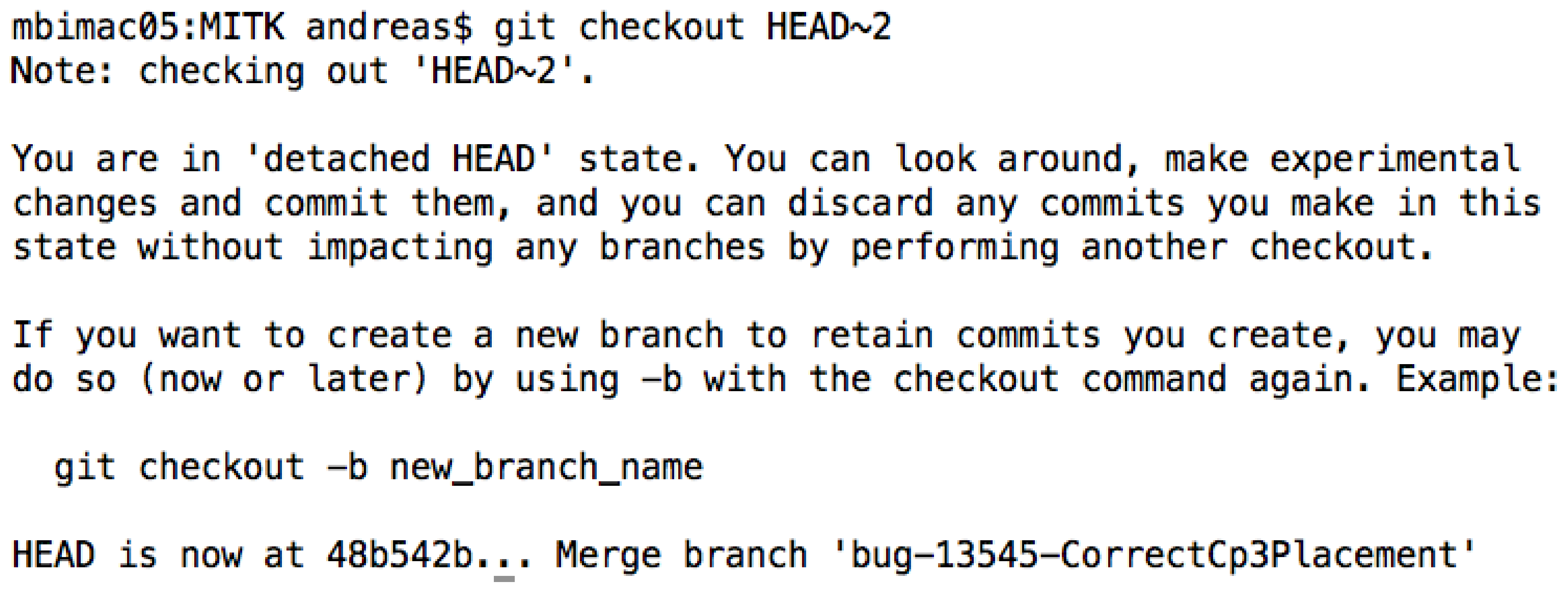
Git detached head

Andreas Fetzer
14.11.2012 MBI

Background



```
mbimac05:MITK andreas$ git checkout HEAD~2
Note: checking out 'HEAD~2'.

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 performing another checkout.

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

  git checkout -b new_branch_name

HEAD is now at 48b542b... Merge branch 'bug-13545-CorrectCp3Placement'
```

What happened?

- HEAD usually is a symbolic reference to (the tip of) a branch.

  e.g. `cat .git/HEAD` prints

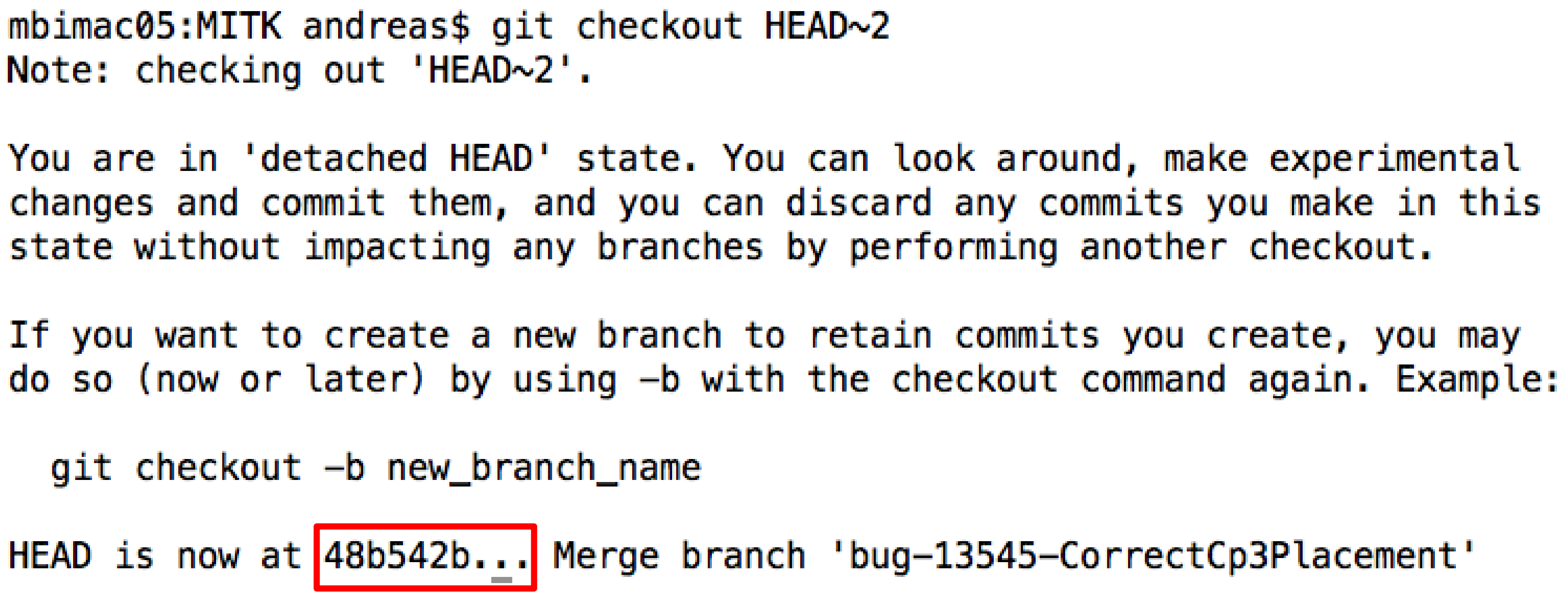
  ```
  refs: refs/heads/master
  ```

  Or:

  ```
  refs: refs/heads/myBranch
  ```

What happened

- However if you checkout anything that is not a proper local branch name HEAD simply contains the SHA-1 hash of the commit you are pointing to:



```
mbimac05:MITK andreas$ git checkout HEAD~2
Note: checking out 'HEAD~2'.

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 performing another checkout.

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

  git checkout -b new_branch_name

HEAD is now at 48b542b... Merge branch 'bug-13545-CorrectCp3Placement'
```

Command which detached you HEAD

```
git checkout master^        # parent of master
git checkout HEAD~2         # grandparent of current HEAD
git checkout origin/master # a non-local branch
git checkout tagname        # since you cant commit to a tag
```

Re-attaching the HEAD

- Even if you made a few commits before you realized you lost your HEAD:

```
git checkout -b newbranch
```

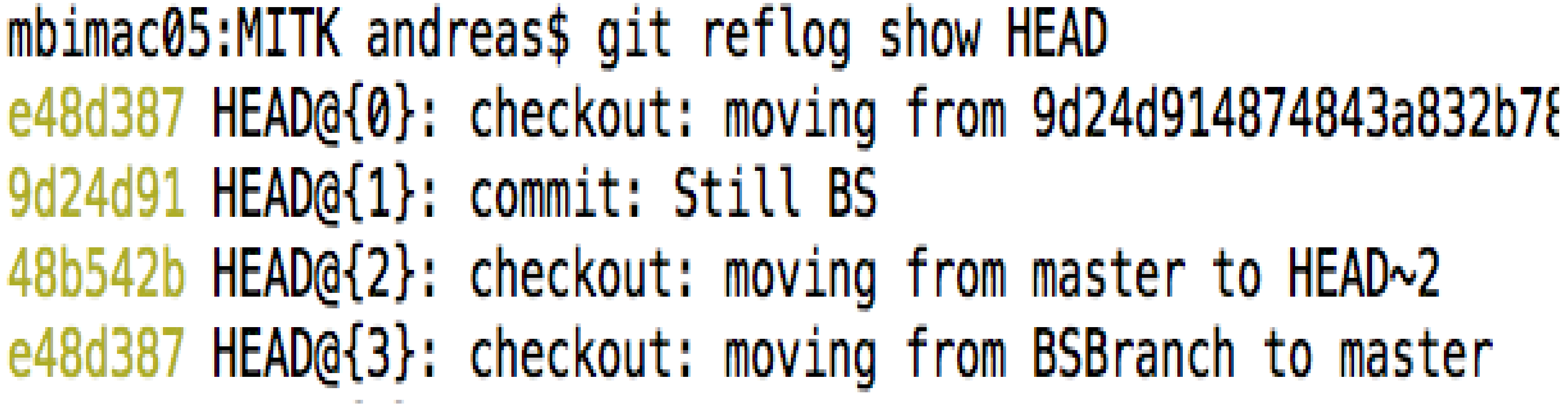
- Now HEAD is again pointing to a branch which is based upon the commit your HEAD was pointing to and everything is fine

Re-attaching the HEAD

- But what if you completely didn't realize you had no HEAD, made some (livesafing) commits and then switched to a existing branch...?
- The SHA-1 hash of your previous HEAD is overwritten since HEAD points now to the existing branch
- Is you lifesafing commit lost forever?

Git reflog

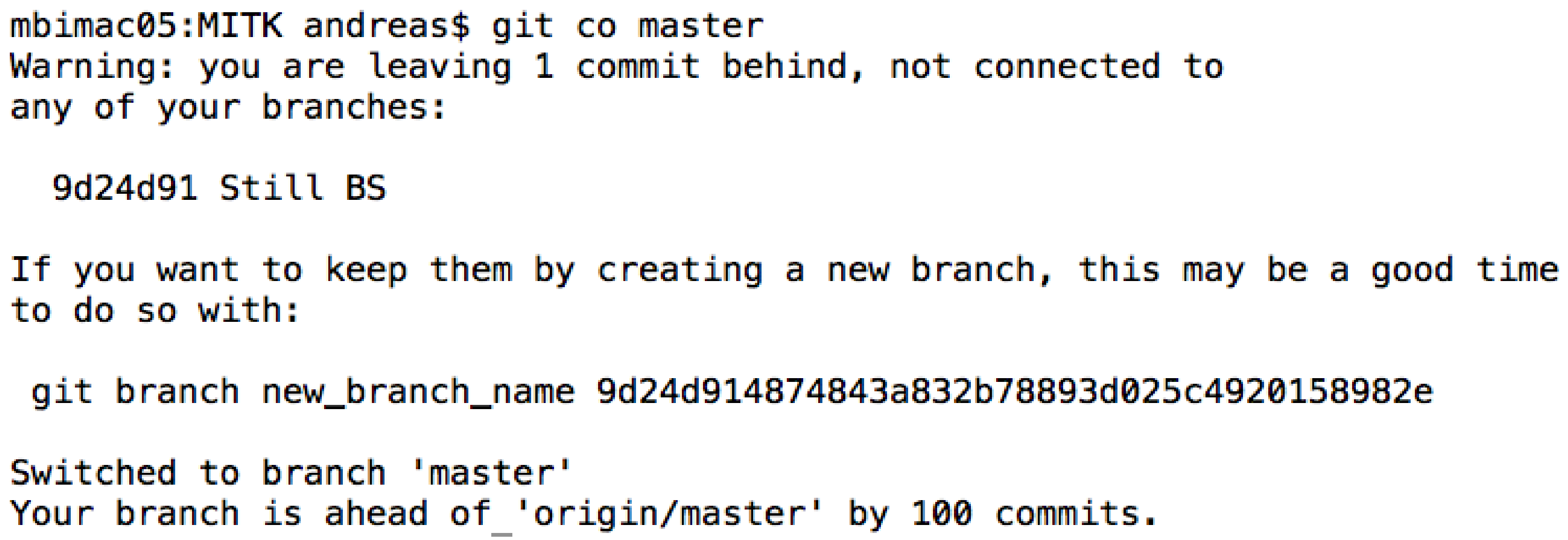
- Stores the changes which have been done, which command invoked those changes and when the changes did happen...**up to 30 days!!**



```
mbimac05:MITK andreas$ git reflog show HEAD
e48d387 HEAD@{0}: checkout: moving from 9d24d914874843a832b7
9d24d91 HEAD@{1}: commit: Still BS
48b542b HEAD@{2}: checkout: moving from master to HEAD~2
e48d387 HEAD@{3}: checkout: moving from BSBranch to master
```

Git reflog

- You can easily retrieve the „lost“ commit by simply creating a new branch basing on it.
- Because Git is awesome it gives your the correct suggestion:



```
mbimac05:MITK andreas$ git co master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  9d24d91 Still BS

If you want to keep them by creating a new branch, this may be a good time
to do so with:

 git branch new_branch_name 9d24d914874843a832b78893d025c4920158982e

Switched to branch 'master'
Your branch is ahead of_'origin/master' by 100 commits.
```

That's it!

- Very good explanation at:

  http://sitaramc.github.com/concepts/detached-head.html