feature branch" > mergefile.txt

GIT Training > git add --all

GIT Training > git commit -m "Initial version of mergefile.txt"
[feature fbdb7dd] Initial version of mergefile.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mergefile.txt
```

INTELLIJ

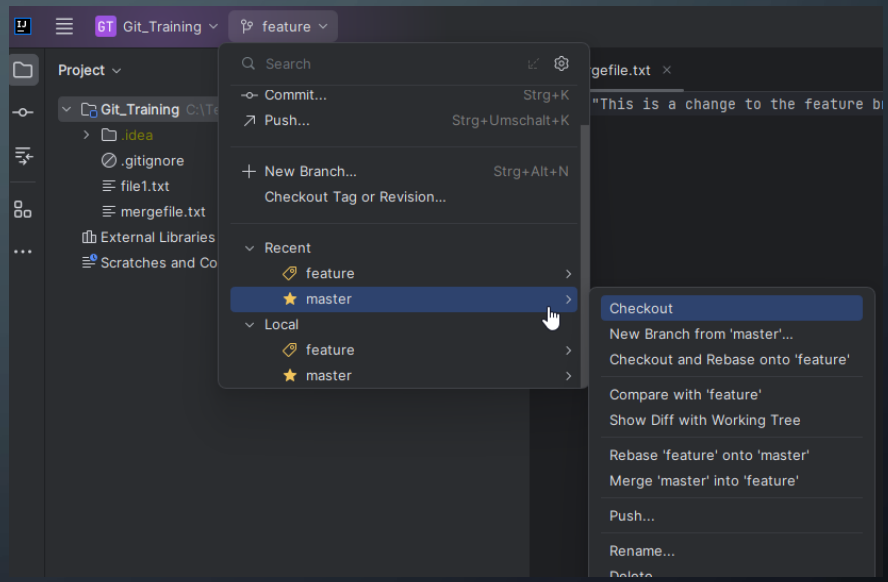


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > git switch master
Switched to branch 'master'

GIT Training > echo "This is a change to the master branch" > mergefile.txt

GIT Training > git add --all

GIT Training > git commit -m "Initial version of mergefile.txt"
[master 954bba2] Initial version of mergefile.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

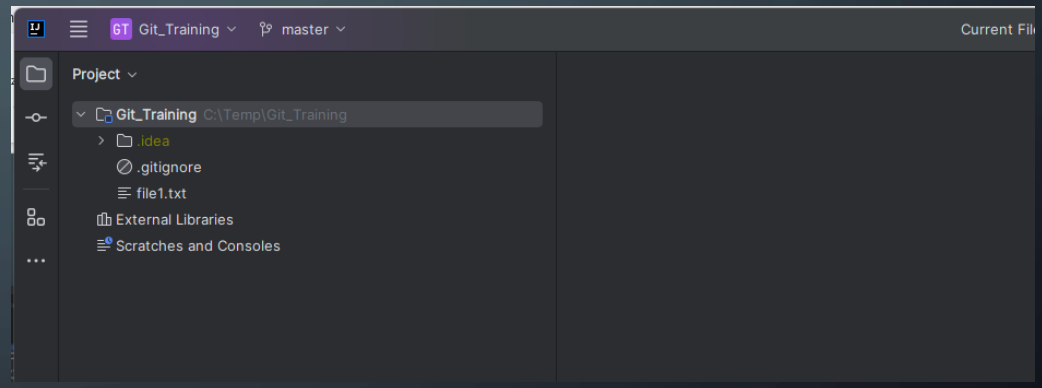
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

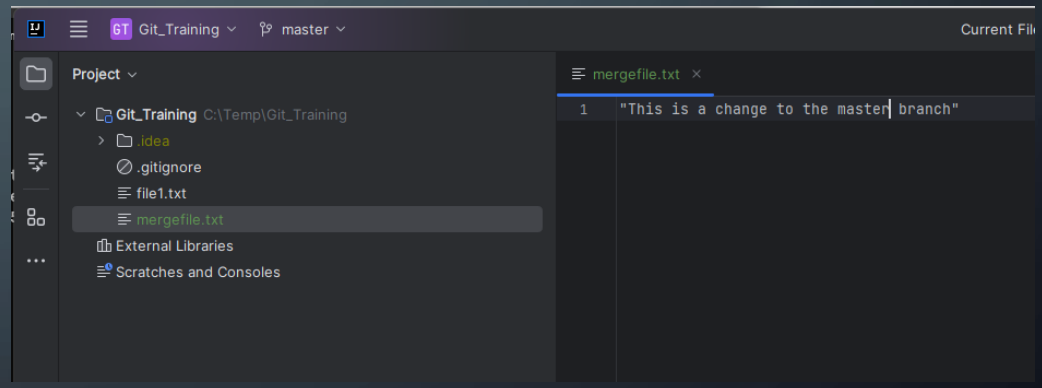
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

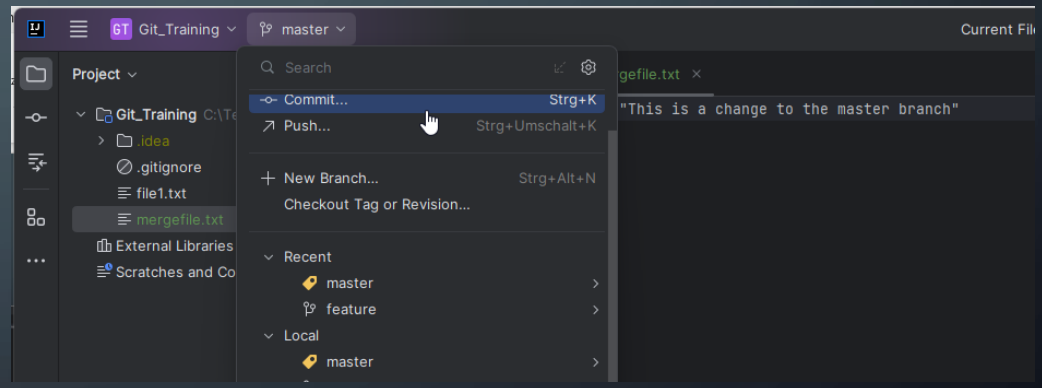
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

INTELLIJ

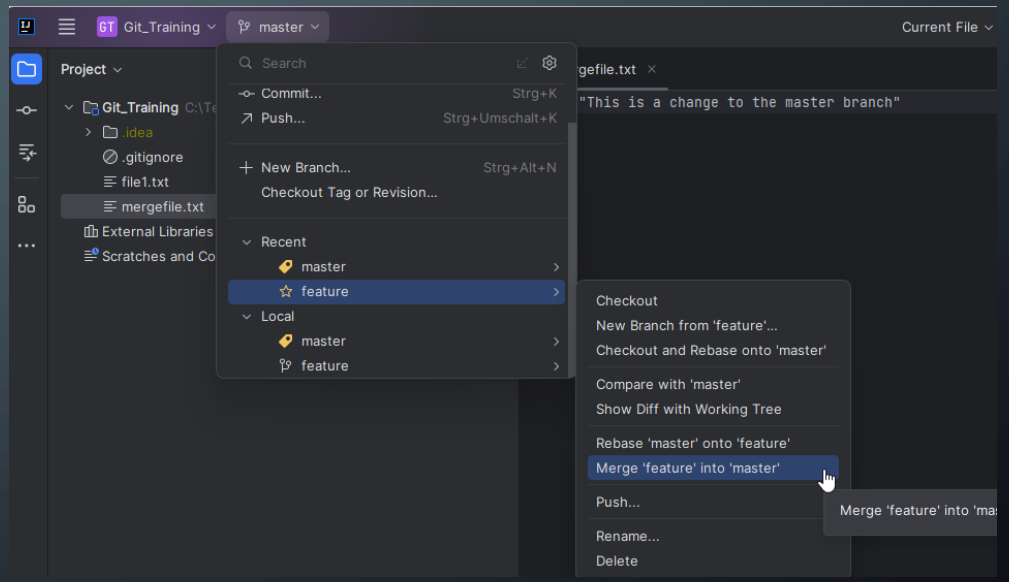


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > git merge feature
Auto-merging mergefile.txt
CONFLICT (add/add): Merge conflict in mergefile.txt
Automatic merge failed; fix conflicts and then commit the result.

GIT Training > type mergefile.txt
<<<<<<< HEAD
"This is a change to the master branch"
=======
"This is a change to the feature branch"
>>>>>>> feature
```

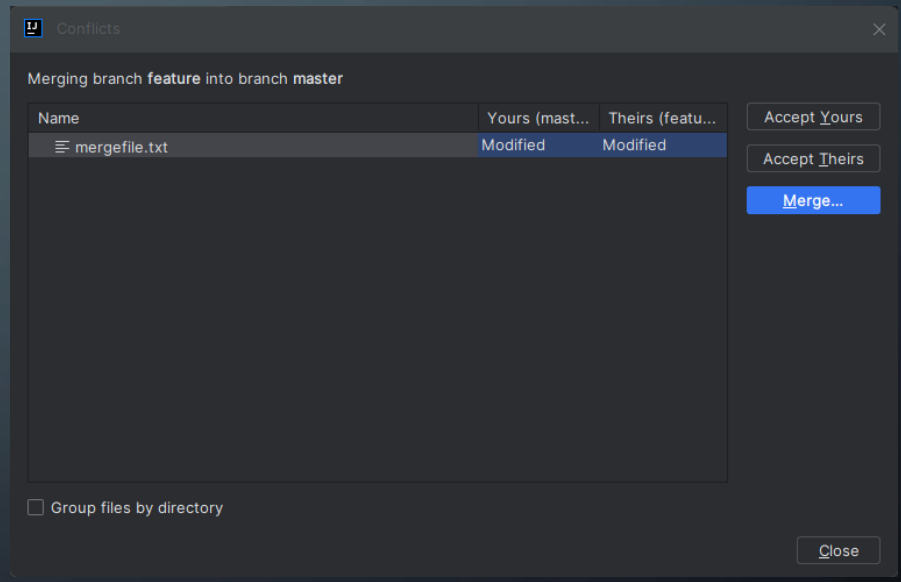
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

INTELLIJ

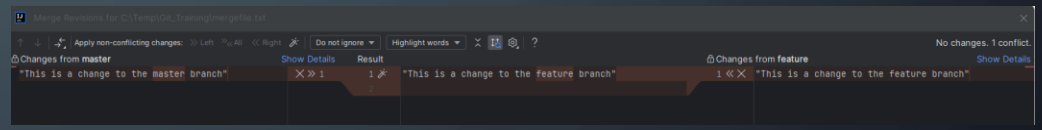


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

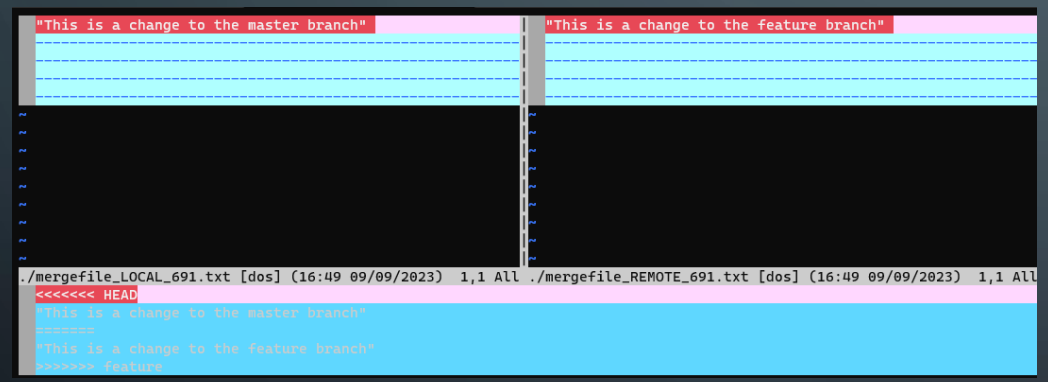
```
GIT Training > git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
mergefile.txt

Normal merge conflict for 'mergefile.txt':
  {local}: created file
  {remote}: created file
Hit return to start merge resolution tool (vimdiff):
```

INTELLIJ

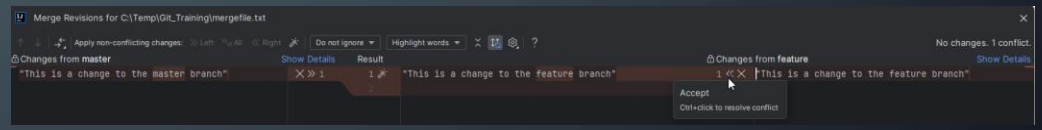


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT



INTELLIJ

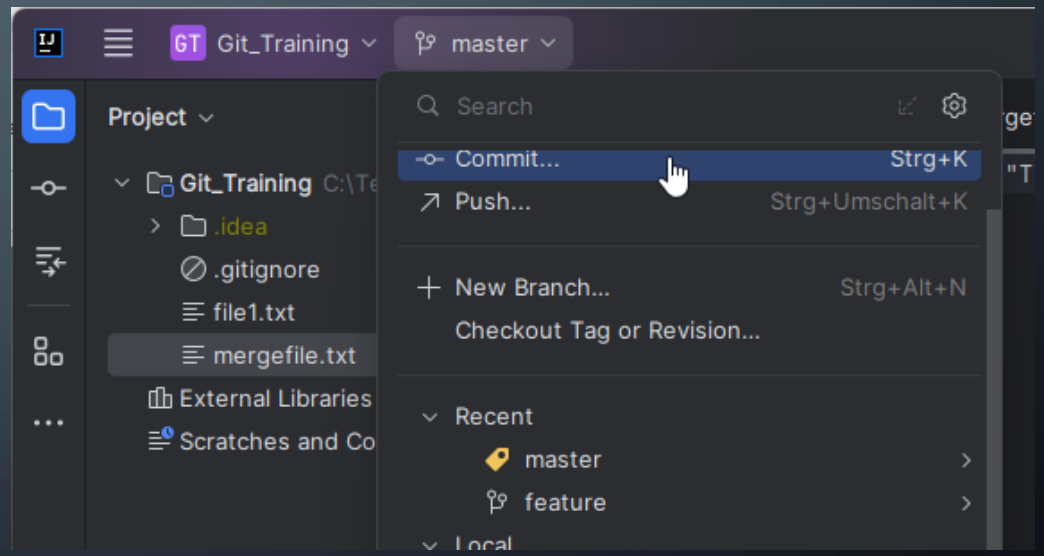


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > type mergefile.txt
"This is a change to the feature branch"

GIT Training > git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   mergefile.txt

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:   mergefile.txt


GIT Training > git add --all

GIT Training > git commit -m "Implemented FEATURE in mergefile.txt"
[master a924276] Implemented FEATURE in mergefile.txt
```

INTELLIJ

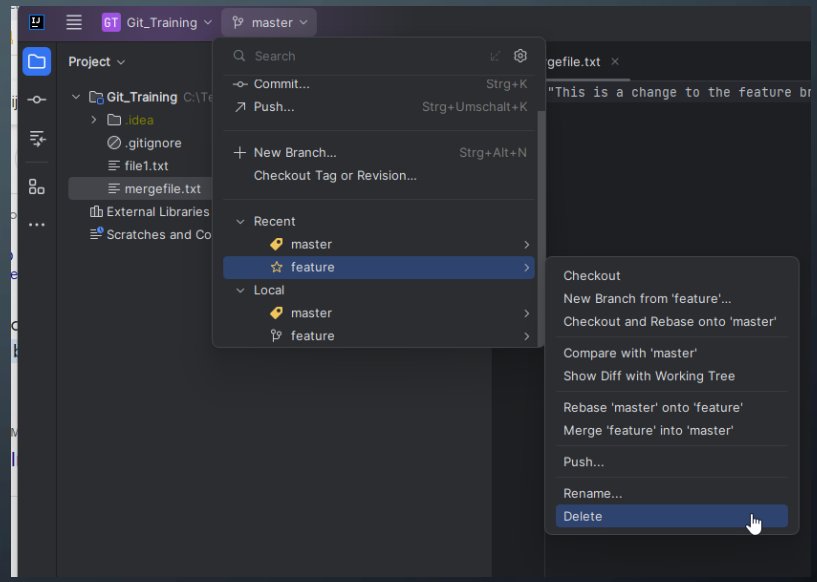


EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

```
GIT Training > git branch
  feature
* master

GIT Training > git status
On branch master
nothing to commit, working tree clean

GIT Training > git branch -d feature
Deleted branch feature (was a924276).
```

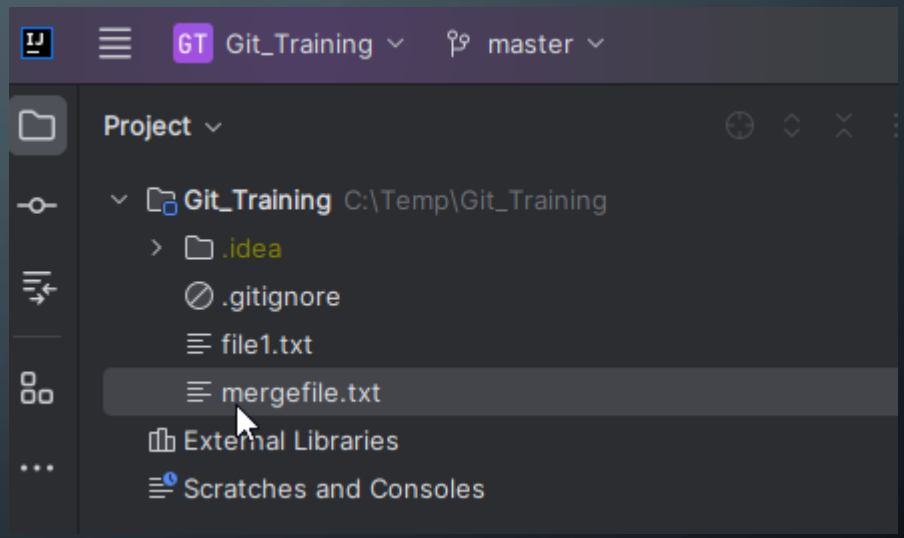
INTELLIJ



EXERCISE: WORKING WITH BRANCHES

COMMAND PROMPT

INTELLIJ



WORKING WITH BRANCHES

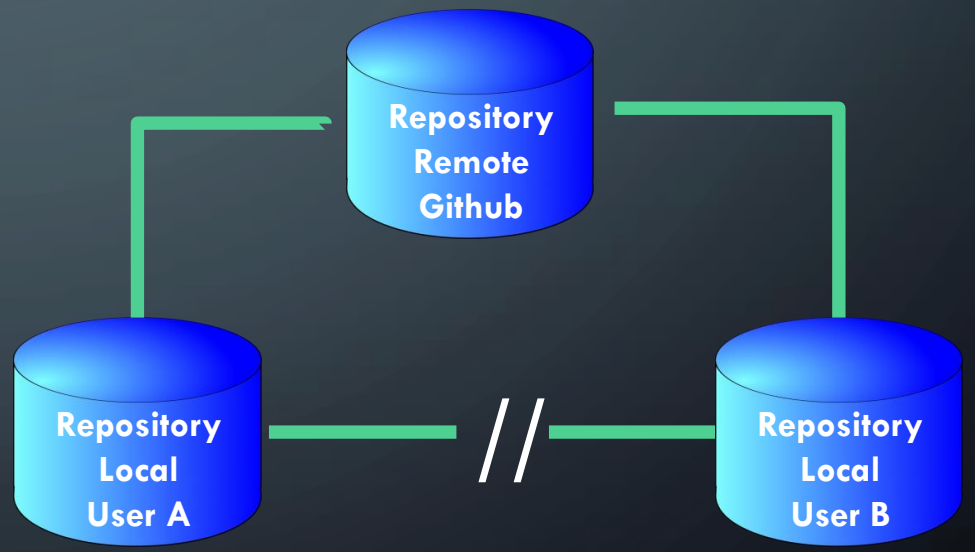
- There is no predominant branch in GIT. This is decided by convention.
- The GIT tool uses "master" as the default branch
- Because of the connection of the word master with racism, GITHUB renamed their default branch into "main"
- Theoretically one could have any other branch name as the default one.

REMOTE REPOSITORIES

EXPLANATION

- When synced the three repositories contain the same information
- Usually, the remote repository is considered to be the main one
- User A & B sync from remote, but not between themselves

USUAL SETUP OF REMOTE AND LOCAL REPOSITORIES



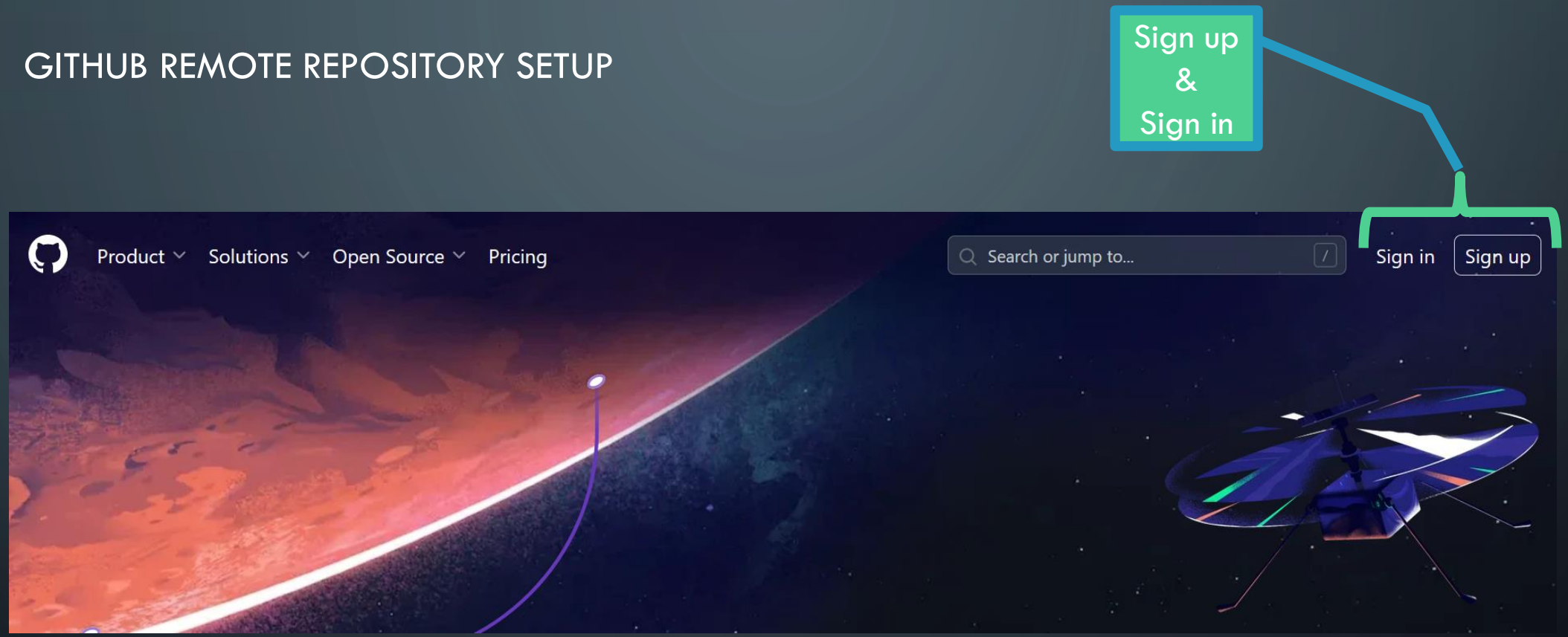
REMOTE REPOSITORIES

GITHUB REMOTE REPOSITORY SETUP

1. If not already done: Sign up @ github.com
2. Sign into your account
3. Create a repository for the upcoming exercises

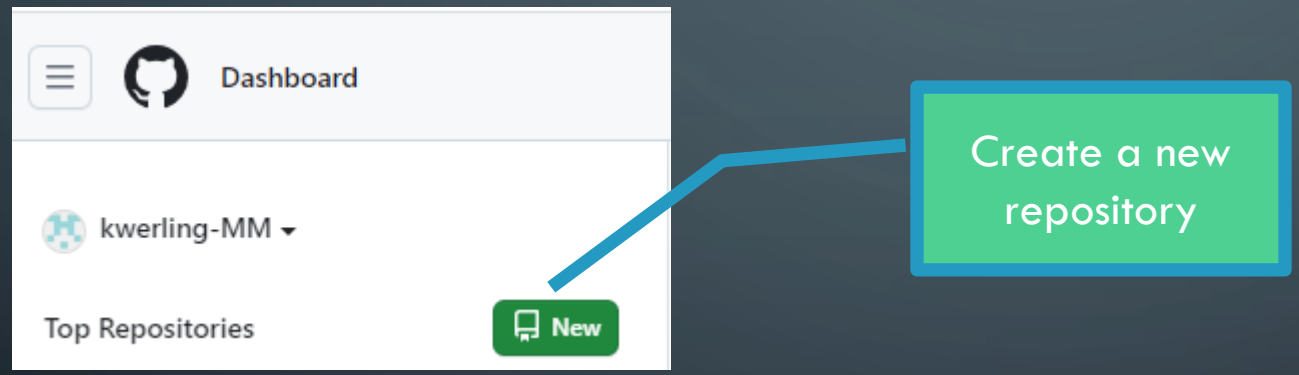
REMOTE REPOSITORIES

GITHUB REMOTE REPOSITORY SETUP



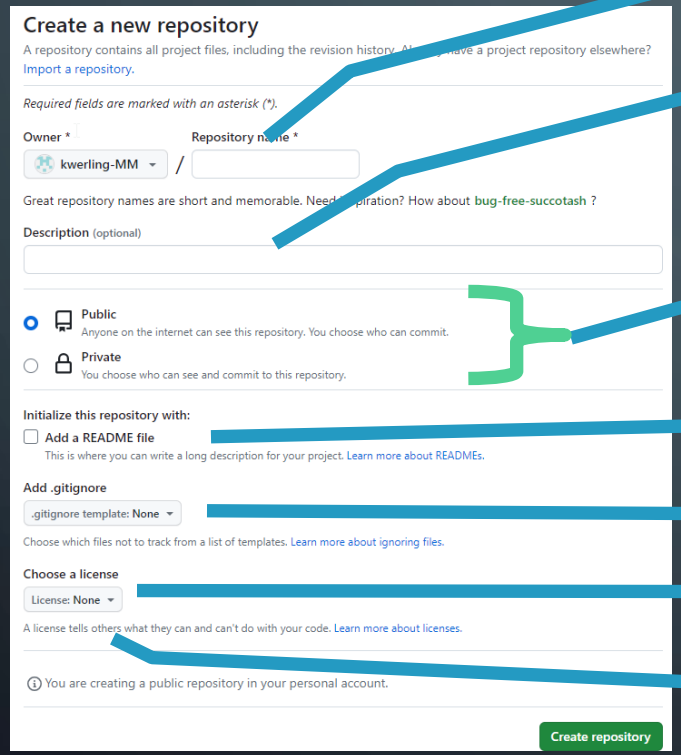
REMOTE REPOSITORIES

GITHUB REMOTE REPOSITORY SETUP



REMOTE REPOSITORIES

GITHUB REMOTE REPOSITORY SETUP



Give it a name

Explain, what the repository is for

Decide, it the repository is public or private

Do you want a Readme.md file?

Get a head start on .gitignore for your dev env

Choose a license, important for public repositories

Set the default branch name

Create the repository

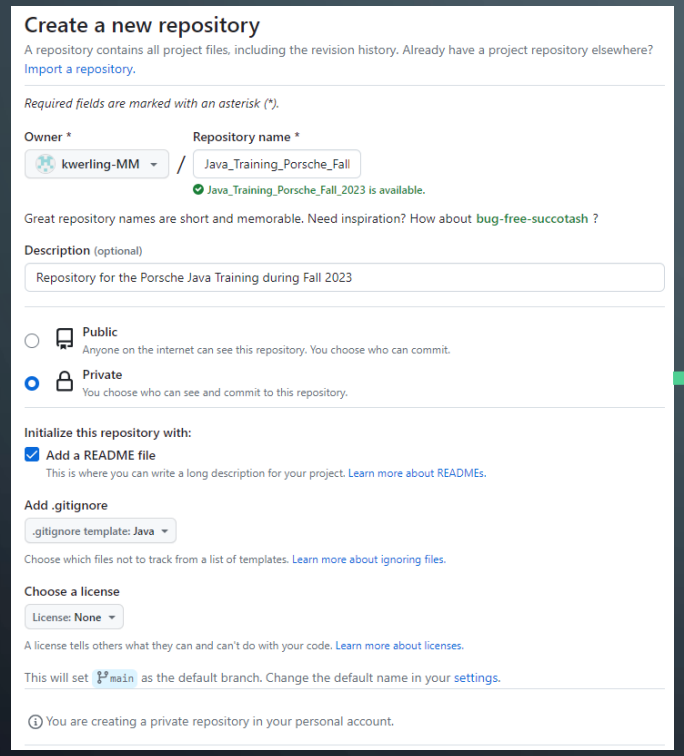
REMOTE REPOSITORIES

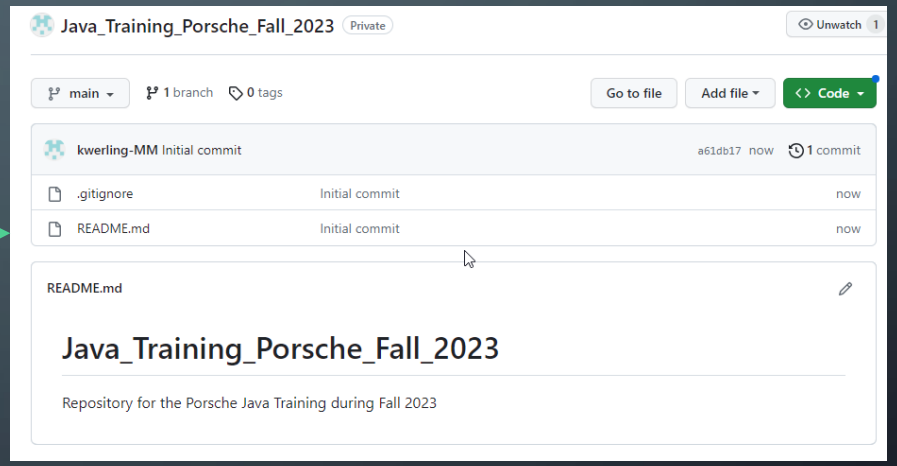
GITHUB REMOTE REPOSITORY SETUP

- Github uses the default branch "main"
- Our repository uses the default branch "master"
- This can be taken care of in the setup of the remote repository in our local repository

REMOTE REPOSITORIES

GITHUB REMOTE REPOSITORY SETUP





REMOTE REPOSITORIES

- A remote repository is a associated with one (or many) local repositories.
- The remote repository does not configure the local ones, it is vice versa.
- The access to a remote repository can be managed and by that restricted to individual users. Write or read access can also be configured. There is no finer granularity than the individual remote repository.
- GitHub, GitLab, BitBucket, Google, AWS, and other platforms provide remote repository services for GIT.

REMOTE REPOSITORIES

- Here are the command you'll need in that context:

Command		Explanation
git remote	add <<name>> <<url>>	Adds a new remote
	-v	Lists info on remotes
	remove <<name>>	Removes a remote
	set-head <<name>>	Sets the default branch
git push		Push local changes to remote
git pull (== git fetch + git merge)		Sync up with remote
git fetch		Syncs remote into local tracking branches
git merge		Merges tracking branches with local repository

REMOTE REPOSITORIES

- "origin" is the default name for the remote repository
- Name it like that to save some typing
- A branch can have many remote repositories associated with it

REMOTE REPOSITORIES

- Use "git clone << url >>" to copy the content of a remote repository to a local one
- As of then the usual comments "git add", "git commit", and "git push" can be used without any additional configuration.

EXERCISE: REMOTE REPOSITORIES

- Add your GitHub remote repository as "origin" to the local repository
- Push your local directory to "origin"
- On GitHub: Switch to the master branch
- On GitHub: Simulate another user working with the repository by adding the line "Changed by someone else" to file1.txt and commit
- Local: Add the line "Local change" to the file mergefile.tx

EXERCISE: REMOTE REPOSITORIES

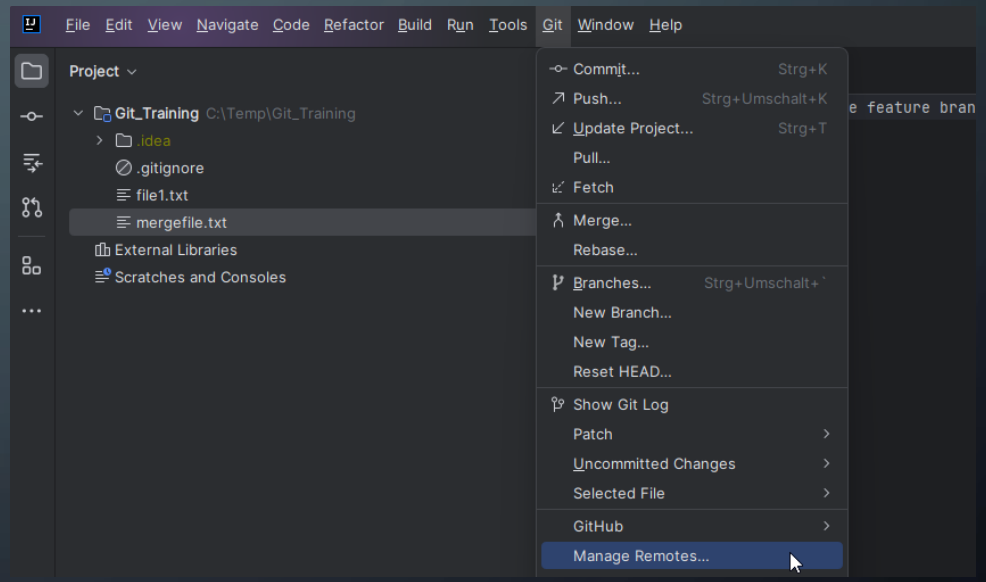
- Commit the change
- Push it to "origin"

EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git remote -v

GIT Training > git remote add origin https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git

GIT Training > git remote -v
origin  https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git (fetch)
origin  https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git (push)
```

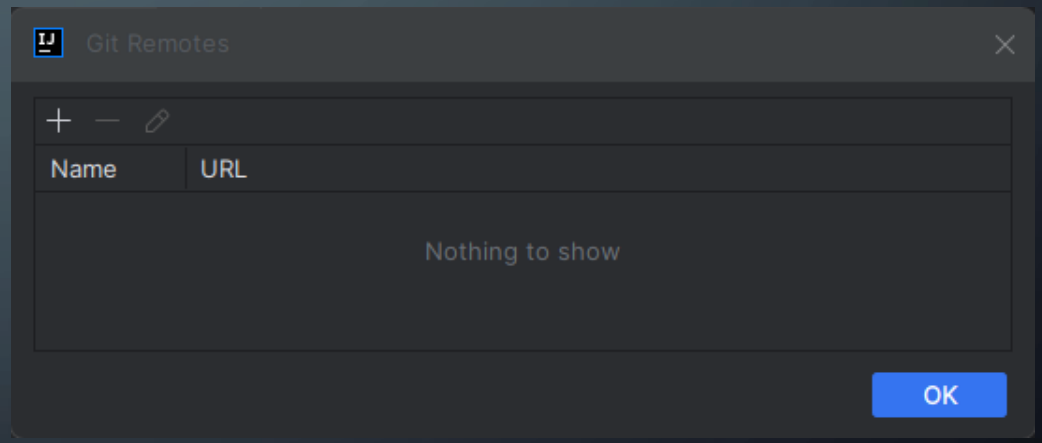
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

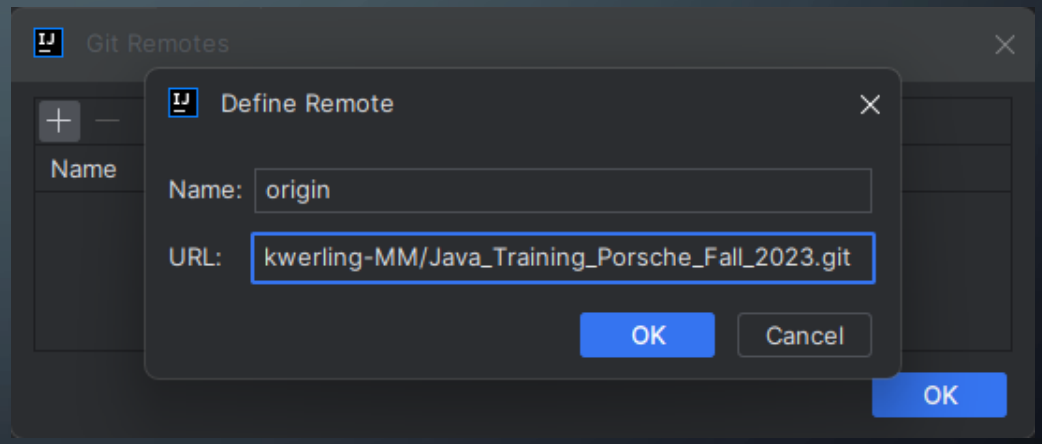
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

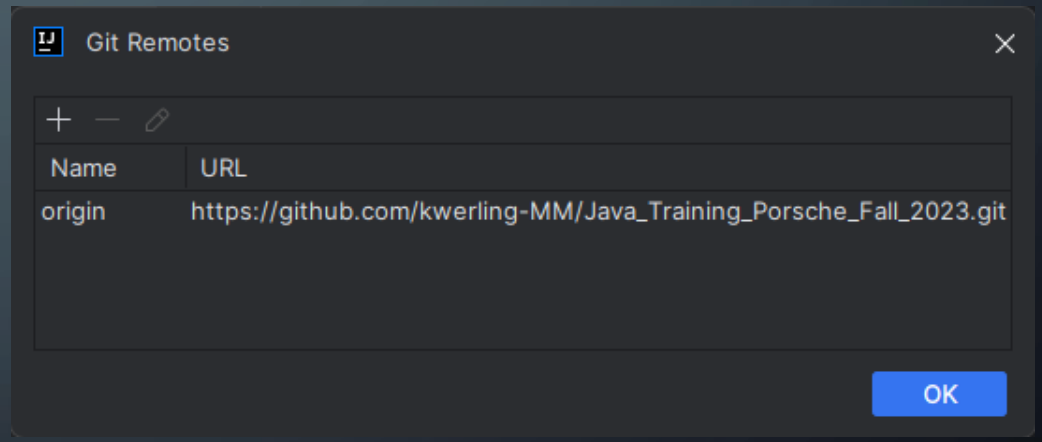
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

INTELLIJ

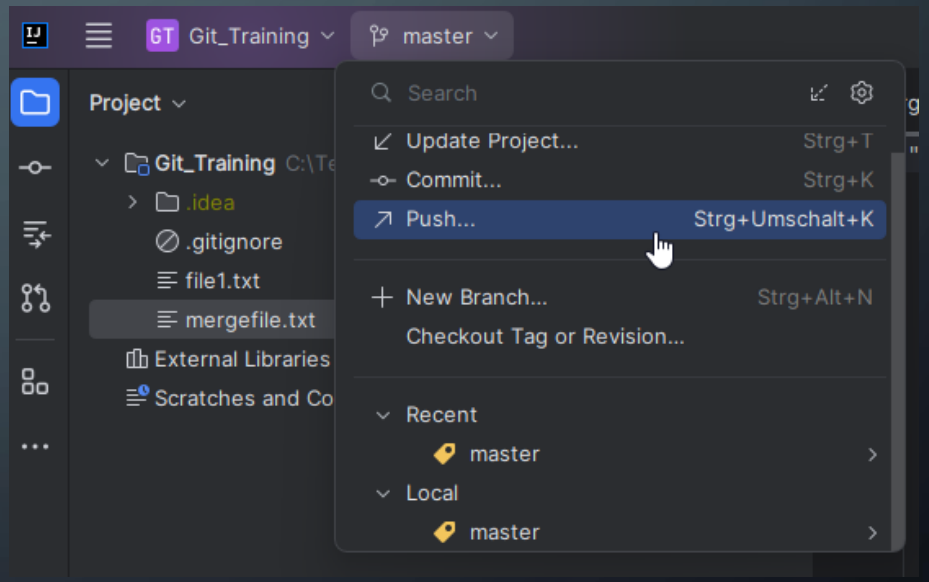


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


GIT Training > git push origin
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

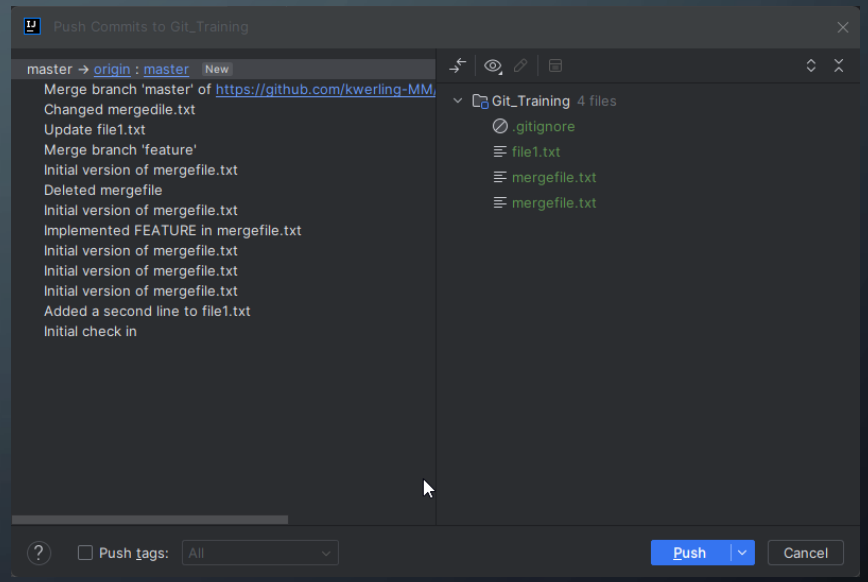
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

INTELLIJ

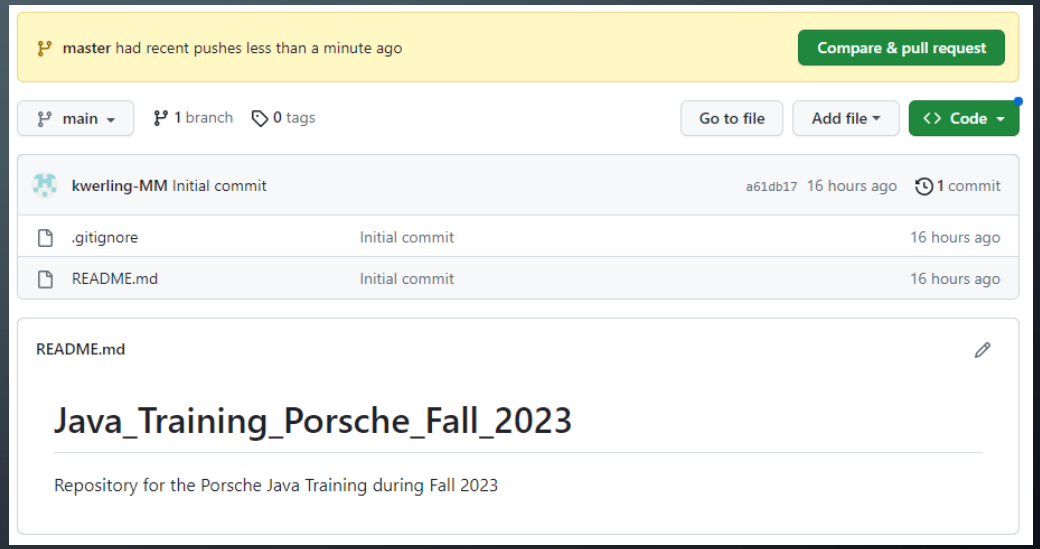


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git push origin master
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (23/23), 2.08 KiB | 710.00 KiB/s, done.
Total 23 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023/pull/new/master
remote:
To https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git
 * [new branch]      master -> master
```

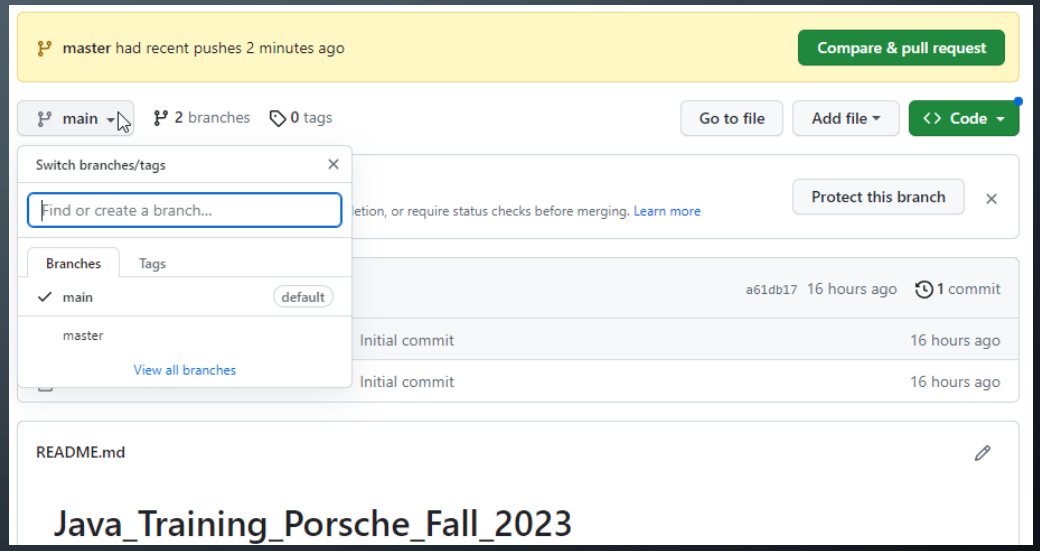
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

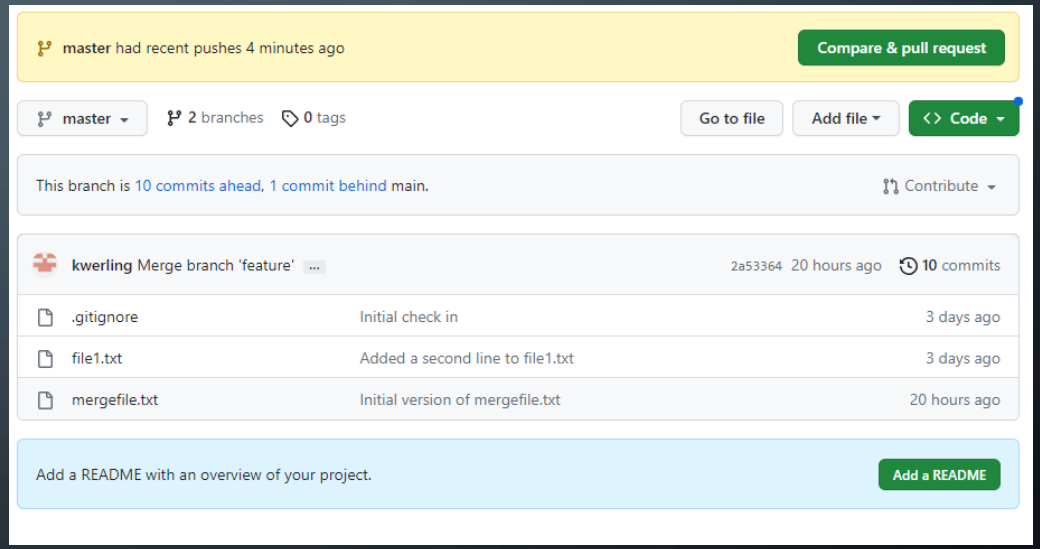
GITHUB
(AFTER REFRESH)



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

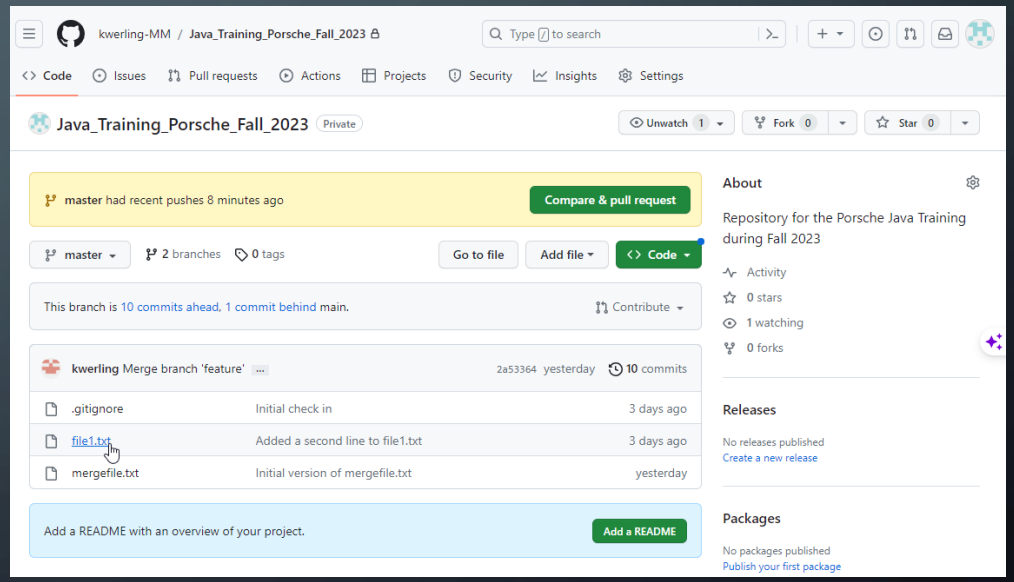
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

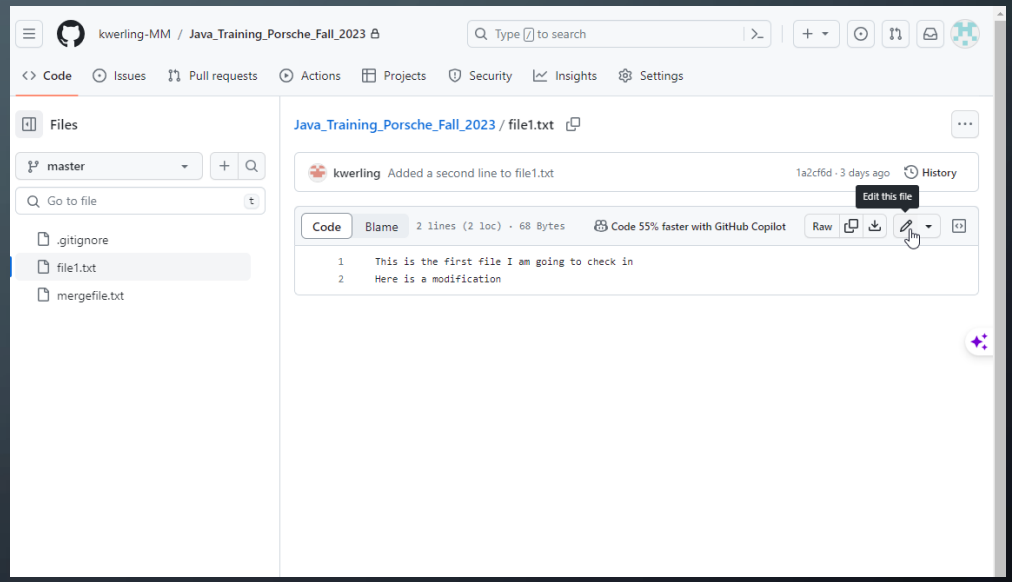
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

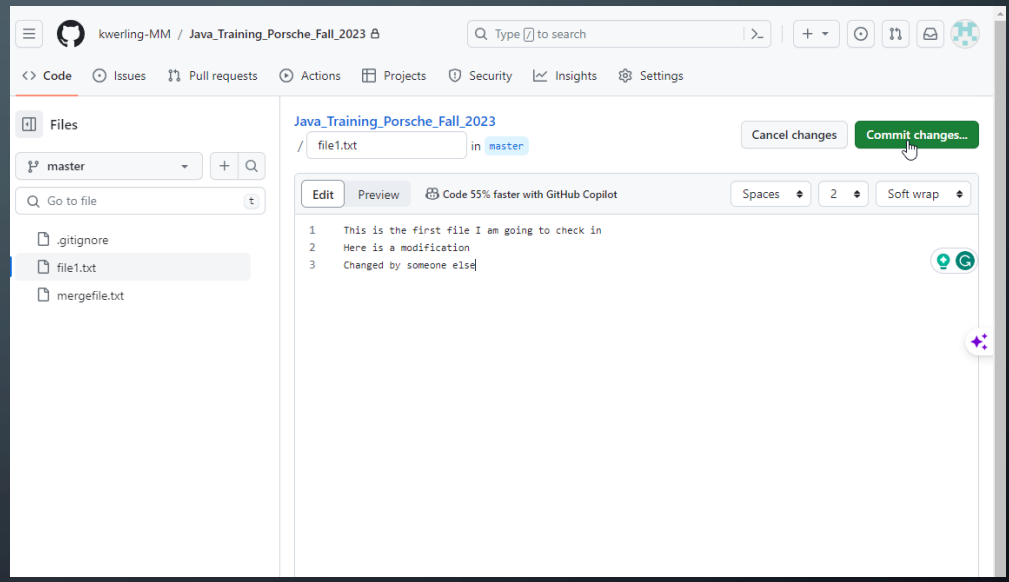
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

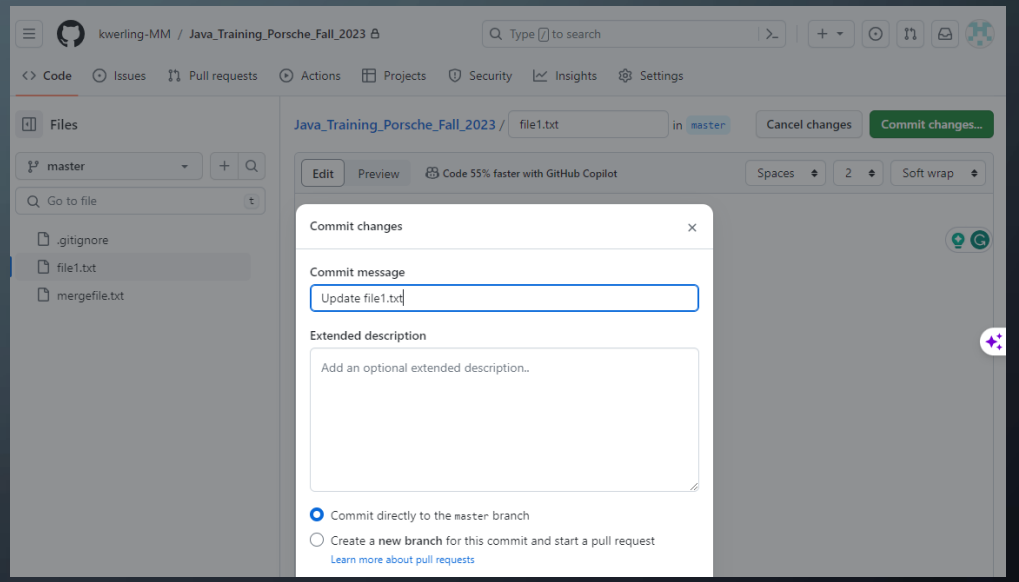
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

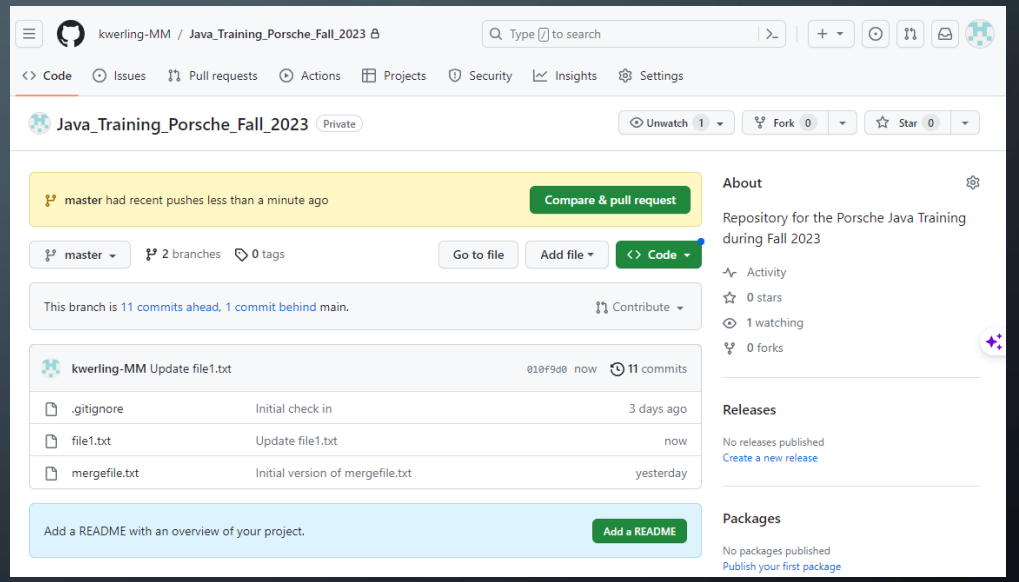
GITHUB



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

GITHUB

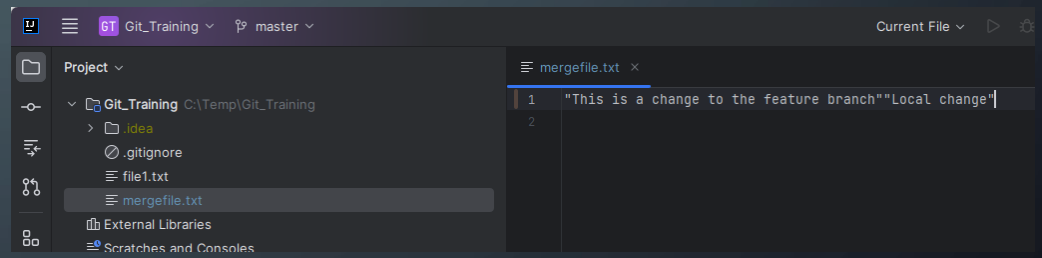


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > echo "Local change" >> mergefile.txt

GIT Training > type mergefile.txt
"This is a change to the feature branch""Local change"
```

INTELLIJ

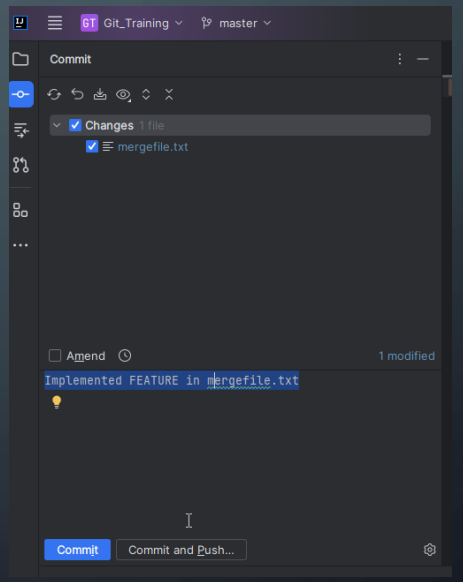


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git add --all

GIT Training > git commit -m "Changed mergedile.txt"
[master 0e6ebec] Changed mergedile.txt
 1 file changed, 1 insertion(+), 1 deletion(-)

GIT Training > git push origin master
To https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

INTELLIJ

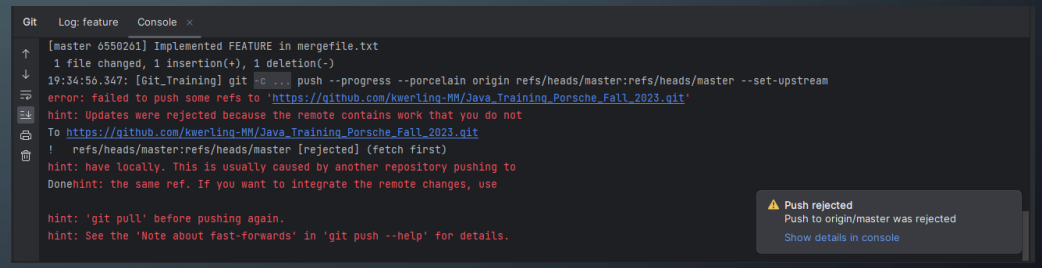


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git pull
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), 1.62 KiB | 184.00 KiB/s, done.
From https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023
 + a61db17...5187148 main       -> origin/main  (forced update)
   2a53364..010f9d0  master     -> origin/master
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> master


GIT Training > git push origin master
To https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

INTELLIJ

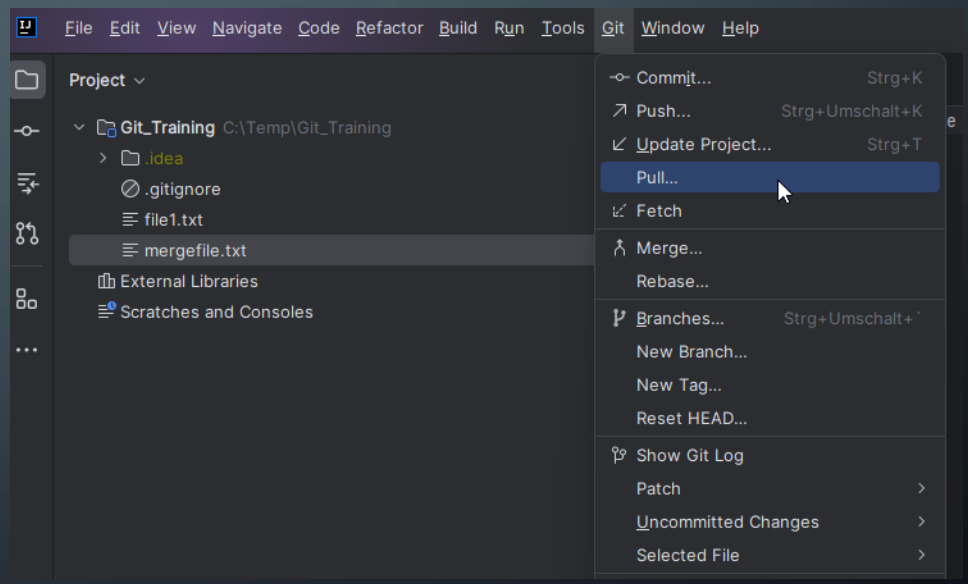


EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

```
GIT Training > git pull origin master
From https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023
 * branch            master     -> FETCH_HEAD
Merge made by the 'ort' strategy.
 file1.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

GIT Training > git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 609 bytes | 609.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/kwerling-MM/Java_Training_Porsche_Fall_2023.git
   010f9d0..66ba974  master -> master
```

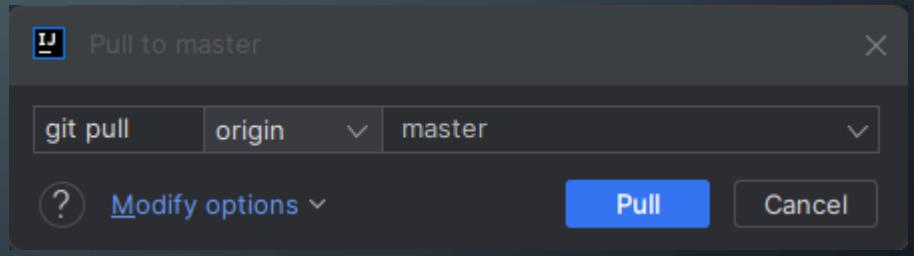
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

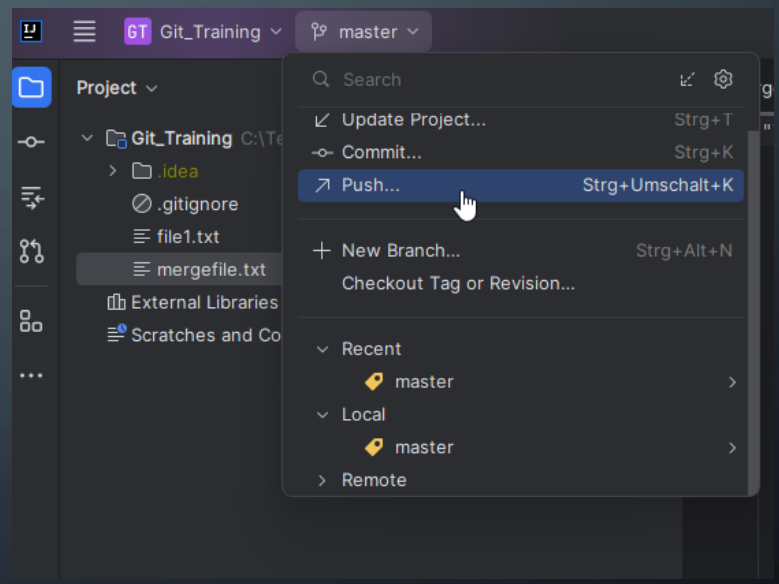
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

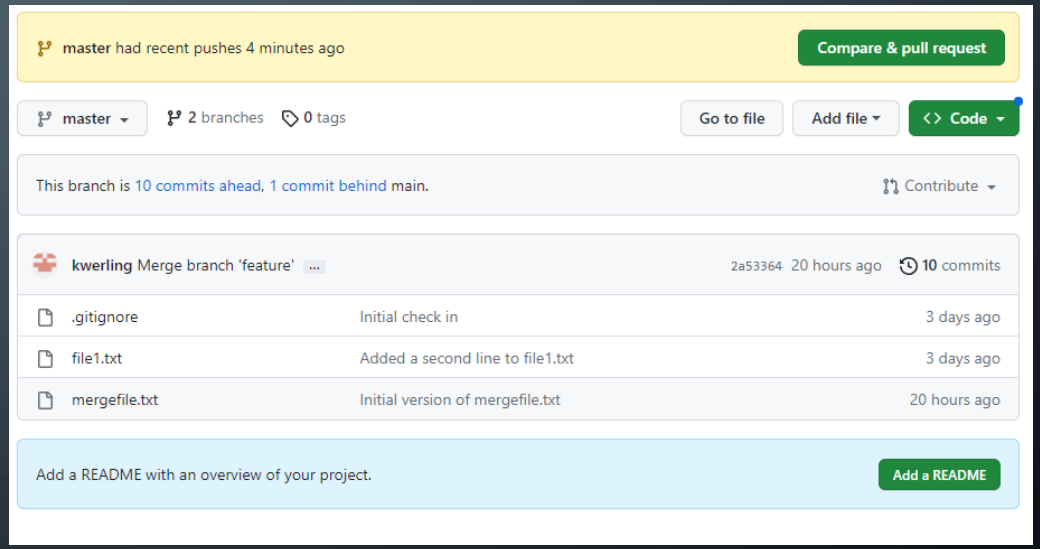
INTELLIJ



EXERCISE: REMOTE REPOSITORIES

COMMAND PROMPT

GITHUB



TAGS IN GIT

- Tags are used to mark specific points in the project's history, such as a release, a milestone, or any other important point.
- A commit can have multiple tags or none at all
- Use "git tag << tag >> << commit hash>>" to tag
- Use "git log" to find the commit hash
- "git show << commit hash >>" provides details on a commit

GIT WORKFLOWS

- There are many different workflows used with GIT
- The following is a summary of the more widely used ones

GIT WORKFLOWS

- There are many different workflows used with GIT
- The following is a summary of the more widely used ones

GIT WORKFLOWS

- In **Trunk-based** developers merge small, frequent updates to a core “trunk” or main branch.
- Best practice:
  - Develop in small batches
  - Usage of Feature flags
  - Implement comprehensive automated testing
  - Merge branches to the trunk at least once a day

GIT WORKFLOWS

- **Gitflow:** The Gitflow workflow is a popular workflow that is used by many teams. It consists of two main branches: the master branch and the develop branch. The master branch is the stable branch that contains the released code. The develop branch is the development branch that contains the latest code changes.

GIT WORKFLOWS

- **Feature branching**: The feature branching workflow is a simple workflow that is often used by small teams. It consists of creating a new branch for each feature that is being developed. Once the feature is complete, the branch is merged into the master branch.

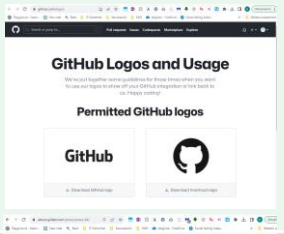
GIT WORKFLOWS

- **Forking workflow:** The forking workflow is a workflow that is often used for open source projects. It consists of creating a fork of the project repository on your own account. You then make changes to the fork and submit pull requests to the main repository.

GIT: BEST PRACTICES

1. **Use a consistent naming convention for your branches.** This will make it easier to identify and track your branches.
2. **Use a branching strategy that works for your team.** There are many different branching strategies, so find one that works well for your team and stick with it.
3. **Commit frequently and concisely.** This will make it easier to track your changes and revert to previous versions if needed. This will help you to keep your working tree clean and organized.
4. **Write meaningful commit messages.** Your commit messages should be clear and concise, and they should explain the changes that you have made.

IMAGES USED

Image	URL	License Info	Comment
git	https://git-scm.com/downloads/logos		
	https://github.com/logos		
GitLab	https://about.gitlab.com/press/press-kit/