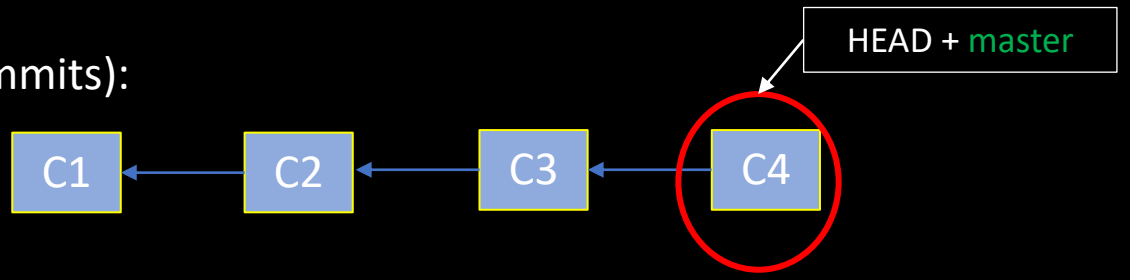
Frühere Versionen untersuchen

History (Commits):



versions.txt

```
1  first line c1
2  second line c2
3  third line c3
4  fourth line c4
5
```

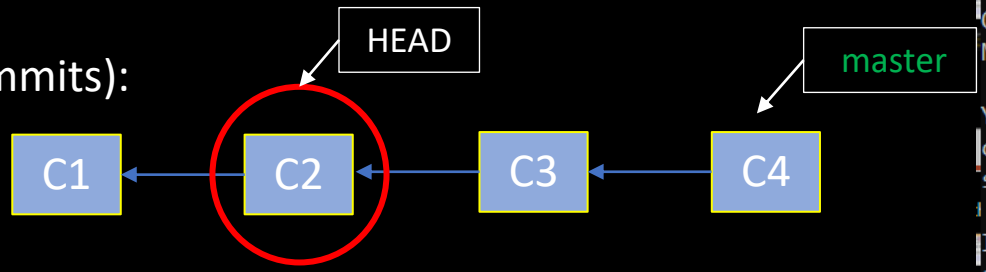
```
C:\Users\...\source\cprsw\versions>git log --pretty=oneline
663a6be27c084e1840a8da39dc56a589bd8ad63c (HEAD -> master) C4
ae6720f22fb857e2ecec13c67ea54bec687866f8 C3
887c61260981824dee6ac32739f52d4433ae314f C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

Aufgabe:

1. Navigiere zu deinem CPRSW-Ordner
2. Erstelle eine Ordner „versions“
3. Erzeuge darin ein lokales Repo
4. Erstelle eine versions.txt-Datei
5. Füge je 4 Zeilen nacheinander hinzu
   - nach jeder Zeile ein add und commit mit message „C1“ – „C4“
6. Lasse dir die History ausgeben (git log)

Frühere Version untersuchen

History (Commits):




```
C:\Users\      \source\cprsw\versions>git checkout 887c6126098
Note: switching to '887c6126098'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 887c612 C2
```

Aufgabe:

1. Wechsle zu Commit „C2“ mittels:
   `git checkout` DEIN_HASHCODE_C2
   (Hashcode kopieren bzw. ersten Zeichen eingeben)
2. Überprüfe den Inhalt deiner versions.txt-Datei
3. Lasse dir die History ausgeben (git log)

versions.txt

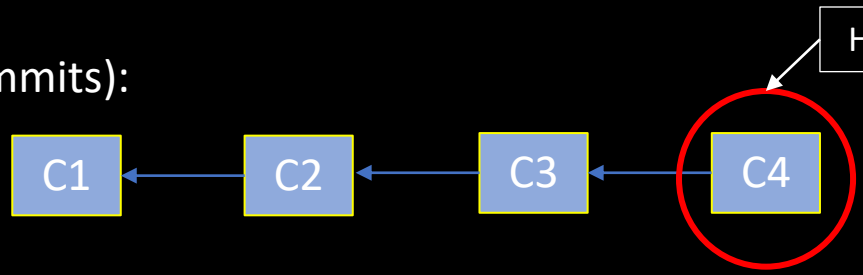
```
1  first line c1
2  second line c2
3
```

```
C:\Users\      ```source\cprsw\versions>git log --pretty=oneline
887c61260981824dee6ac32739f52d4433ae314f (HEAD) C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

Zur aktuellsten Version zurückwechseln

HEAD: Wie Zeiger, `checkout` dient dazu, die Markierung „HEAD“ zu verschieben

History (Commits):




```
C:\Users\…  …\source\cprsw\versions>git checkout master
Previous HEAD position was 887c612 C2
Switched to branch 'master'
```

Aufgabe:

1. Wechsle zurück zur aktuellsten Version „C4“ mittels:
   `git checkout` **`master`**
   (Hashcode nicht bekannt = nutze **master**)
2. Überprüfe den Inhalt deiner versions.txt-Datei
3. Lasse dir die History ausgeben (git log)

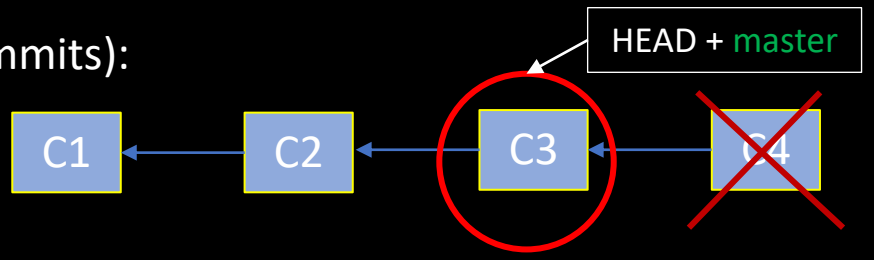
versions.txt

```
1  first line c1
2  second line c2
3  third line c3
4  fourth line c4
5
```

```
C:\Users\…  …\source\cprsw\versions>git log --pretty=oneline
663a6be27c084e1840a8da39dc56a589bd8ad63c (HEAD -> master) C4
ae6720f22fb857e2ecec13c67ea54bec687866f8 C3
887c61260981824dee6ac32739f52d4433ae314f C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

Frühere Version wiederherstellen

History (Commits):



versions.txt

```
1  first line c1
2  second line c2
3  third line c3
4
```

```
C:\Users\...  \source\cprsw\versions>git reset --hard ae6720f22
HEAD is now at ae6720f C3
```

Aufgabe:

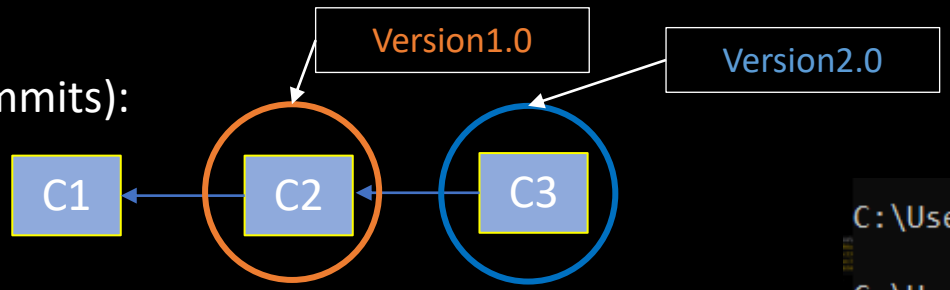
1. C3 wiederherstellen (nachfolgende Änderungen löschen) mittels:
   `git reset --hard` DEIN_HASHCODE_C3
   (Hashcode kopieren bzw. ersten Zeichen eingeben)
2. Lasse dir die History ausgeben (git log)

```
C:\Users\...\source\cprsw\versions>git log --pretty=oneline
ae6720f22fb857e2ecec13c67ea54bec687866f8 (HEAD -> master) C3
887c61260981824dee6ac32739f52d4433ae314f C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

„master“ (=aktuellste Version) ist nun C3

Versionierung mittels Tags

History (Commits):



Aufgabe:

1. Versioniere C3: C3 muss **aktuell ausgecheckt (HEAD)** sein:
   `git tag version2.0`
   (*KEINE LEERZEICHEN* beim Versionsnamen!)
2. Versioniere C2 (ohne dieses auszuchecken) mittels:
   `git tag version1.0` DEIN_HASHCODE_C2
3. Anzeige der Tags mittels „`git log`“
4. Checke Version 1.0 aus mittels:
   `git checkout version1.0`
5. Überprüfe den Inhalt deiner versions.txt-Datei

```
C:\Users\ \source\cprsw\versions>git tag version2.0

C:\Users\ \source\cprsw\versions>git tag version1.0 887c612
```

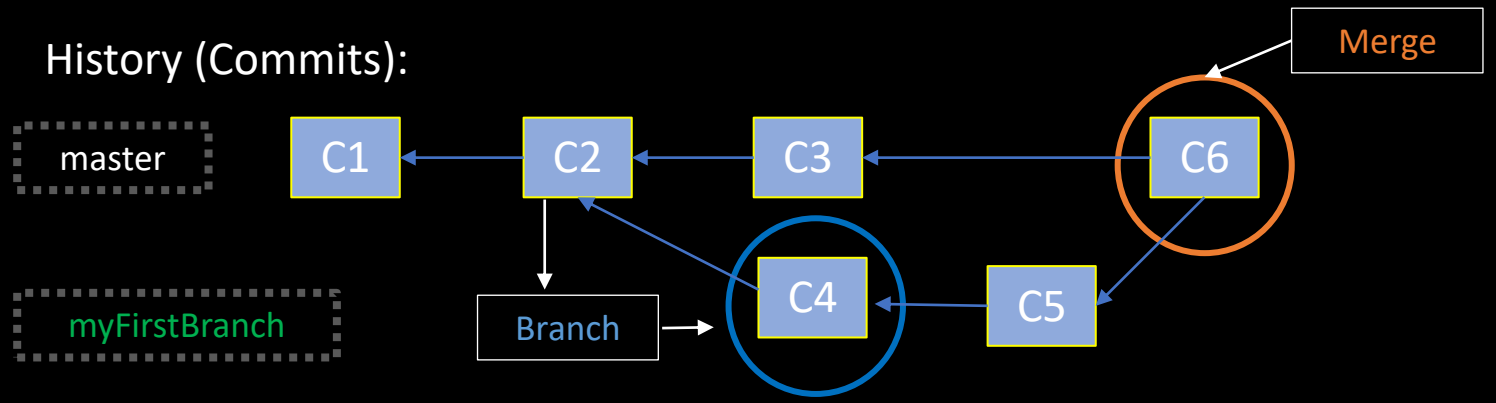
```
C:\Users\ \source\cprsw\versions>git log --pretty=oneline
ae6720f22fb857e2ecec13c67ea54bec687866f8 (HEAD -> master, tag: version2.0) C3
887c61260981824dee6ac32739f52d4433ae314f (tag: version1.0) C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

```
C:\Users\ \source\cprsw\versions>git log --pretty=oneline
887c61260981824dee6ac32739f52d4433ae314f (HEAD, tag: version1.0) C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1
```

versions.txt

```
1  first line c1
2  second line c2
3
```

Branches erstellen



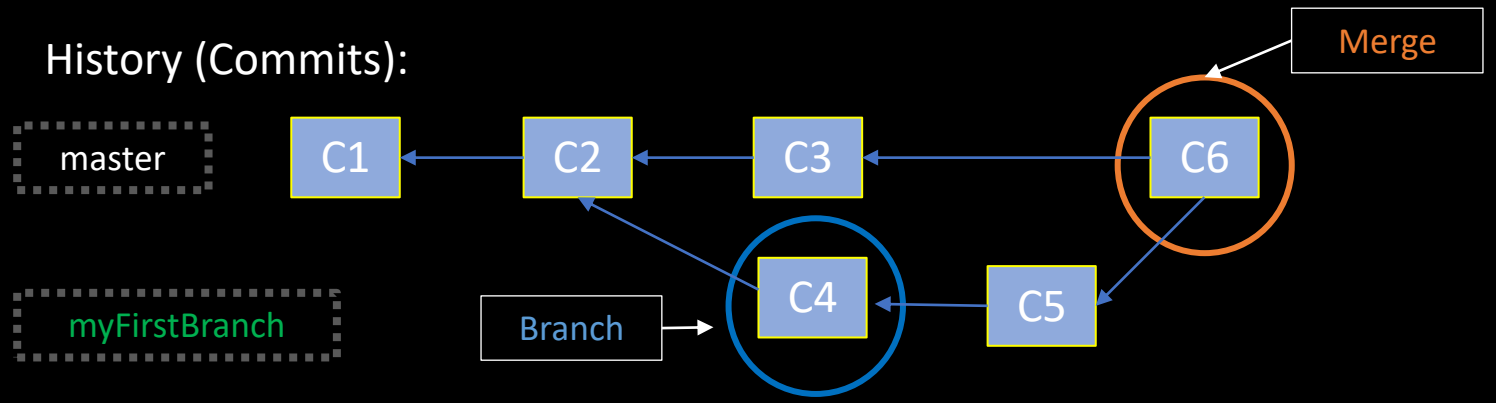
Aufgabe:

1. C2 muss **aktuell ausgecheckt (HEAD)** sein. Erstelle Branch:
   `git branch myFirstBranch`
2. Switche zu myFirstBranch:
   `git checkout myFirstBranch`
3. Anzeige mittels „`git log`“
4. Anzeige der Branches mittels:
   `git branch`

```
C:\Users\...\source\cprsw\versions>git log --pretty=oneline
887c61260981824dee6ac32739f52d4433ae314f (HEAD -> myFirstBranch, tag: version1.0) C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1

C:\Users\...\source\cprsw\versions>git branch
  master
* myFirstBranch
```

Branches erstellen

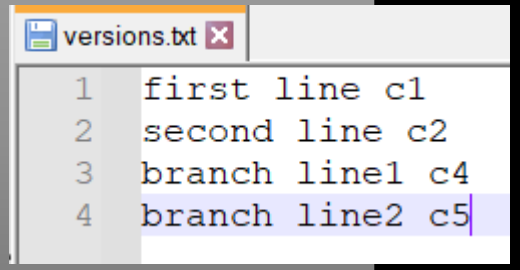
History (Commits):



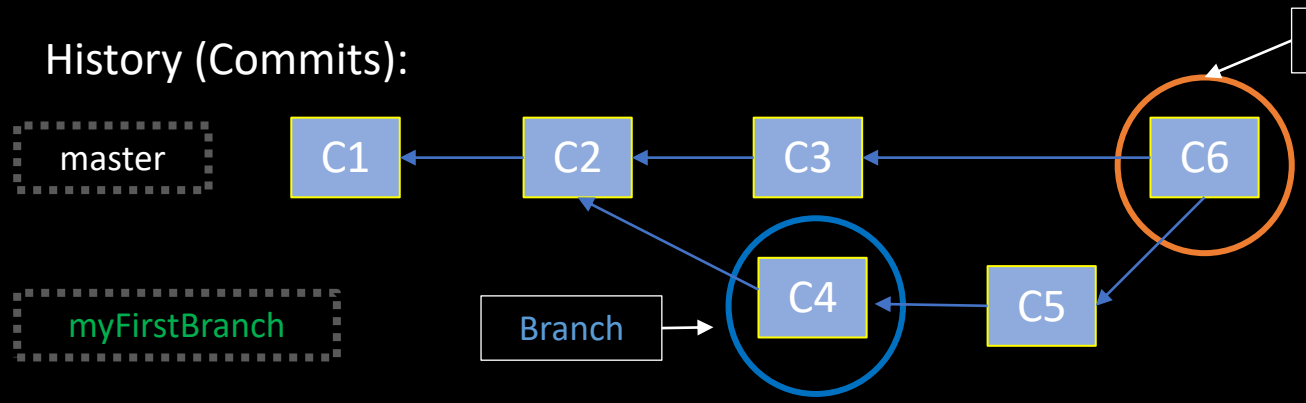

```
rsw\versions>git log --pretty=oneline
d8666c73ede0d1cd (HEAD -> myFirstBranch) C4
39f52d4433ae314f (tag: version1.0) C2
c364527c02cd5f1b C1

rsw\versions>git commit -a -m "C5"
C5
tions(+), 1 deletion(-)

rsw\versions>git log --pretty=oneline
720520a78d367ae2 (HEAD -> myFirstBranch) C5
d8666c73ede0d1cd C4
39f52d4433ae314f (tag: version1.0) C2
c364527c02cd5f1b C1
```

Aufgabe:

1. Ändern der versions.txt:
   „branch line1 c4“ + adden/committen
2. Ändern der versions.txt:
   „branch line2 c5“ + adden/committen
3. Anzeige mittels „`git log`“



Branches erstellen/Merge



Aufgabe:

1. Switche zurück zu **master**:
   `git checkout master`
2. **Merge** mittels:
   `git merge myFirstBranch`
3. Anzeige mittels „`git log`“
4. **Löse** den angezeigten **Konflikt manuell**

```
C:\Users\      \source\cprsw\versions>git checkout master
Switched to branch 'master'

C:\Users\      \source\cprsw\versions>git log --pretty=oneline
ae6720f22fb857e2ecec13c67ea54bec687866f8 (HEAD -> master, tag: version2.0) C3
887c61260981824dee6ac32739f52d4433ae314f (tag: version1.0) C2
2ccfcb1916365828d39b5446c364527c02cd5f1b C1

C:\Users\      \source\cprsw\versions>git merge myFirstBranch
Auto-merging versions.txt
CONFLICT (content): Merge conflict in versions.txt
Automatic merge failed; fix conflicts and then commit the result.
```

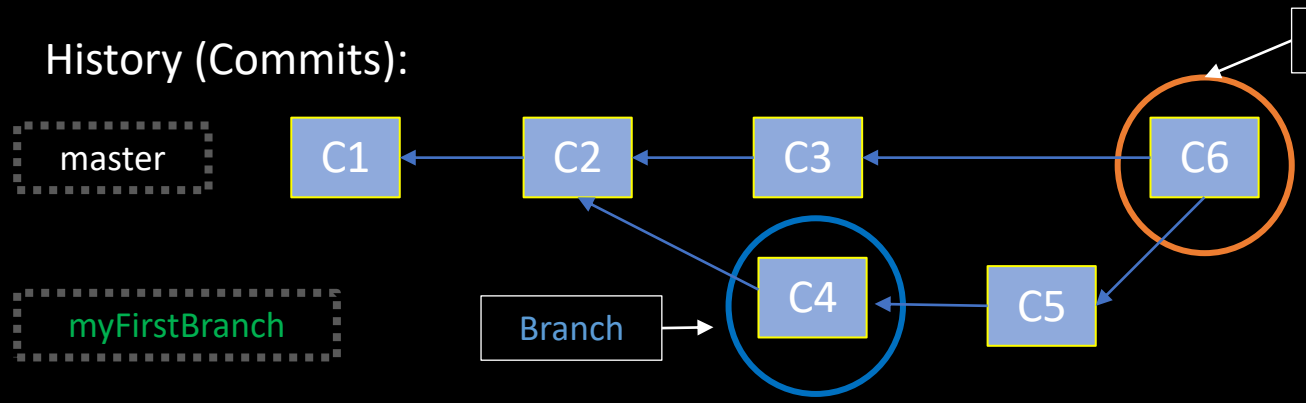
versions.txt

```
1  first line c1
2  second line c2
3  <<<<<<< HEAD
4  third line c3
5  =======
6  branch line1 c4
7  branch line2 c5
8  >>>>>>> myFirstBranch
9
```

versions.txt

```
1  first line c1
2  second line c2
3  third line c3
4  branch line1 c4
5  branch line2 c5
6
```

Branches erstellen/Merge



Aufgabe:

1. Speichern des gelösten Konflikts in version.txt
2. Add/Commit
3. Anzeige mittels „`git log`“

```
\source\cprsw\versions>git log --pretty=oneline
f967494008308 58be92cd23d3d (HEAD -> master) C6
f2b09399287205 20a78d367ae2 (myFirstBranch) C5
1b0419e542d8666c73ede0d1cd C4
e2ecec13c67ea54bec687866f8 (tag: version2.0) C3
4dee6ac32739f52d4433ae314f (tag: version1.0) C2
28d39b5446c364527c02cd5f1b C1
```

versions.txt

```
first line c1
second line c2
third line c3
branch line1 c4
branch line2 c5
```

Lösche nicht mehr benötigten Branch abschließend:

`git branch -d myFirstBranch`