Configuration Management Assignments Week #1

Zoe Arendsz – 350933

1. What is the meaning of the symbols < and > ?

   The <> symbols are next to oranges and grapefruits.
   Those are what the two .txt files have in common.

2. In the output you can see "2d1" and "4c3,4". What do the letters a, d and c stand for?

   The letters a means add, d means delete and c means copy.

3. What is the meaning of the + and – signs, for example “-oranges”?

   It shows the difference between the two files. That one file is -orange (minus orange) and the +grapefruit means that it includes the grapefruit in that file.

4. Issue a git status –s. Explain what you see.

   Shows nothing because I didn’t do anything with the file.

5. What are the possible states of a file in Git?

- Untracked
- Tracked
- Modified
- Repo

6. Issue a git status –s. Explain what you see.

   Shows that the file has been modified.

7. Do a git diff. Again, explain what you see.

   It shows the difference between the files.

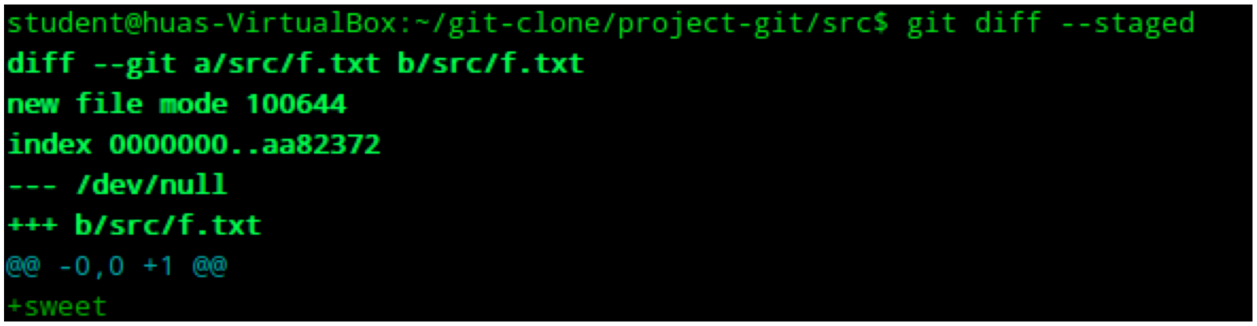
8. Do a git commit –m “b.txt added”. Again, do a git status –s. Explain what you see.

   I see M b.txt (b.txt file has been modified)

9. What is the difference between a git diff en git diff - -staged?

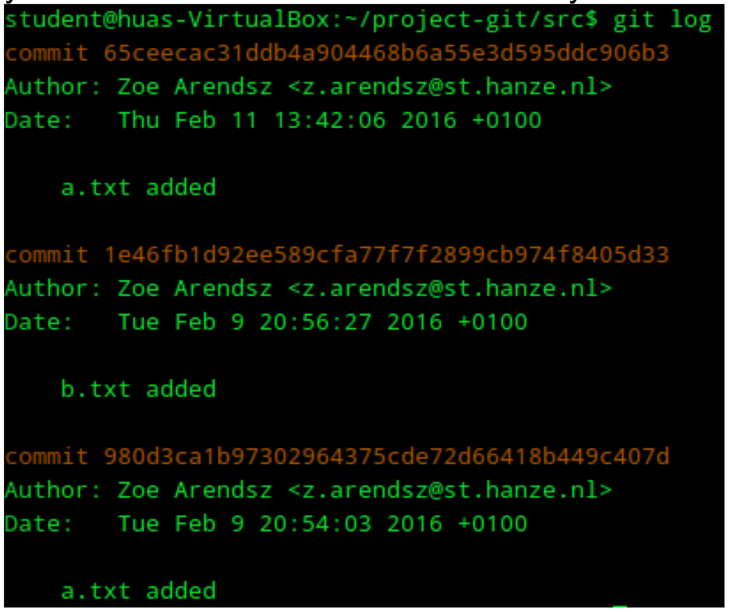
   Staged is like a cached version of the file.

10. Do a git add b.txt. Do a git diff --staged. Explain what you see.



```
student@huas-VirtualBox:~/git-clone/project-git/src$ git diff --staged
diff --git a/src/f.txt b/src/f.txt
new file mode 100644
index 0000000..aa82372
--- /dev/null
+++ b/src/f.txt
@@ -0,0 +1 @@
+sweet
```

11. Do a git commit –m “b.txt v2 added”. Do a git status –s. Use git log to view all your commits. Make a screenshot of all your commits and add this to your report.



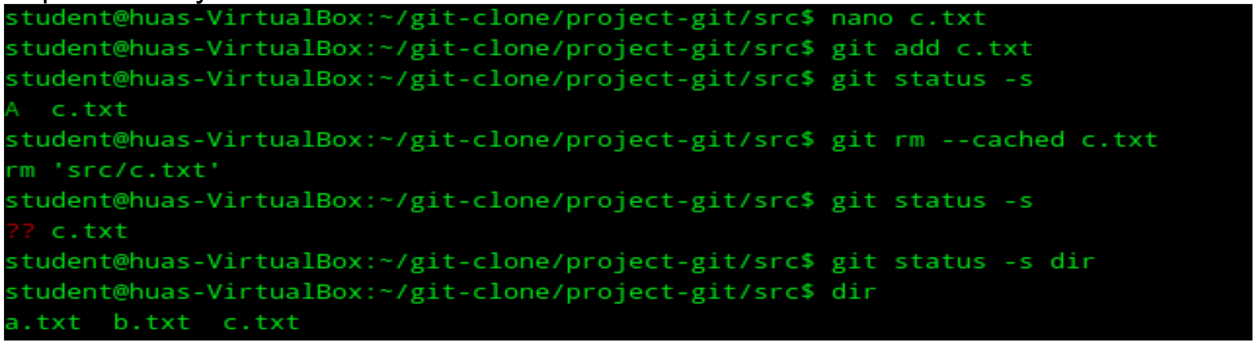
```
student@huas-VirtualBox:~/project-git/src$ git log
commit 65ceecac31ddb4a904468b6a55e3d595ddc906b3
Author: Zoe Arendsz <z.arendsz@st.hanze.nl>
Date:   Thu Feb 11 13:42:06 2016 +0100

    a.txt added

commit 1e46fb1d92ee589cfa77f7f2899cb974f8405d33
Author: Zoe Arendsz <z.arendsz@st.hanze.nl>
Date:   Tue Feb 9 20:56:27 2016 +0100

    b.txt added

commit 980d3ca1b97302964375cde72d66418b449c407d
Author: Zoe Arendsz <z.arendsz@st.hanze.nl>
Date:   Tue Feb 9 20:54:03 2016 +0100

    a.txt added
```

1.3

1. What do you see? What is the meaning of –cached?

   To store away.

2. Explain what you see.

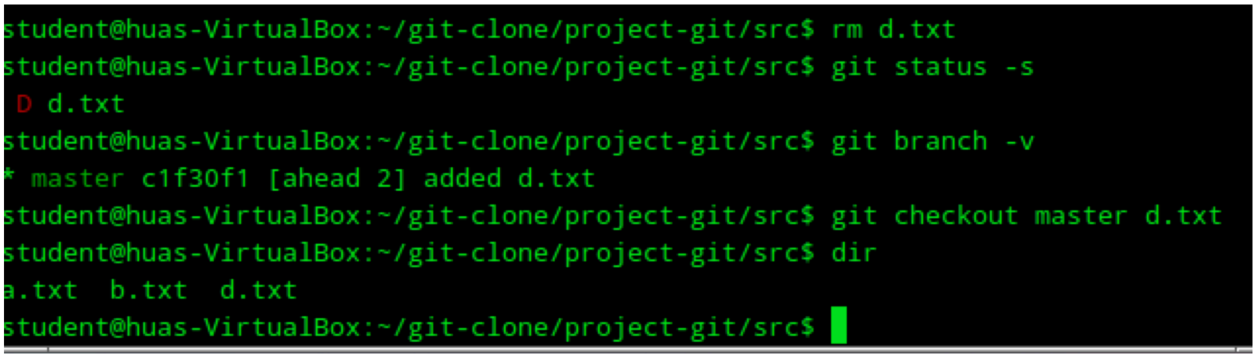


```
student@huas-VirtualBox:~/git-clone/project-git/src$ nano c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git add c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git status -s
A  c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git rm --cached c.txt
rm 'src/c.txt'
student@huas-VirtualBox:~/git-clone/project-git/src$ git status -s
?? c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git status -s dir
student@huas-VirtualBox:~/git-clone/project-git/src$ dir
a.txt  b.txt  c.txt
```

3. Is Git aware of the fact that c.txt is removed from your working directory?
   Yes, Git is aware. It only displayes M d.txt. The M is in red color font.

4. Is Git aware of the fact that d.txt was deleted?
   Yes, it displays D d.txt. The D is in red color font.

5. Do you see the file d.txt is back again in your working directory?
   Yes.



```
student@huas-VirtualBox:~/git-clone/project-git/src$ rm d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git status -s
 D d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git branch -v
* master c1f30f1 [ahead 2] added d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git checkout master d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ dir
a.txt  b.txt  d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ 
```

1.4

1. On which branch are you working now?

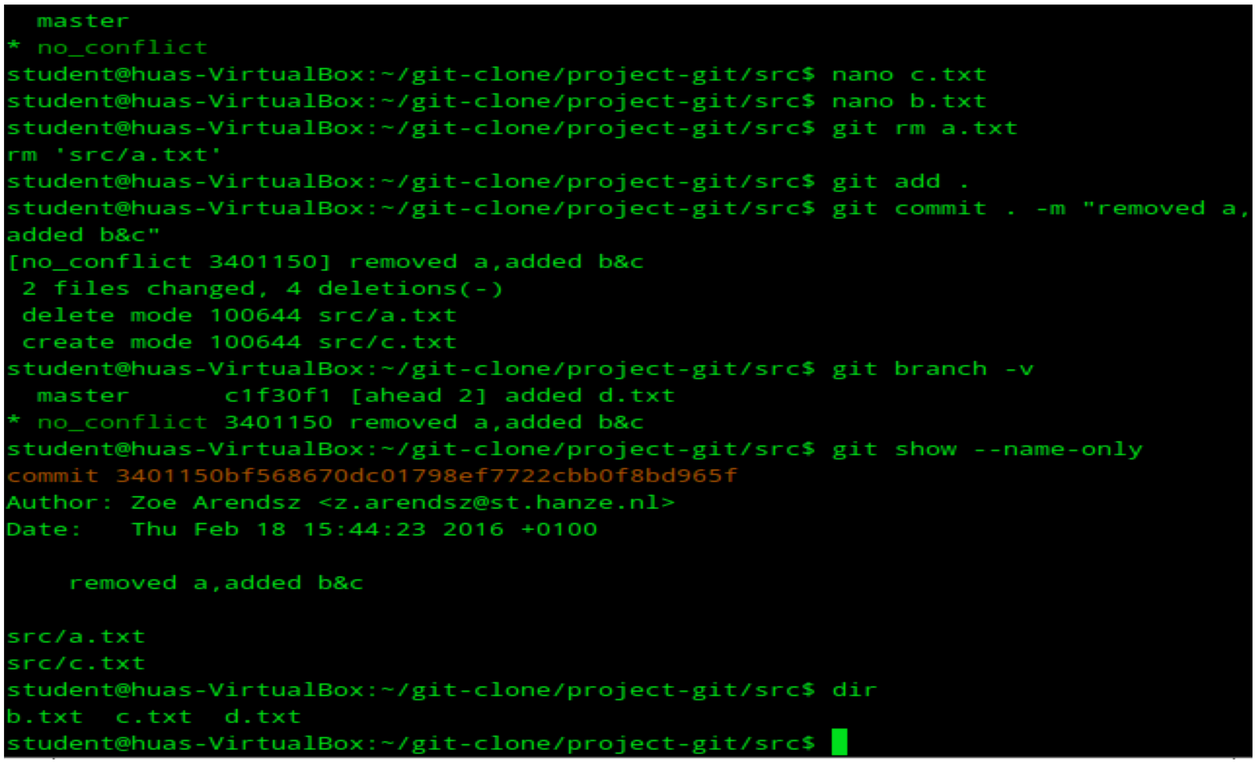
   * master c1f30f1 [ahead 2] added d.txt

2. Which files do you see?

   a.txt b.txt d.txt

3. Which files do you see?

   b.txt c.txt d.txt



```
  master
* no_conflict
student@huas-VirtualBox:~/git-clone/project-git/src$ nano c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ nano b.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git rm a.txt
rm 'src/a.txt'
student@huas-VirtualBox:~/git-clone/project-git/src$ git add .
student@huas-VirtualBox:~/git-clone/project-git/src$ git commit . -m "removed a,
added b&c"
[no_conflict 3401150] removed a,added b&c
 2 files changed, 4 deletions(-)
 delete mode 100644 src/a.txt
 create mode 100644 src/c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git branch -v
  master      c1f30f1 [ahead 2] added d.txt
* no_conflict 3401150 removed a,added b&c
student@huas-VirtualBox:~/git-clone/project-git/src$ git show --name-only
commit 3401150bf568670dc01798ef7722cbb0f8bd965f
Author: Zoe Arendsz <z.arendsz@st.hanze.nl>
Date:   Thu Feb 18 15:44:23 2016 +0100

    removed a,added b&c

src/a.txt
src/c.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ dir
b.txt  c.txt  d.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ 
```

4. What do you see?

   The commit viewer window opens, where I can see the files and check the tree and branches.

5. Do a dir on the src directory. Explain what you see.

   The files and folders in the src folder, which are: a.txt b.txt and git-clone folder.

6. Write down the hash-values of the latest commits in both branches.

   * master 8455e81 added a & b.

   no_conflict 53a2a37 removed a, added b&c.

7. What will happen in case the no-conflict branch and the master branch are merged?

   It would merge the files in the no_conflict and the master branch together.

8. Can you see that it is a fast forward merge? What is the meaning of “fast forward merge”?

   It means that that all of the commits we did will be merged in a linear form.

9. What strikes you? What do you see using gitk?

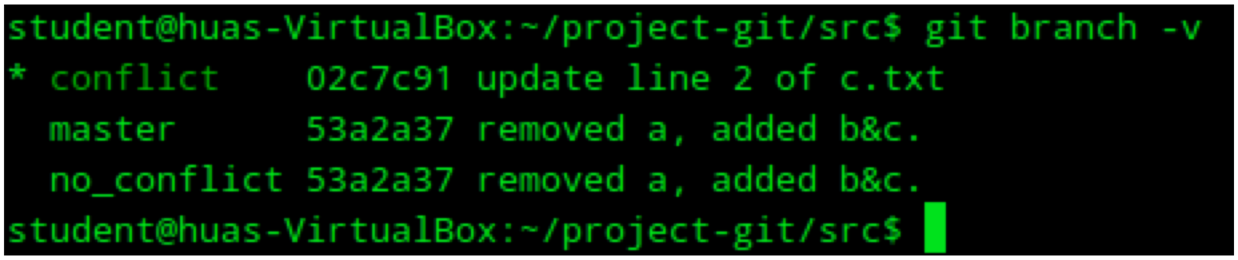
   I see that master and no_conflict are on the same line, and the commits are below each other in a linear form because I merged them.

1.5

1. Explain in which situation this is likely to happen.

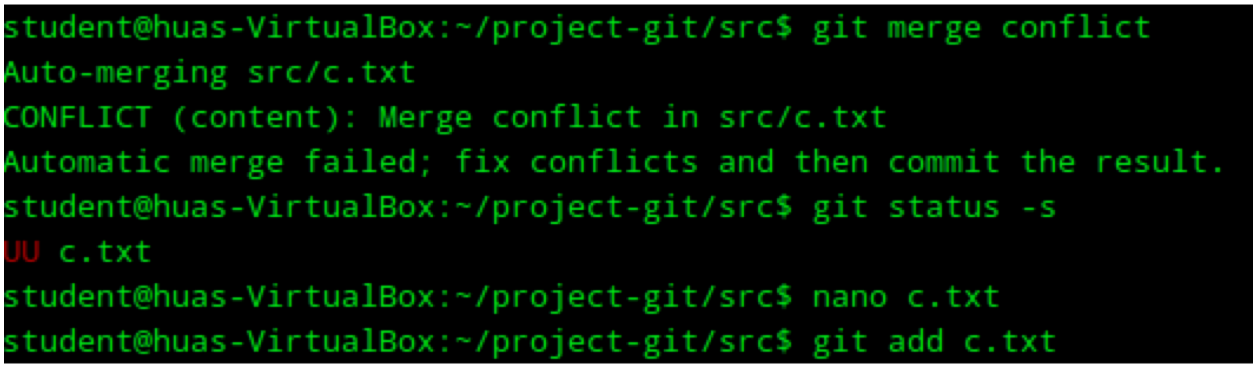
   When two people try to merge at the same time. Git cannot decide which file is the best, and prompts an error.

2. Write down the hash-values of the last commits on both branches.



```
student@huas-VirtualBox:~/project-git/src$ git branch -v
* conflict    02c7c91 update line 2 of c.txt
  master      53a2a37 removed a, added b&c.
  no_conflict 53a2a37 removed a, added b&c.
student@huas-VirtualBox:~/project-git/src$
```

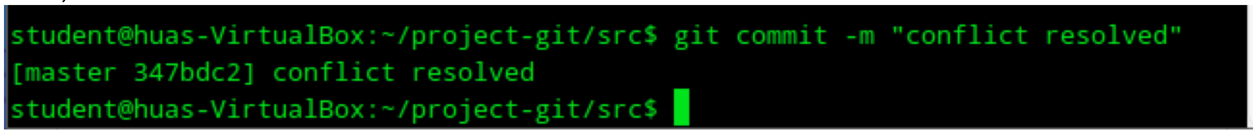
3. Make a screenshot of the output. Explain what you see.



```
student@huas-VirtualBox:~/project-git/src$ git merge conflict
Auto-merging src/c.txt
CONFLICT (content): Merge conflict in src/c.txt
Automatic merge failed; fix conflicts and then commit the result.
student@huas-VirtualBox:~/project-git/src$ git status -s
UU c.txt
student@huas-VirtualBox:~/project-git/src$ nano c.txt
student@huas-VirtualBox:~/project-git/src$ git add c.txt
```

4. Write down the hash-value of the commit. Is this a new commit?

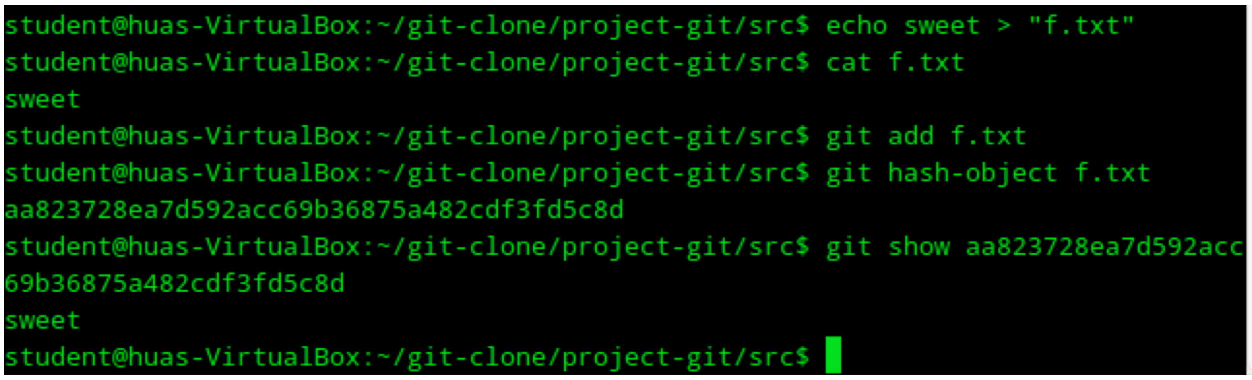
Yes, it’s a new commit.



```
student@huas-VirtualBox:~/project-git/src$ git commit -m "conflict resolved"
[master 347bdc2] conflict resolved
student@huas-VirtualBox:~/project-git/src$
```

1.6

1. What do you see?



```
student@huas-VirtualBox:~/git-clone/project-git/src$ echo sweet > "f.txt"
student@huas-VirtualBox:~/git-clone/project-git/src$ cat f.txt
sweet
student@huas-VirtualBox:~/git-clone/project-git/src$ git add f.txt
student@huas-VirtualBox:~/git-clone/project-git/src$ git hash-object f.txt
aa823728ea7d592acc69b36875a482cdf3fd5c8d
student@huas-VirtualBox:~/git-clone/project-git/src$ git show aa823728ea7d592acc
69b36875a482cdf3fd5c8d
sweet
student@huas-VirtualBox:~/git-clone/project-git/src$ 
```