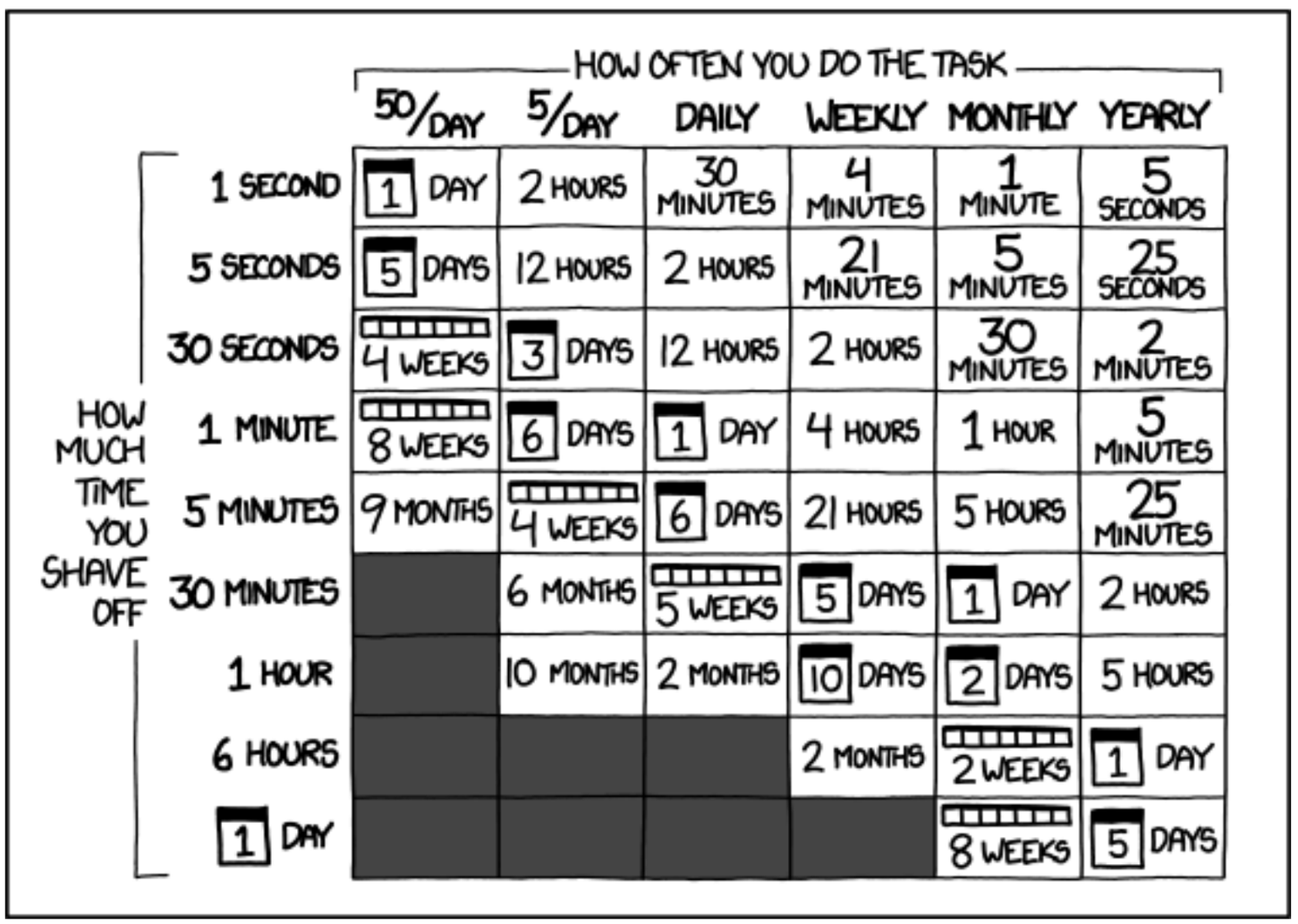
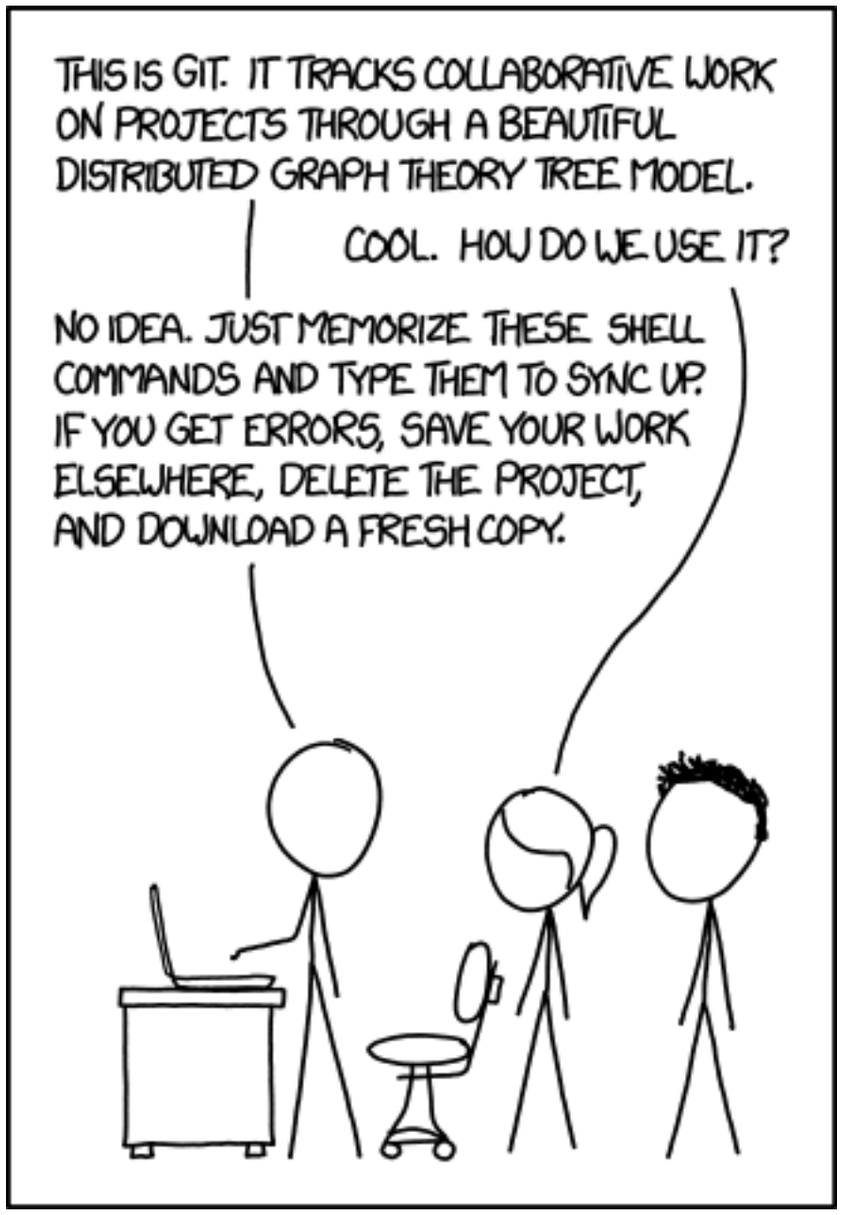




An introduction to git

Benedikt Daurer

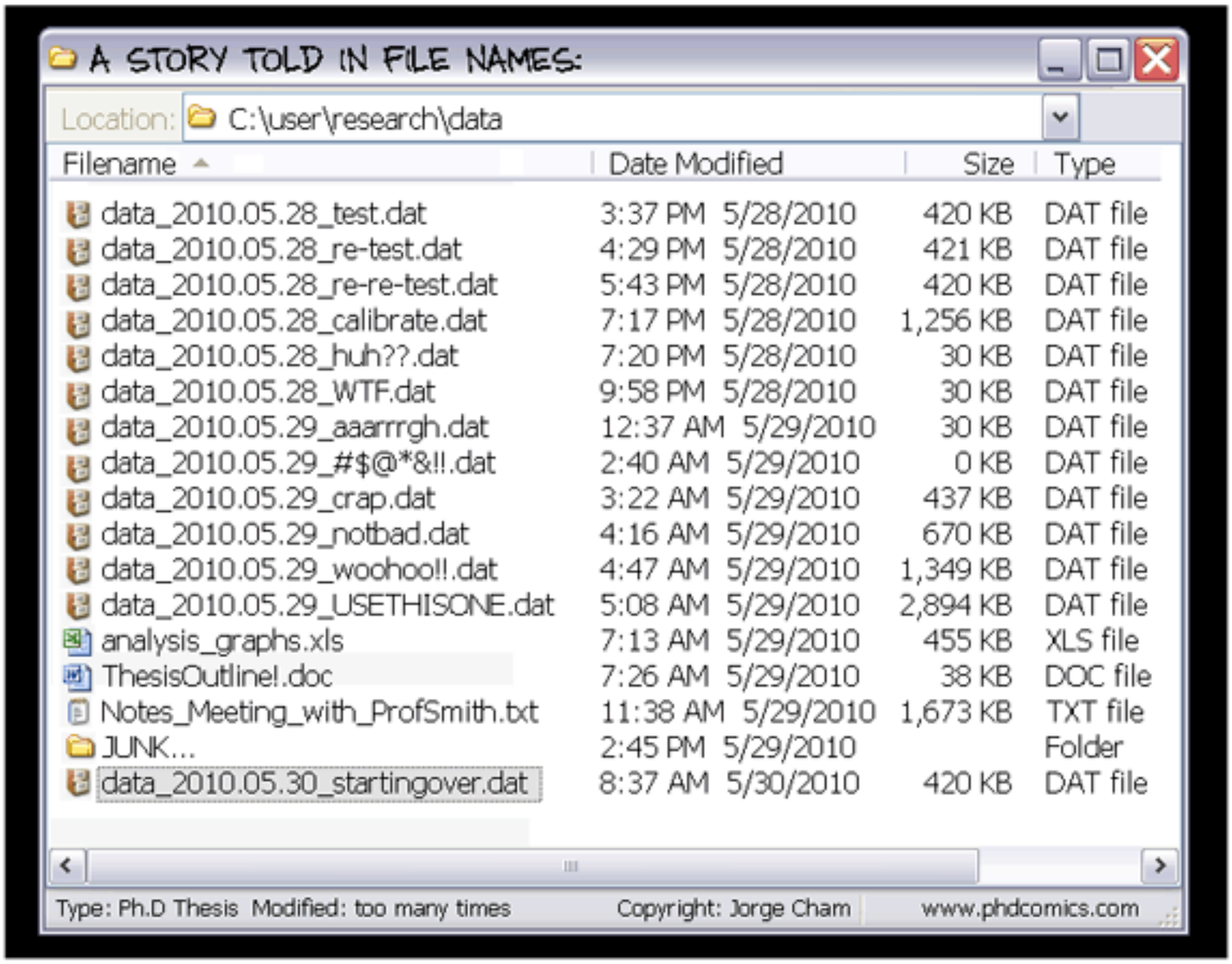
UPPSALA UNIVERSITET

Overview

- When should I use git and why?
- The basic concept behind version control
- How does it work in practice
- Basic git commands
- Hands-on: Getting familiar with the basics
- More advanced concepts
- Hands-on: Contribute to a collaborative project
- Summary, take-home messages

https://github.com/uu-python/lecture-git

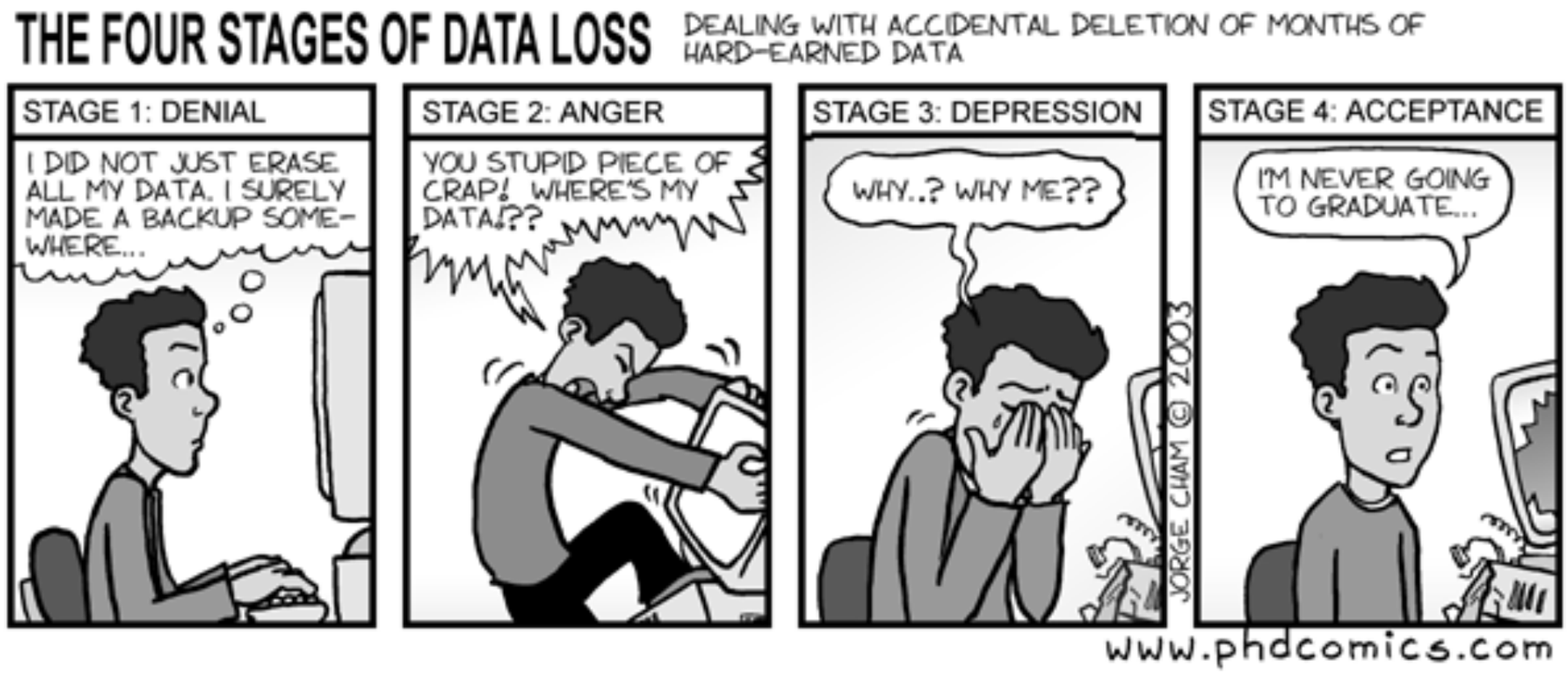
Why do I need version control?



by Jorge Cham
www.phdcomics.com

Why do I need version control?

- Your files are better organized
- You keep a history of all previous versions
- Your research is faster, more efficient and more reproducible
- Version control benefits collaborative work
- You always have a backup



How do I use git?

Creating a new project

```
$ git init
```

Cloning an existing project

```
$ git clone https://github.com/…/project.git
```

Adding new files to be commited

```
$ git add README.md
```

Commit all new files

```
$ git commit –m “Useful message"
```

Updating the local copy (“PULLING”)

```
$ git pull
```

Updating the remote (“PUSHING”)

```
$ git push
```

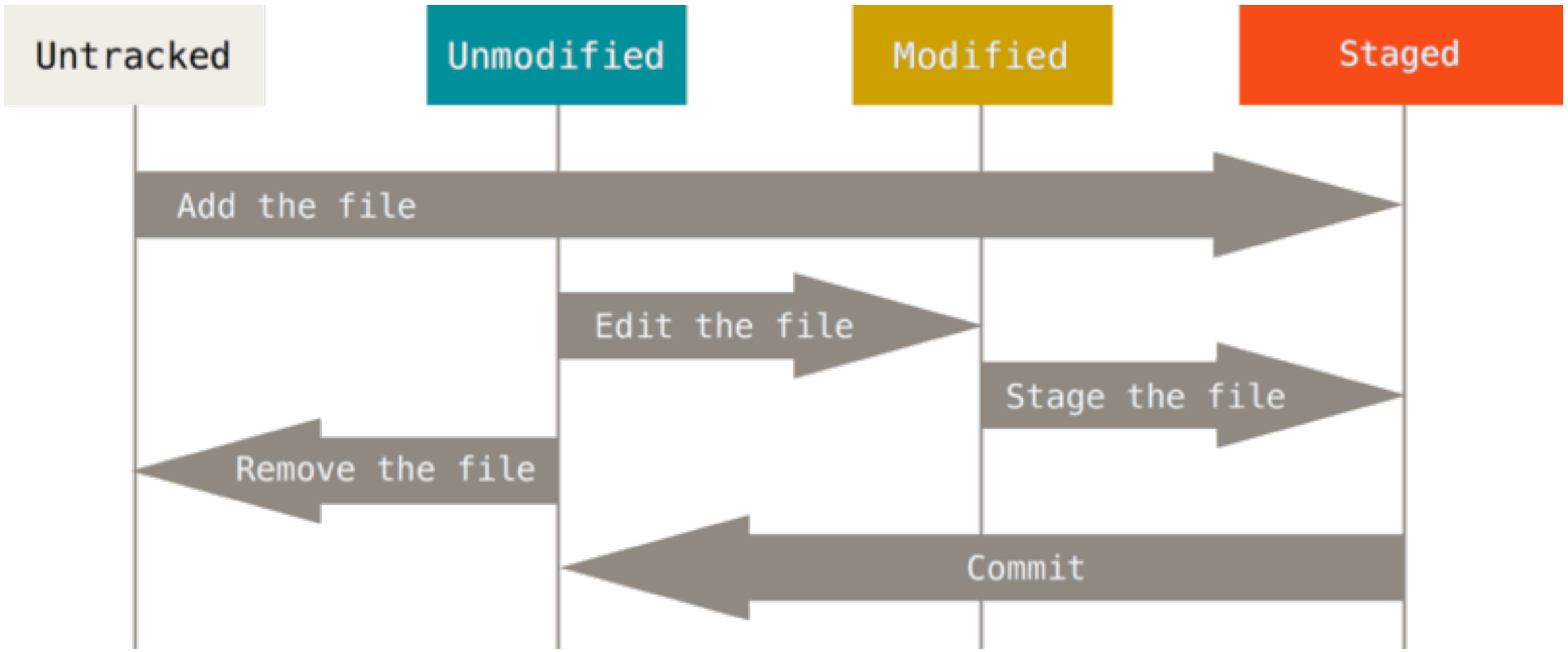
Current status of all files of repository

```
$ git status
```

Show the history (commit log)

```
$ git log
```

The lifecycle of the status of your files



Pro Git Boot, by Scott Chacon: http://git-scm.com/book

Setting up git

Local configurations (only the current repository is affected)

```
$ git config [options]
```

Global configurations (only the user's configuration is modified)

```
$ git config  --global [options]
```

System configurations (all users are affected)

```
$ git config  --system [options]
```

Change your identity

```
$ git config --global user.name "Benedikt J. Daurer"
$ git config --global user.email "benedikt.daurer@icm.uu.se"
```

Set your favourite editor (e.g. emacs or vim)

```
$ git config --global core.editor emacs
```

Check your current settings

```
$ git config --list
```

Demonstration: The basic git commands

Exercise 1: Getting familiar with basic commands

- 1. Create a local copy (clone) of the following project: https://github.com/uu-python/particpants
- 2. Create a new file YOURNAME.md
- 3. Write something about yourself and add your file to the files tracked by git
- 4. Commit your changes and give a meaningful log message
- 5. Update your local repository by pulling from the remote
- 6. Update the remote repository by pushing your local changes

Creating a new project

```
$ git init
```

Cloning an existing project

```
$ git clone https://github.com/…/project.git
```

Adding new files to be commited

```
$ git add README.md
```

Commit all new files

```
$ git commit –m “Useful message"
```

Updating the local copy (“PULLING”)

```
$ git pull
```

Updating the remote (“PUSHING”)

```
$ git push
```

Current status of all files of repository

```
$ git status
```

Show the history (commit log)

```
$ git log
```

Deleting, moving, cancelling, resetting

Deleting a tracked file

```
$ git rm FILE
```

Deleting a tracked file (but keeping an untracked copy)

```
$ git rm --cached FILE
```

Moving a file (renaming)

```
$ git mv FILE TARGET
```

Unstaging a file

```
$ git reset HEAD FILE
```

Unmodify unstaged files

```
$ git checkout -- FILE1 FILE2
```

Checkout a previous version

```
$ git checkout HASH
```

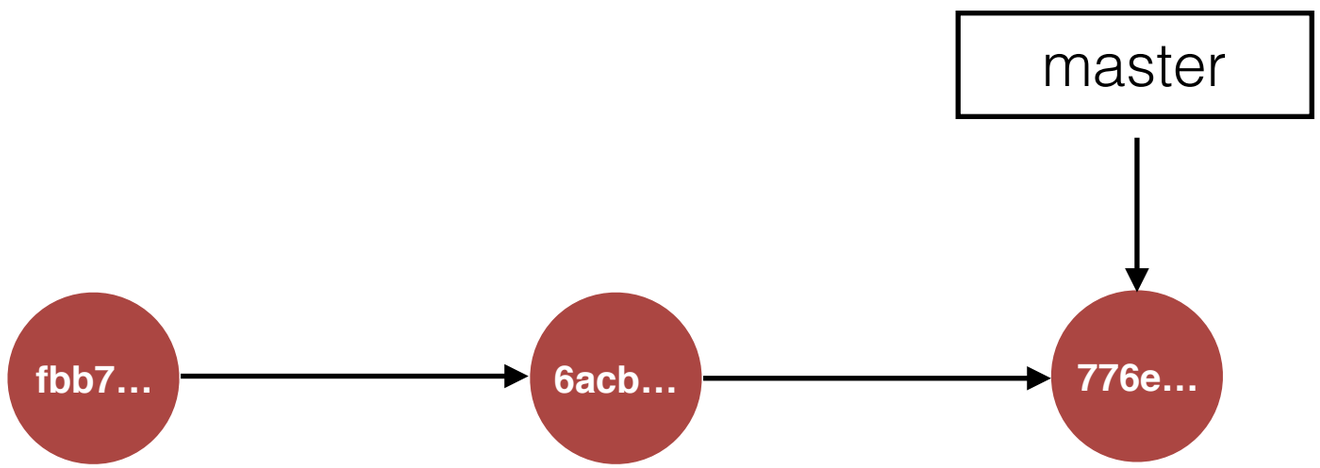
Branching and merging

Check the history of the branch (git log)

```
benedikt@icm-241-135:~/particpants$ git log

commit 776e7c4f19493d88a85832dcef44ff2e569586bb
Author: Benedikt Daurer <benedikt.daurer@icm.uu.se>
Date:   Mon Nov 21 15:59:00 2016 +0100

    Fixed typo

commit 6acb9e49d9de91a8aa1536a659f3003f477e9c7d
Author: Benedikt Daurer <benedikt.daurer@icm.uu.se>
Date:   Mon Nov 21 15:57:53 2016 +0100

    Fixed link

commit fbb7ef51497d8aa77304280419c7cc4c67511626
Author: Benedikt Daurer <benedikt.daurer@icm.uu.se>
Date:   Mon Nov 21 15:56:53 2016 +0100

    Initial commit
```

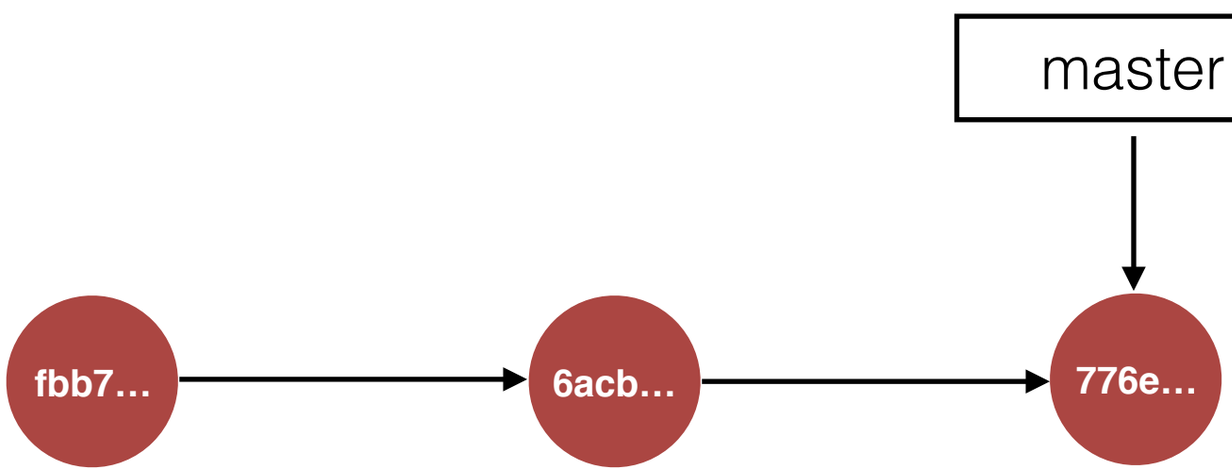
Check on the status of the branch (git status)

```
benedikt@icm-241-135:~/particpants$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```



Branching and merging

Branching and merging



Branching and merging

Create a new branch "testing"

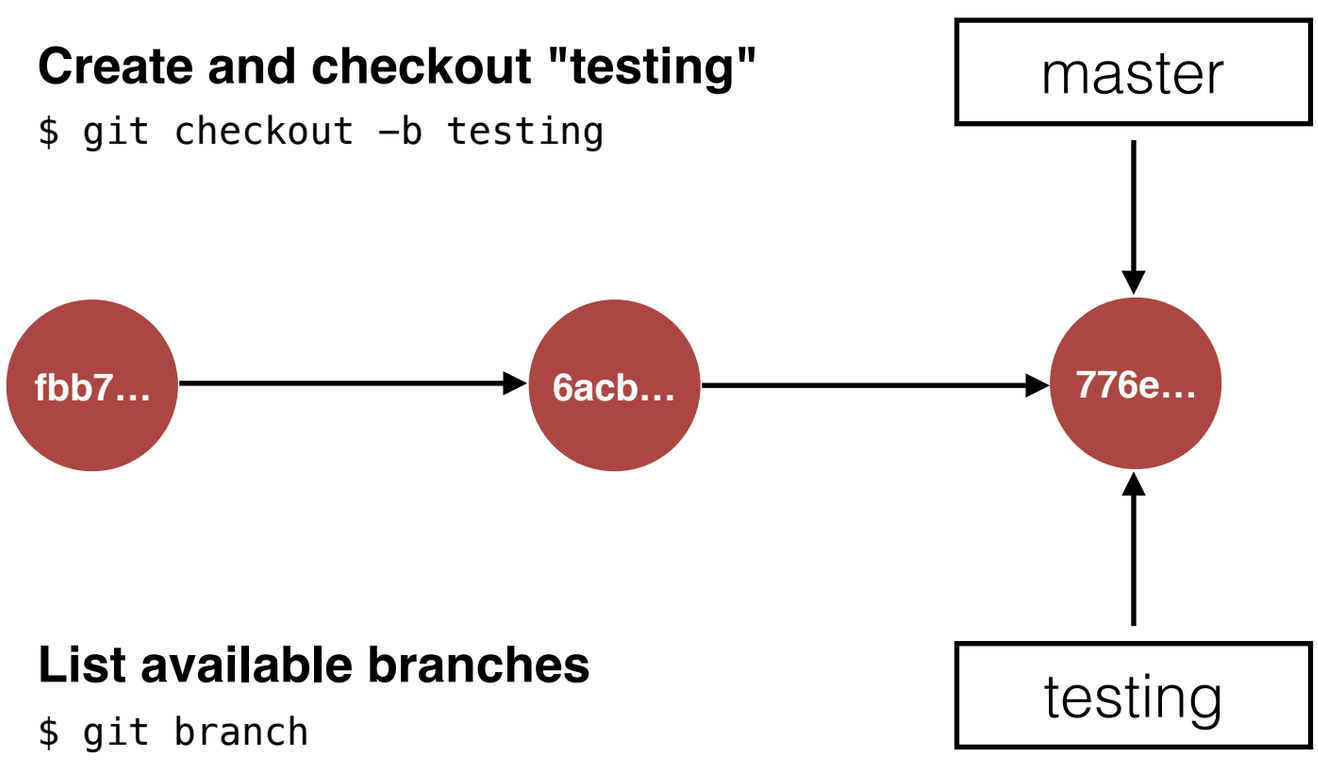
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Branching and merging

Create a new branch "testing"

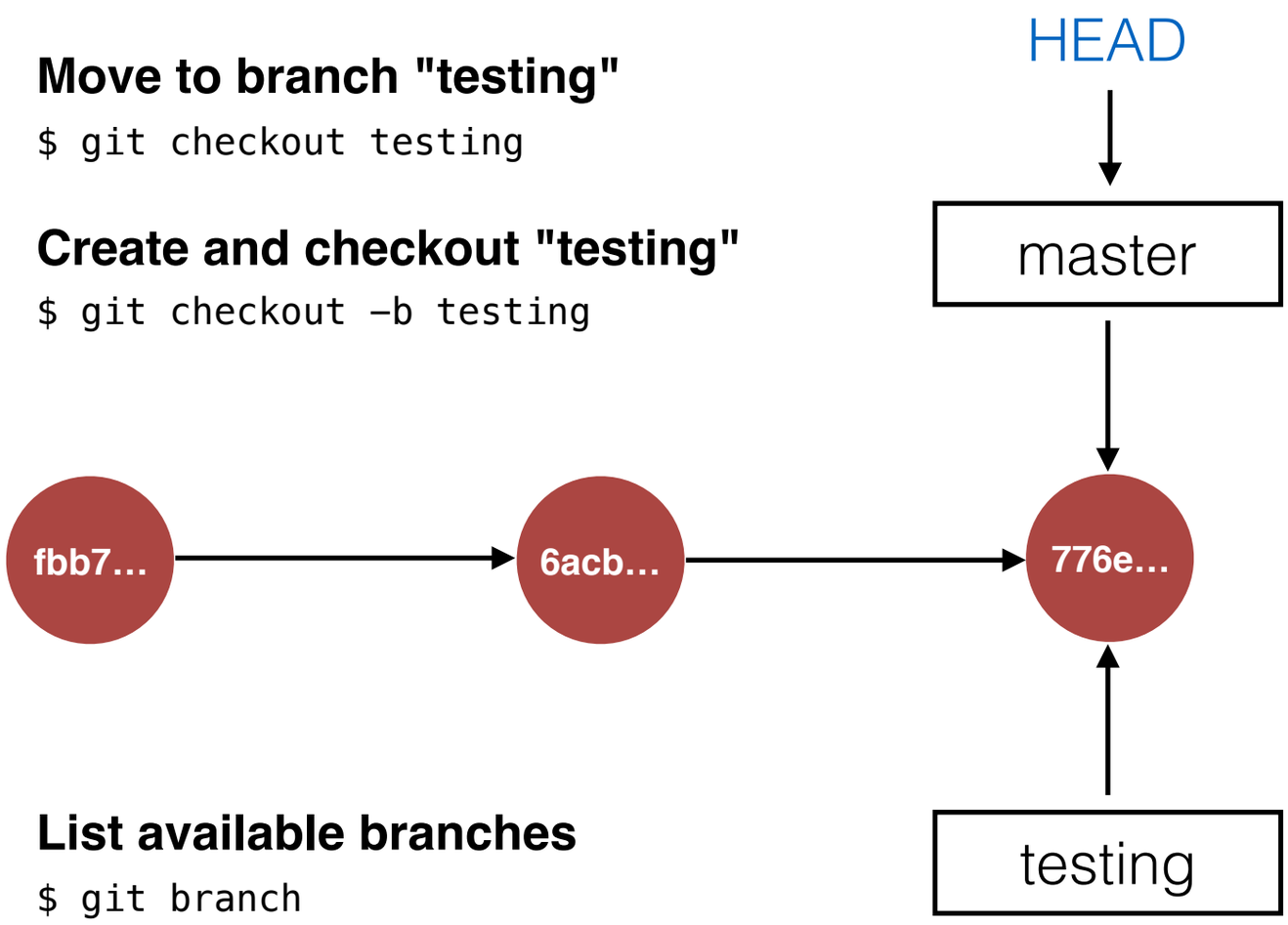
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Branching and merging

Create a new branch "testing"

```
$ git branch testing
```

Move to branch "testing"

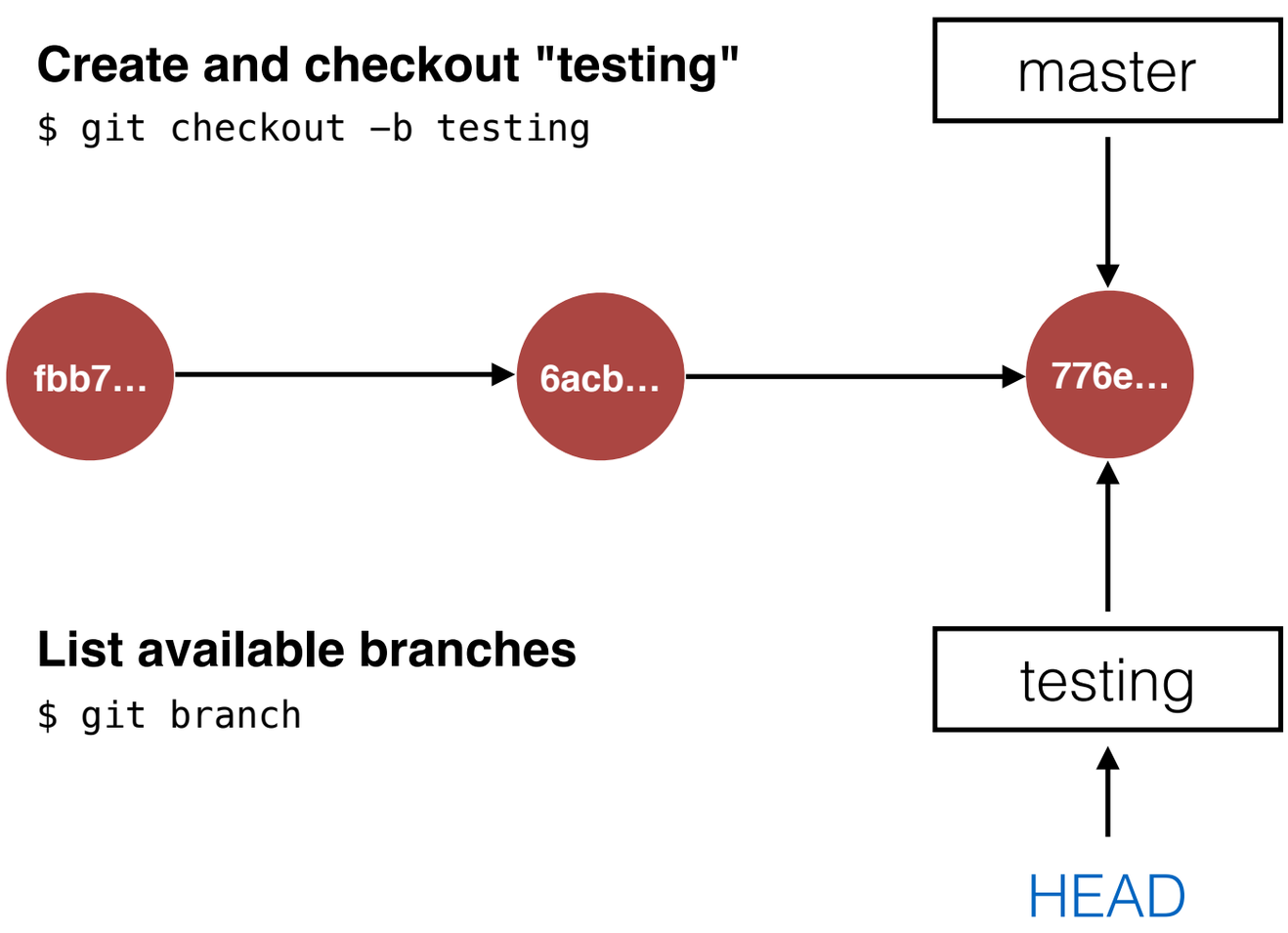
```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```

List available branches

```
$ git branch
```



Branching and merging

Create a new branch "testing"

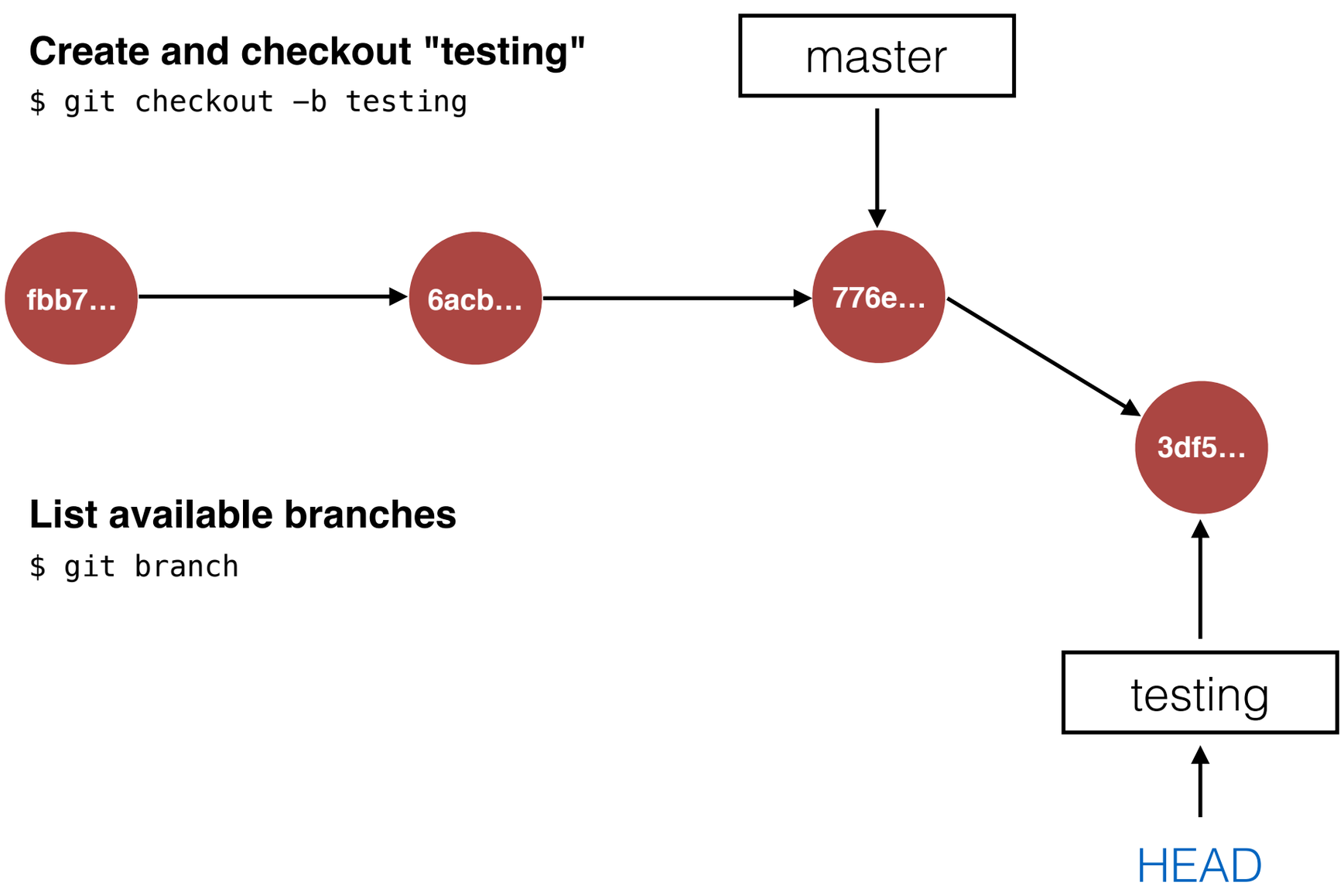
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Branching and merging

Create a new branch "testing"

```
$ git branch testing
```

Move to branch "testing"

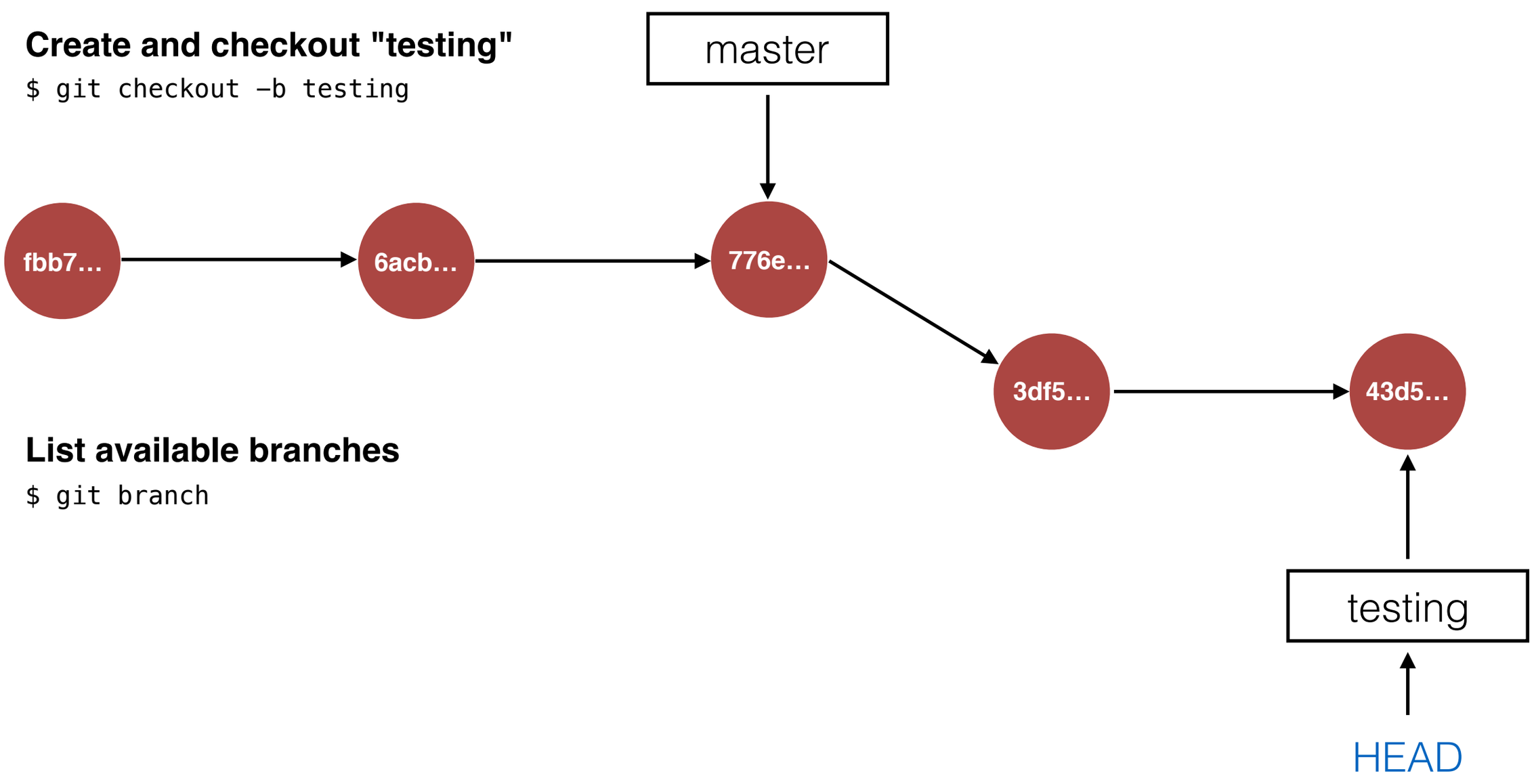
```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```

List available branches

```
$ git branch
```



Branching and merging

Create a new branch "testing"

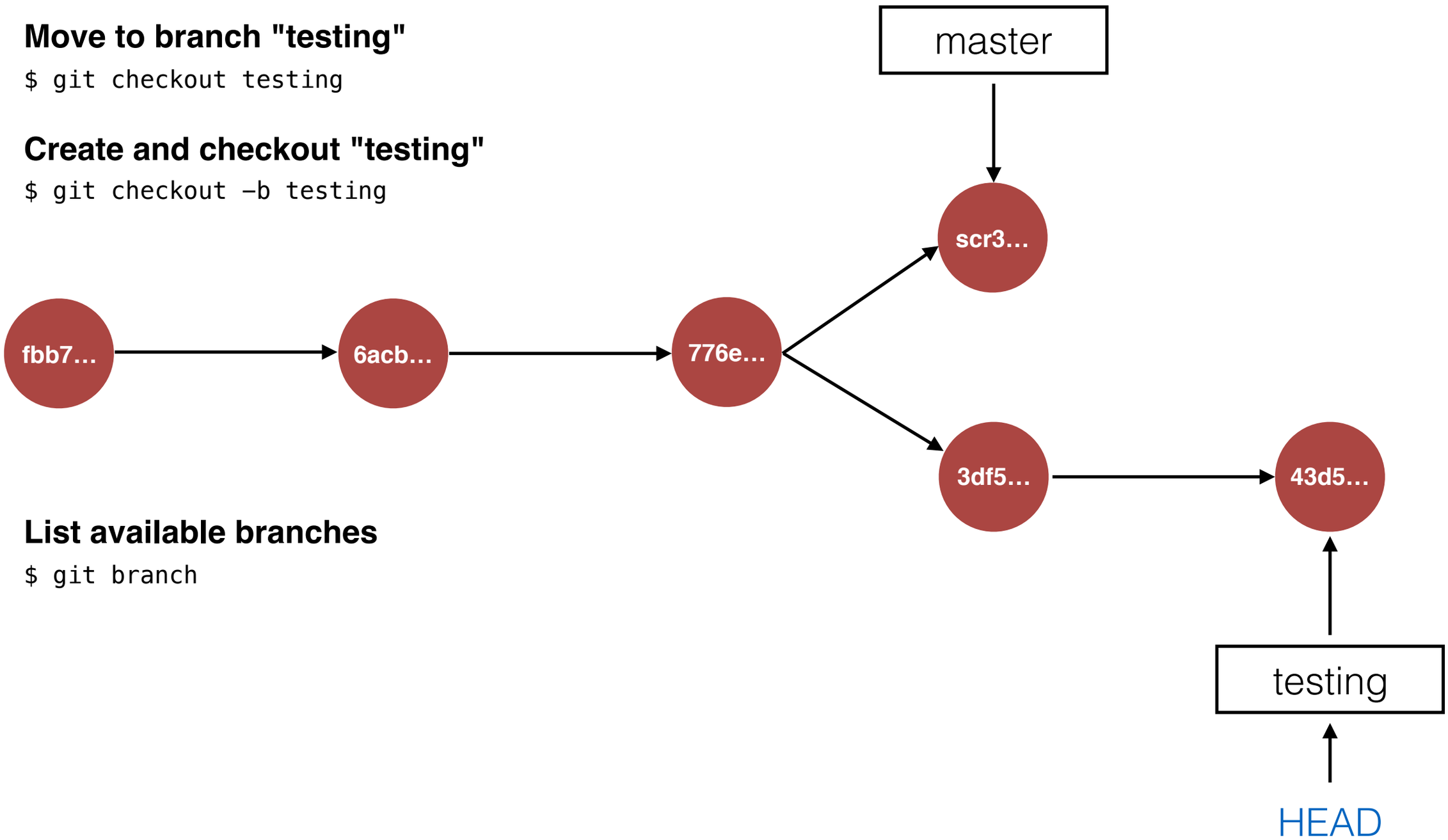
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Branching and merging

Create a new branch "testing"

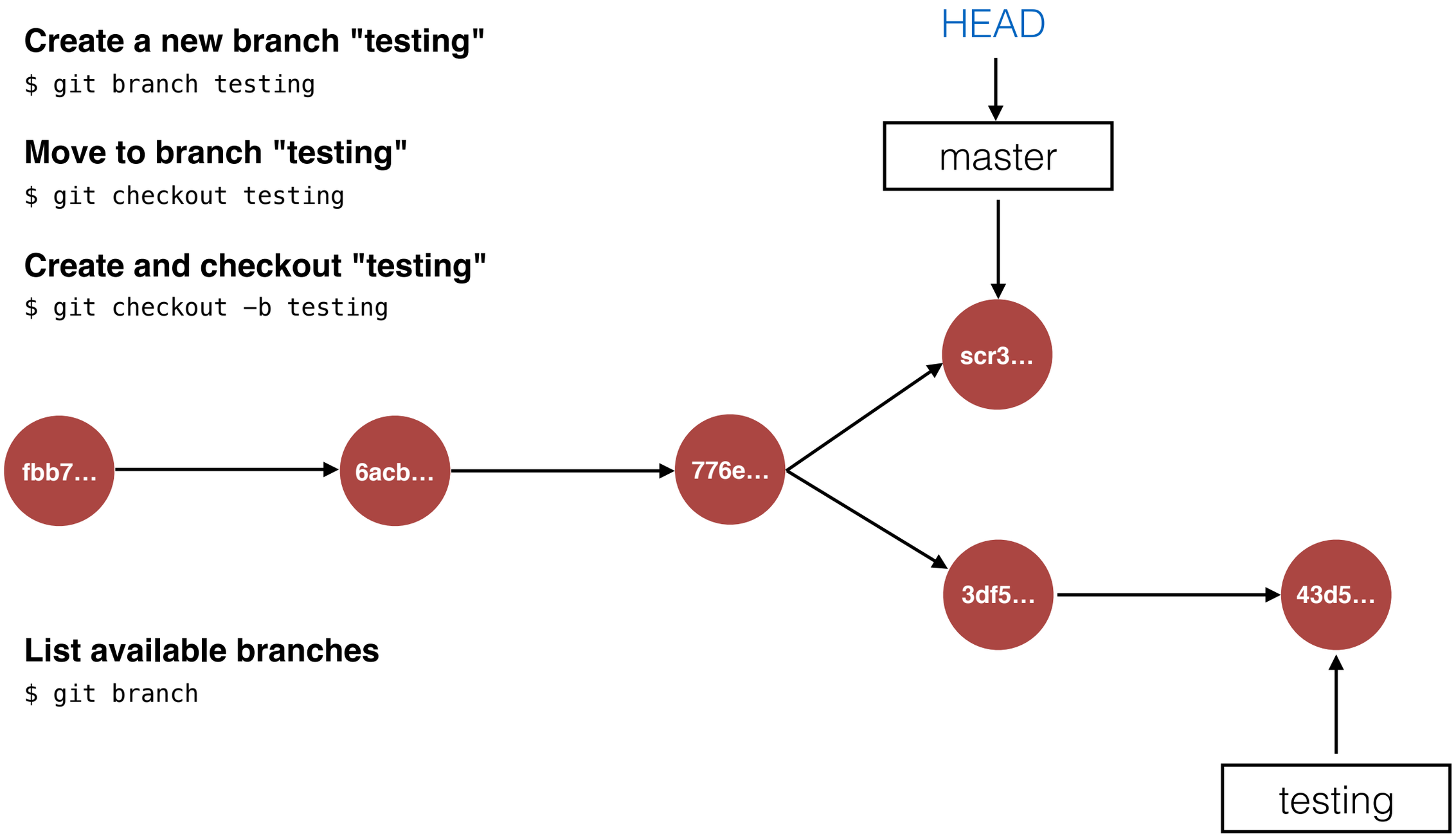
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Branching and merging

Create a new branch "testing"

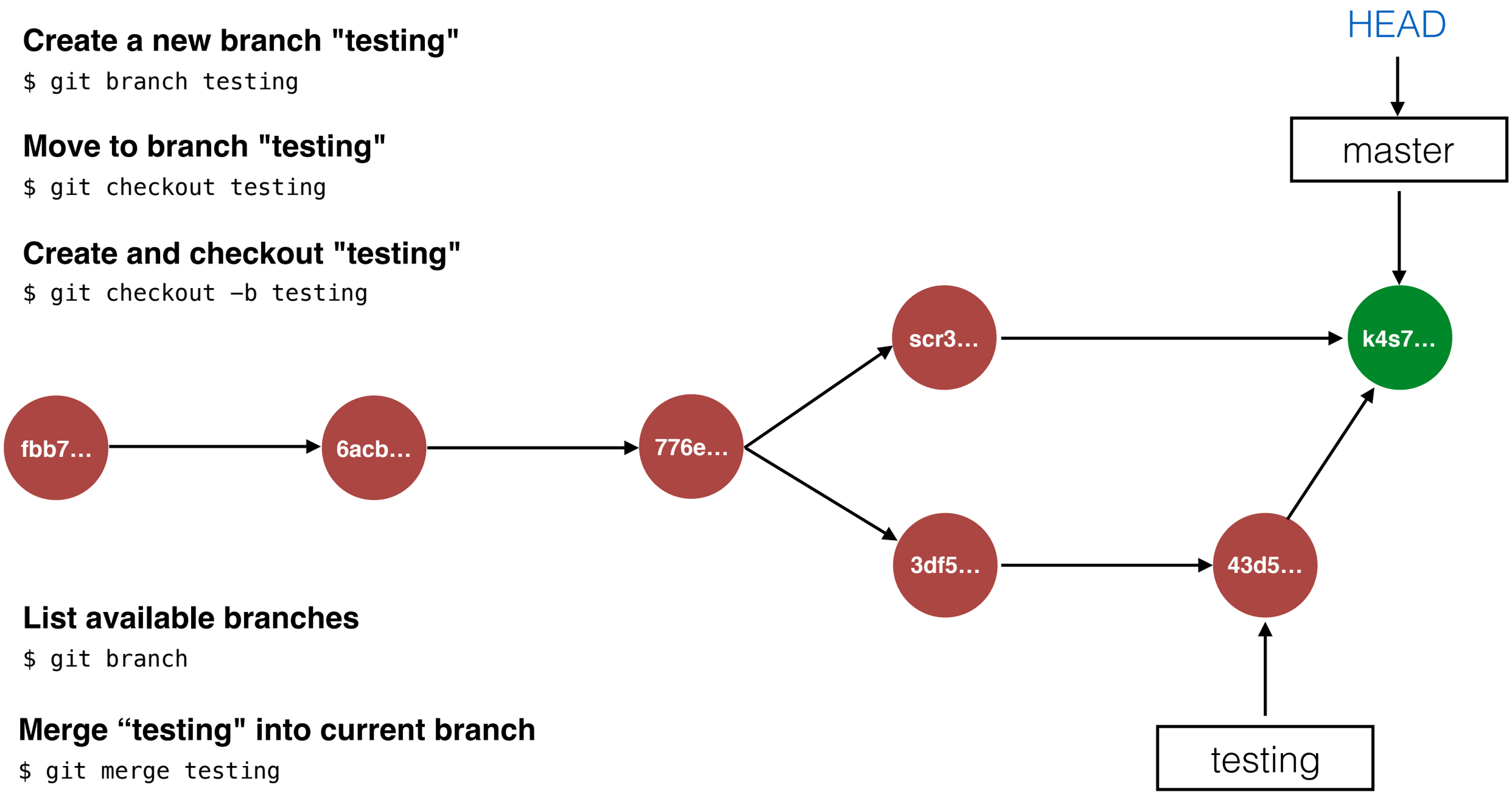
```
$ git branch testing
```

Move to branch "testing"

```
$ git checkout testing
```

Create and checkout "testing"

```
$ git checkout -b testing
```



List available branches

```
$ git branch
```

Merge “testing" into current branch

```
$ git merge testing
```

Demonstration: Branching and merging

Exercise 2: Contribute to a collaborative project

- 1. Create a local copy (clone) of the following project: https://github.com/uu-python/particpants
- 2. Create a new branch "yourname"
- 3. Edit a certain section of the file “cheatsheet.md” which was given to you by the teachers.
- 4. Commit your changes and give a meaningful log message
- 5. Push your local to remote branch with the same name
- 6. Switch to the master branch and merge the branch “yourname" into master
- 7. Update local master branch (pull)
- 8. Update remote master branch (push)

Adding new files to be commited

```
$ git add README.md
```

Commit all new files

```
$ git commit –m “Useful message"
```

Updating the local copy (“PULLING”)

```
$ git pull
```

Updating the remote (“PUSHING”)

```
$ git push
```

Create and checkout "testing"

```
$ git checkout –b testing
```

Push a local branch to remote "testing"

```
$ git push ––set–upstream origin testing
```

Switch to branch "testing"

```
$ git checkout testing
```

Take home messages

- Version control using git helps you better organize your work (e.g. code, documentation, manuscripts, thesis, webpages, …)
- git (together with github or bitbucket) enables collaborative coding/writing
- You always have a backup and you can easily go back to previous versions

Further reading and playing

- The Pro Git Book (https://git-scm.com/book/en/v2)
- Githug - a game for learning git (https://github.com/Gazler/githug)