Labor-Protokoll SEW

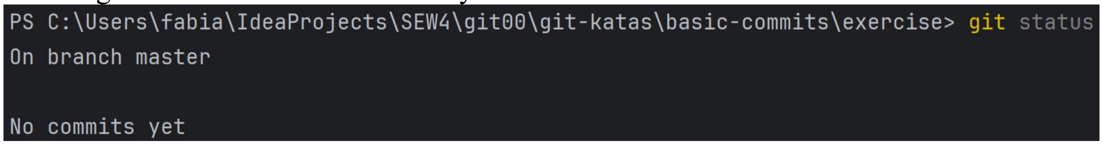
Name	**Fabian Ha**
4-stellige Login-Nummer	**1182**
Klasse	**4CN**
Datum der Übung	16.09.2024
Datum der Abgabe	27.09.2024
Übungsnummer	00
Auftraggeber	BRE/ZAI
Thema der Übung	**Git Einführung**

Inhaltsverzeichnis

1 Git-katas

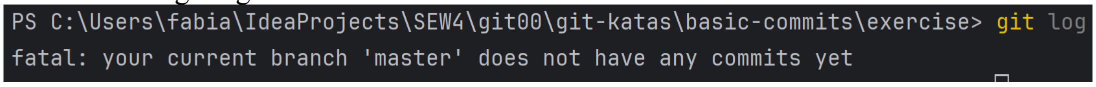
1.1 basic-commits

1. Use `git status` to see which branch you are on.



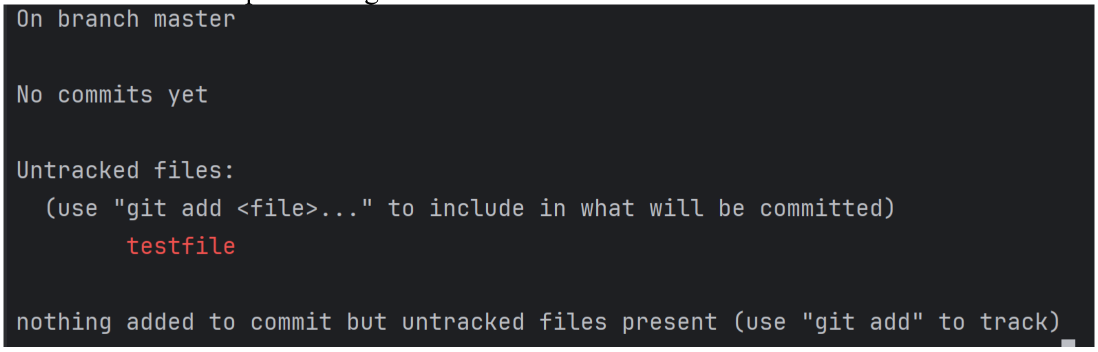
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master

No commits yet
```

2. What does `git log` look like?



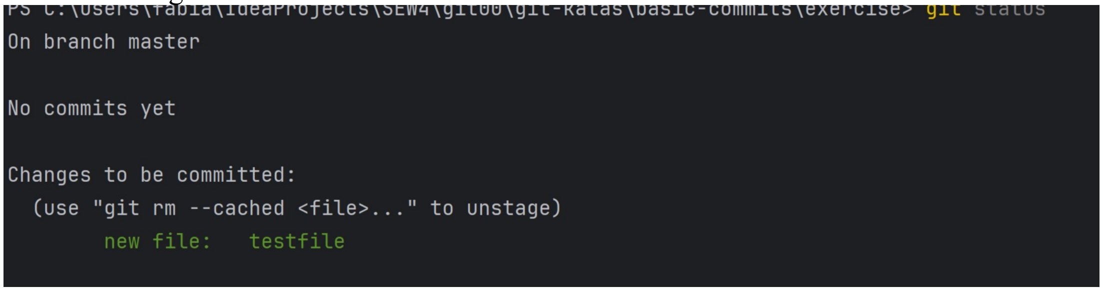
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git log
fatal: your current branch 'master' does not have any commits yet
```

3. Create a file
echo > testfile
4. What does the output from `git status` look like now?



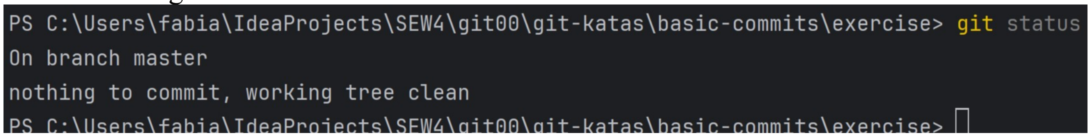
```
On branch master

No commits yet

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

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

5. `add` the file to the staging area
git add testfile
6. How does `git status` look now?



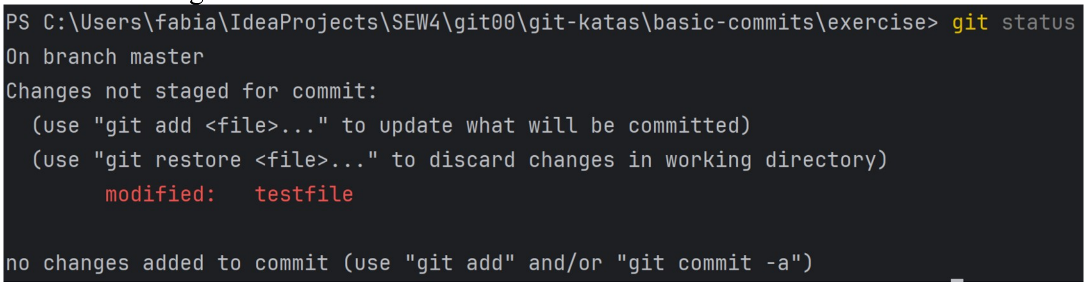
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master

No commits yet

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

7. `commit` the file to the repository
git commit -m "testfile committed"
8. How does `git status` look now?



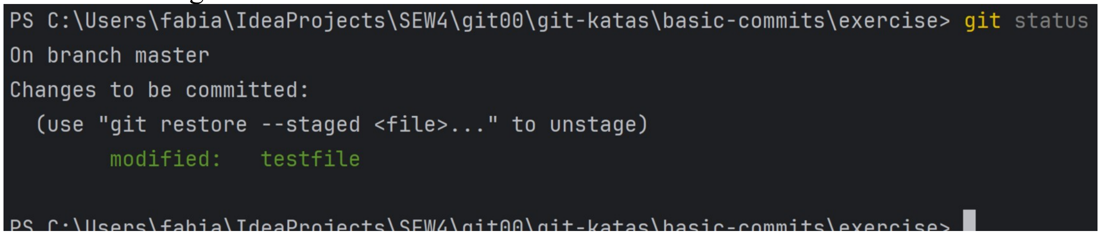
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise>
```

9. Change the content of the file you created earlier
echo "hallo" > testfile
10. What does `git status` look like now?

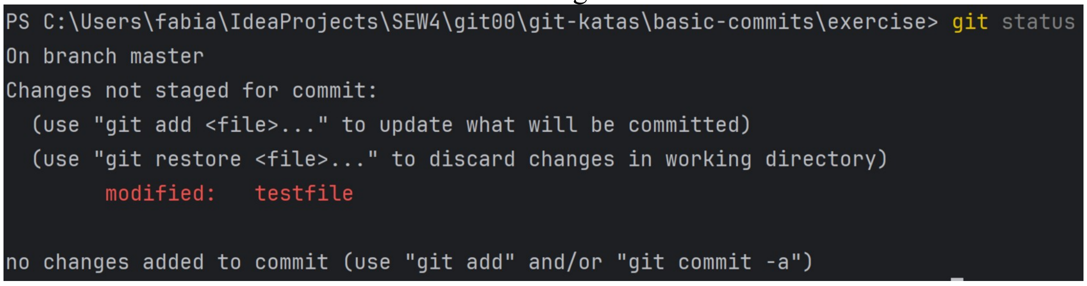


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   testfile

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

11. `add` the file change

git add testfile
12. What does `git status` look like now?


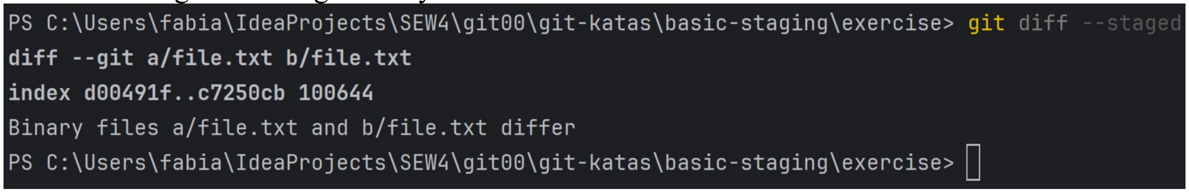
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   testfile

PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise>
```

13. Change the file again
echo "jonas" >> testfile
14. Make a `commit`
git commit -m "testfile changes committed"
15. What does the status` look like now? The `log`?


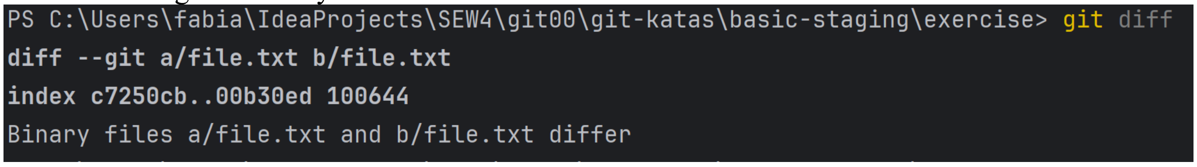
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-commits\exercise> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   testfile

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

16. Add and commit the newest change
git add testfile
git commit -m "changed testfile again"

1.2 basic-staging

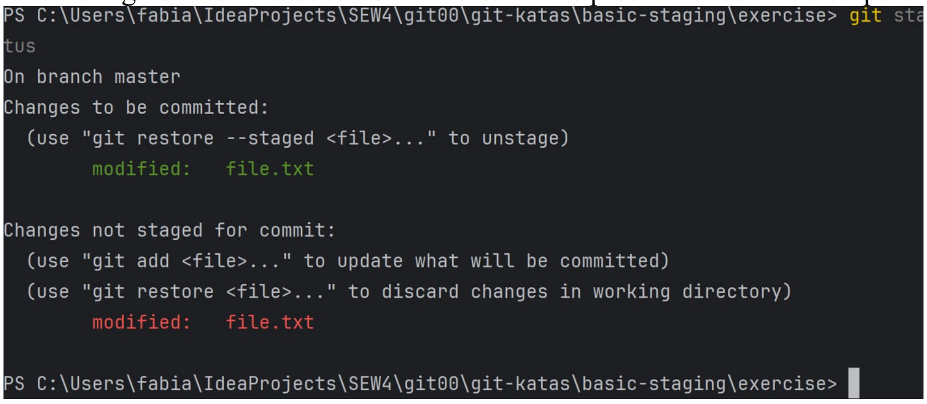
1. What's the content of `file.txt`?
1
2. Overwrite the content in `file.txt`: `echo 2 > file.txt` to change the state of your file in the working directory (or `sc file.txt '2'` in PowerShell)
echo 2 > file.txt
3. What does `git diff` tell you?
Tells us the difference between the staging area and working area
4. What does `git diff --staged` tell you? why is this blank?
Tells us the things that got staged. Its blank because we didn’t add anything to the staging area.
5. Run `git add file.txt` to stage your changes from the working directory.
git add file.txt
6. What does `git diff` tell you?
Nothing
7. What does `git diff --staged` tell you?


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git diff --staged
diff --git a/file.txt b/file.txt
index d00491f..c7250cb 100644
Binary files a/file.txt and b/file.txt differ
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise>
```

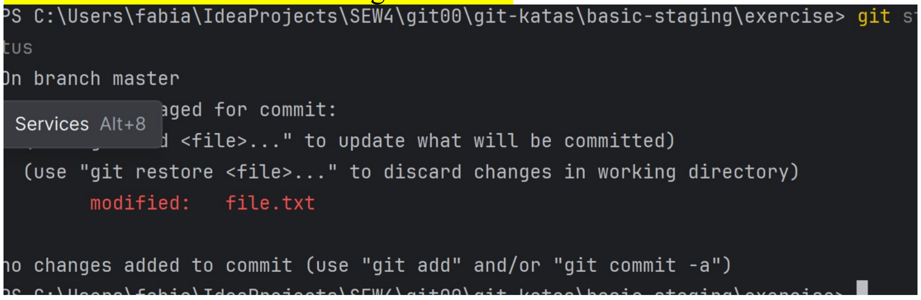
8. Overwrite the content in `file.txt`: `echo 3 > file.txt` to change the state of your file in the working directory (or `sc file.txt '3'` in PowerShell).
echo 3 > file.txt
9. What does `git diff` tell you?


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git diff
diff --git a/file.txt b/file.txt
index c7250cb..00b30ed 100644
Binary files a/file.txt and b/file.txt differ
```

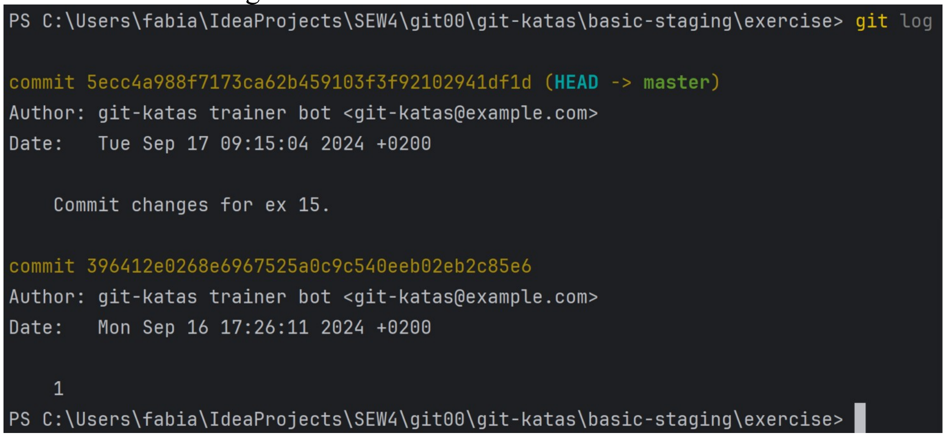
10. What does `git diff --staged` tell you?

the same thing as 7.
11. Explain what is happening
git diff –staged shows the difference in staged files and didn’t change in 10. because I didn’t add the file again to the staged area
12. Run `git status` and observe that `file.txt` are present twice in the output.


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git sta
tus
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.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:   file.txt

PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise>
```

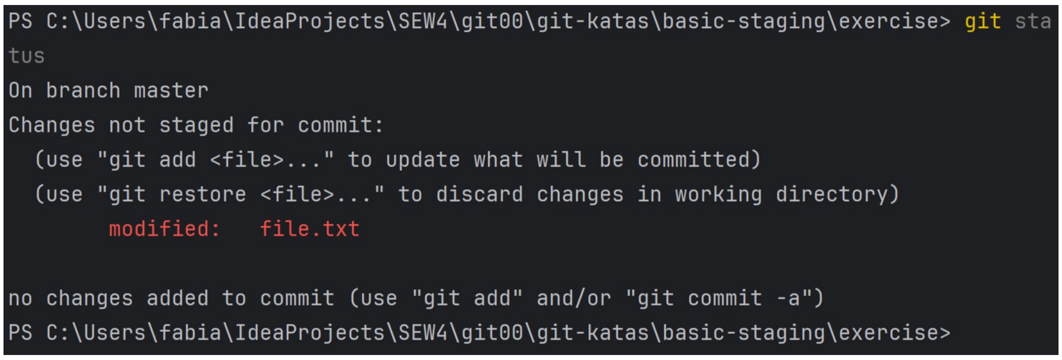
13. Run `git restore --staged file.txt` to unstage the change
14. What does `git status` tell you now?
It removed the committed change to file.txt


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git s
tus
On branch master
                  aged for commit:
                  d <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

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

15. Stage the change and make a commit
git add file.txt
git commit -m "Commit changes for ex 15."
16. What does the log look like?


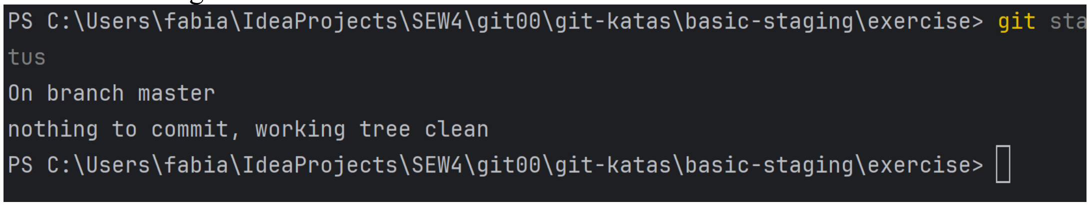
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git log

commit 5ecc4a988f7173ca62b459103f3f92102941df1d (HEAD -> master)
Author: git-katas trainer bot <git-katas@example.com>
Date:   Tue Sep 17 09:15:04 2024 +0200

    Commit changes for ex 15.

commit 396412e0268e6967525a0c9c540eeb02eb2c85e6
Author: git-katas trainer bot <git-katas@example.com>
Date:   Mon Sep 16 17:26:11 2024 +0200

    1
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise>
```

17. Overwrite the content in `file.txt`: `echo 4 > file.txt` (or `sc file.txt '4'` in PowerShell)
18. What is the content of `file.txt`?
4
19. What does `git status` tell us?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git sta
tus
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:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise>
```

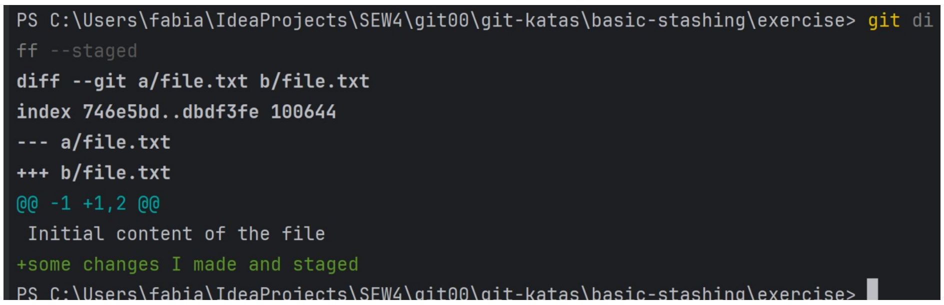
20. Run `git restore file.txt`
21. What is the content of `file.txt`?
3
22. What does `git status` tell us?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise> git sta
tus
On branch master
nothing to commit, working tree clean
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-staging\exercise>
```

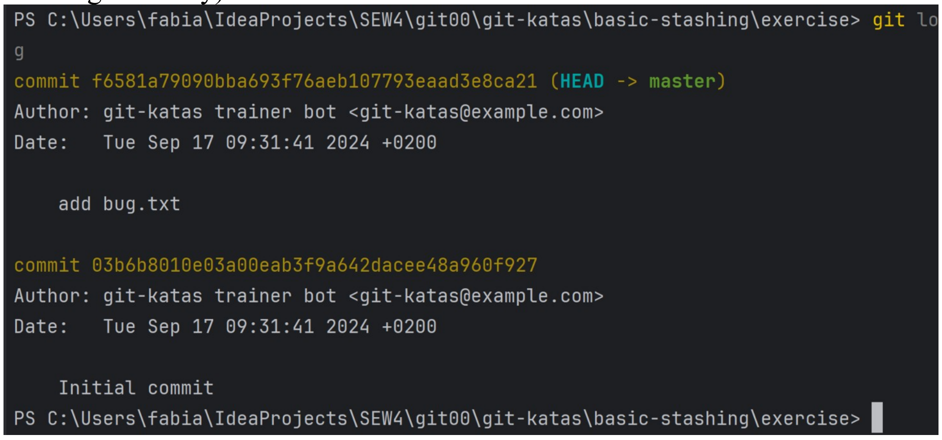
1.3 basic-branching

1. Explore the repo
 1. What work do you have in the working directory?
bug.txt
file.txt
fix.txt
   2. What work do you have staged ?
file.txt



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise> git di
ff --staged
diff --git a/file.txt b/file.txt
index 746e5bd..dbdf3fe 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1,2 @@
 Initial content of the file
+some changes I made and staged
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise>
```

   3. What does the commit log look like ?
   >*Notice that file.txt has some staged changes (i.e. changes in the index) and unstaged changes (changes in the working directory)*



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise> git lo
g
commit f6581a79090bba693f76aeb107793eaad3e8ca21 (HEAD -> master)
Author: git-katas trainer bot <git-katas@example.com>
Date:   Tue Sep 17 09:31:41 2024 +0200

    add bug.txt

commit 03b6b8010e03a00eab3f9a642dacee48a960f927
Author: git-katas trainer bot <git-katas@example.com>
Date:   Tue Sep 17 09:31:41 2024 +0200

    Initial commit
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise>
```

2. Use `git stash` to stash your current work.
   1. Now, what work do you have in the working directory?

nothing to commit

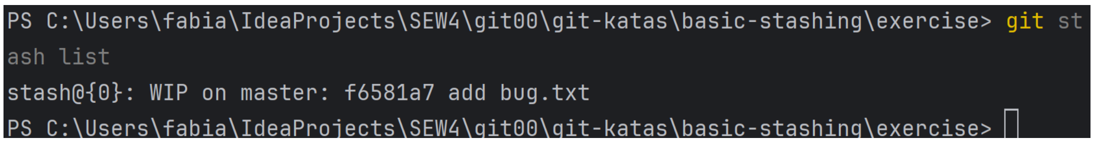
2. What work do you have staged ?

Nothing

3. What does the commit log look like ?

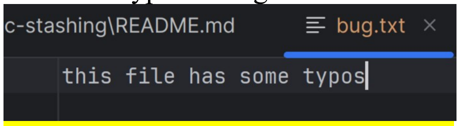
Did not change from 1.3.

4. What does the stash list look like ?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise> git st
ash list
stash@{0}: WIP on master: f6581a7 add bug.txt
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise>
```

3. Fix the typos in bug.txt on master and commit your changes.



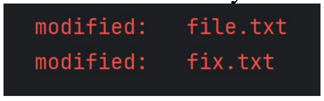
```
this file has some typos
```

git commit -m "fixed typos in bug.txt"

4. Now to get back to your work, apply the stash to master.

git stash apply

1. What work do you have in the working directory?



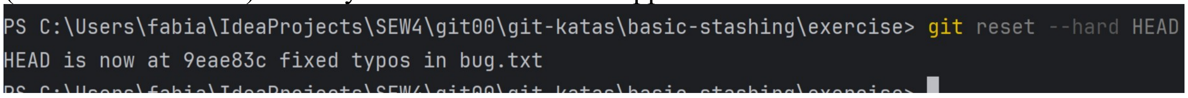
```
  modified:   file.txt
  modified:   fix.txt
```

2. What work do you have staged ?

Nothing

>*Oops. All our changes are unstaged now. This may be undesirable and unexpected*

5. Undo our changes with `git reset --hard HEAD`. This is an unsafe command as it will remove files from your index and working directory permanently, but we have our changes safely stashed so we're ok. Review the [reset](reset/README.md) kata if you're unsure of what happens here.

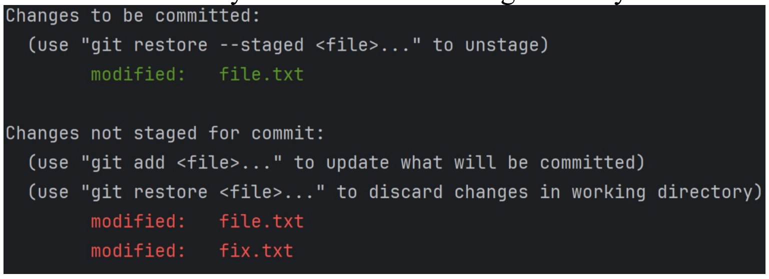


```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise> git reset --hard HEAD
HEAD is now at 9eae83c fixed typos in bug.txt
```

6. Apply the stash to master with the `--index` option.

git stash apply --index

1. What work do you have in the working directory?



```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.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:   file.txt
        modified:   fix.txt
```

2. What work do you have staged ?

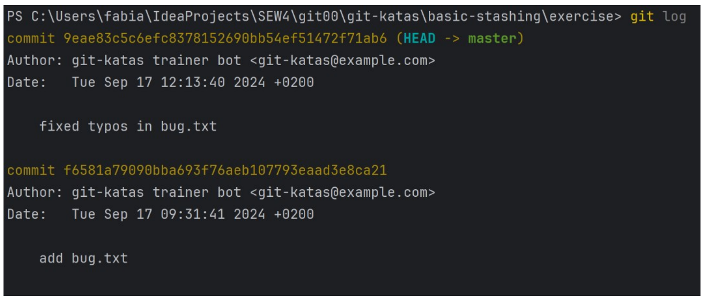
modified file.txt

>*Ok, back to where we were!*

7. We won't need the stash anymore. Drop it.

git stash drop

1. What does the stash list look like ? empty
2. What does the commit log look like ?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\basic-stashing\exercise> git log
commit 9eae83c5c6efc8378152690bb54ef51472f71ab6 (HEAD -> master)
Author: git-katas trainer bot <git-katas@example.com>
Date:   Tue Sep 17 12:13:40 2024 +0200

    fixed typos in bug.txt

commit f6581a79090bba693f76aeb107793eaad3e8ca21
Author: git-katas trainer bot <git-katas@example.com>
Date:   Tue Sep 17 09:31:41 2024 +0200

    add bug.txt
```

1.4 Fast-forward Merge

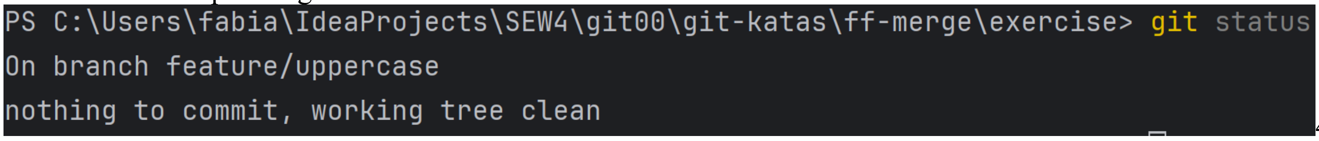
1. Create a (feature)branch called `feature/uppercase` (yes, `feature/uppercase` is a perfectly legal branch name, and a common convention).

git branch feature/uppercase

2. Switch to this branch

git checkout feature/uppercase

3. What is the output of `git status`?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git status
On branch feature/uppercase
nothing to commit, working tree clean
```

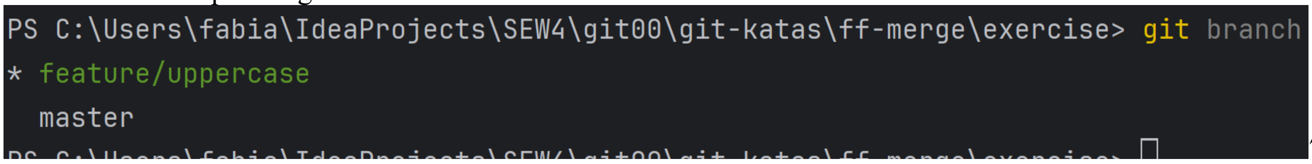
4. Edit the greeting.txt to contain an uppercase greeting

echo "HELLO" > .\greeting.txt

5. Add `greeting.txt` files to staging area and commit

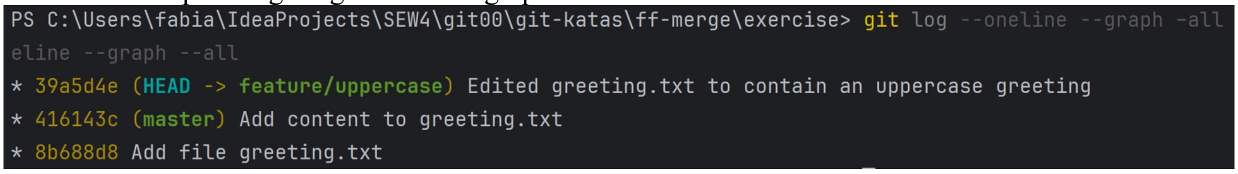
git add .\greeting.txt
git commit -m "Edited greeting.txt to contain an uppercase greeting"

6. What is the output of `git branch`?



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git branch
* feature/uppercase
  master
```

7. What is the output of `git log --oneline --graph –all`



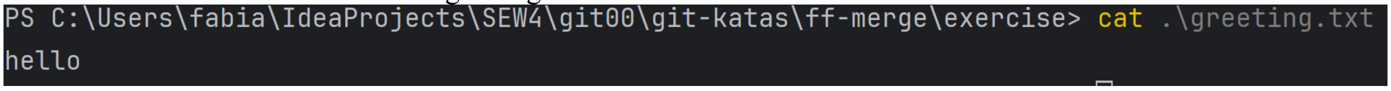
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git log --oneline --graph -all
eline --graph --all
* 39a5d4e (HEAD -> feature/uppercase) Edited greeting.txt to contain an uppercase greeting
* 416143c (master) Add content to greeting.txt
* 8b688d8 Add file greeting.txt
```

Remember: You want to update the master branch so it also has all the changes currently on the feature branch. The command 'git merge [branch name]' takes one branch as argument from which it takes changes. The branch pointed to by HEAD (currently checked out branch) is then updated to also include these changes.

8. Switch to the `master` branch

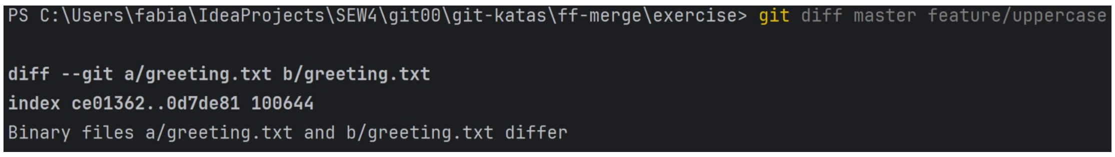
git checkout master

9. Use `cat` to see the contents of the greetings



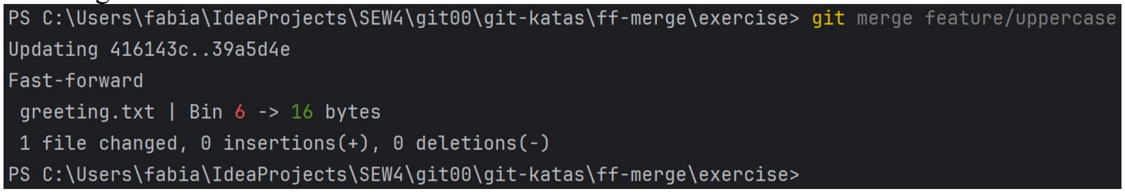
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> cat .\greeting.txt
hello
```

10. Diff the branches



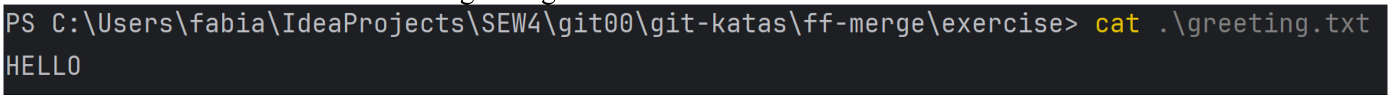
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git diff master feature/uppercase

diff --git a/greeting.txt b/greeting.txt
index ce01362..0d7de81 100644
Binary files a/greeting.txt and b/greeting.txt differ
```

11. Merge the branches



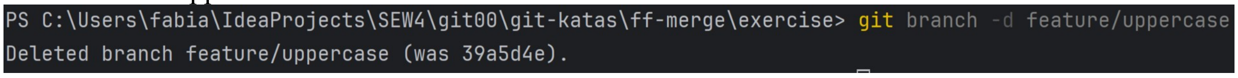
```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git merge feature/uppercase
Updating 416143c..39a5d4e
Fast-forward
 greeting.txt | Bin 6 -> 16 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise>
```

12. Use `cat` to see the contents of the greetings



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> cat .\greeting.txt
HELLO
```

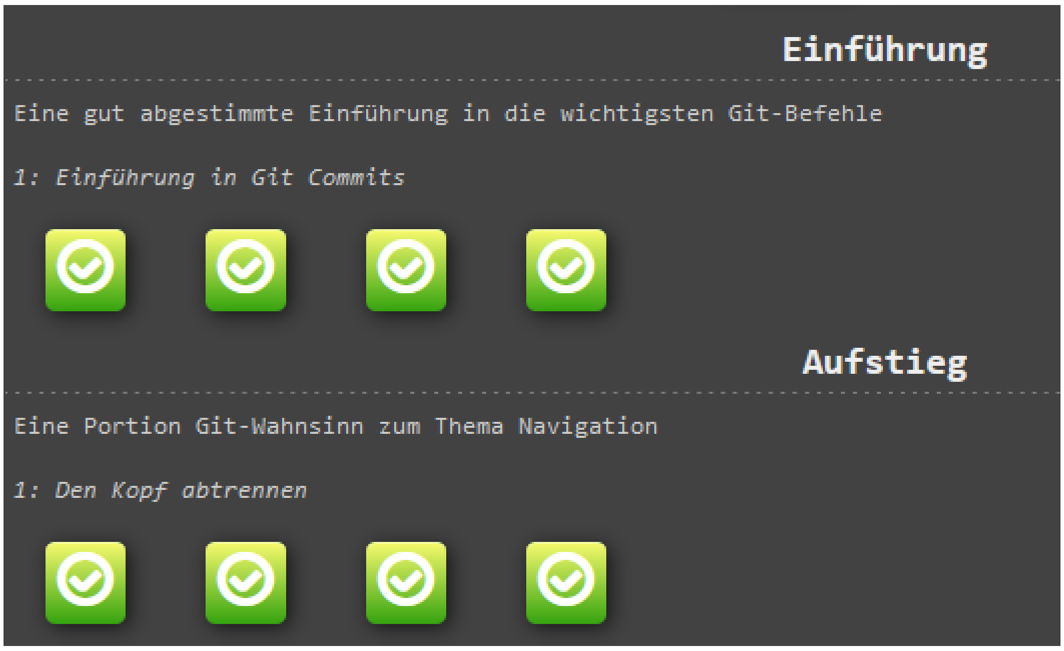
13. Delete the uppercase branch



```
PS C:\Users\fabia\IdeaProjects\SEW4\git00\git-katas\ff-merge\exercise> git branch -d feature/uppercase
Deleted branch feature/uppercase (was 39a5d4e).
```

2 Learn Git Branching

2.1 Haupt Zeile 1 und 2



Einführung

Eine gut abgestimmte Einführung in die wichtigsten Git-Befehle

1: Einführung in Git Commits

✓ ✓ ✓ ✓

Aufstieg

Eine Portion Git-Wahnsinn zum Thema Navigation

1: Den Kopf abtrennen

✓ ✓ ✓ ✓