Git : Part 2
Checkout, Add, Commit

These slides were largely cut-and-pasted from http://excess.org/article/2008/07/ogre-git-tutorial/ , with some additions from other sources. I have deleted a lot from the cited tutorial, and recommend that you listen to the entire tutorial on line, if you can.

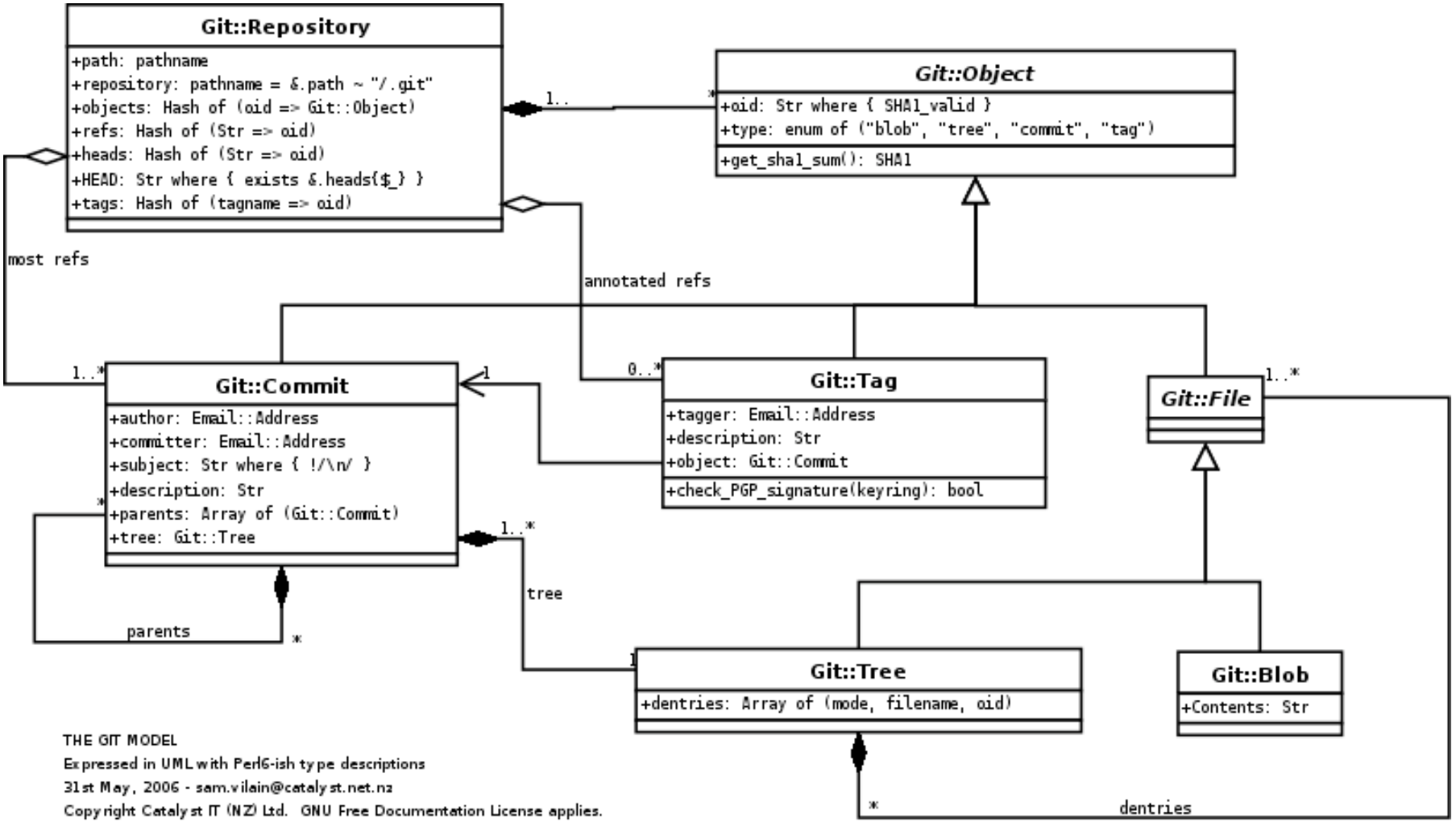
Topics

- Review
- Creating an empty repository
- Adding file changes to it
- Committing the changes
- Git naming conventions for commits
- Git commands for getting information about a repository

Review

Core git concepts

Git object model

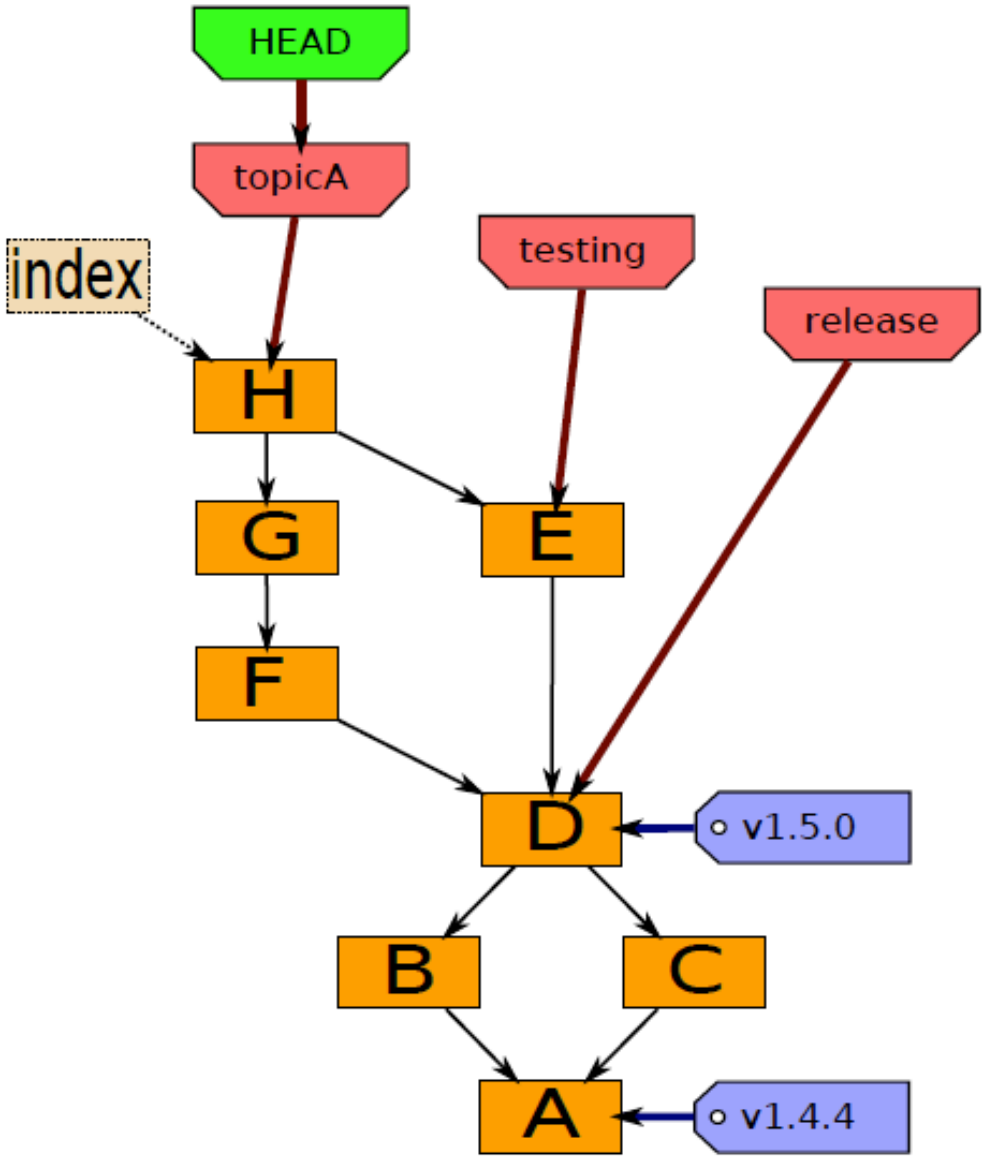


http://utsl.gen.nz/talks/git-svn/git-model.png

Git components

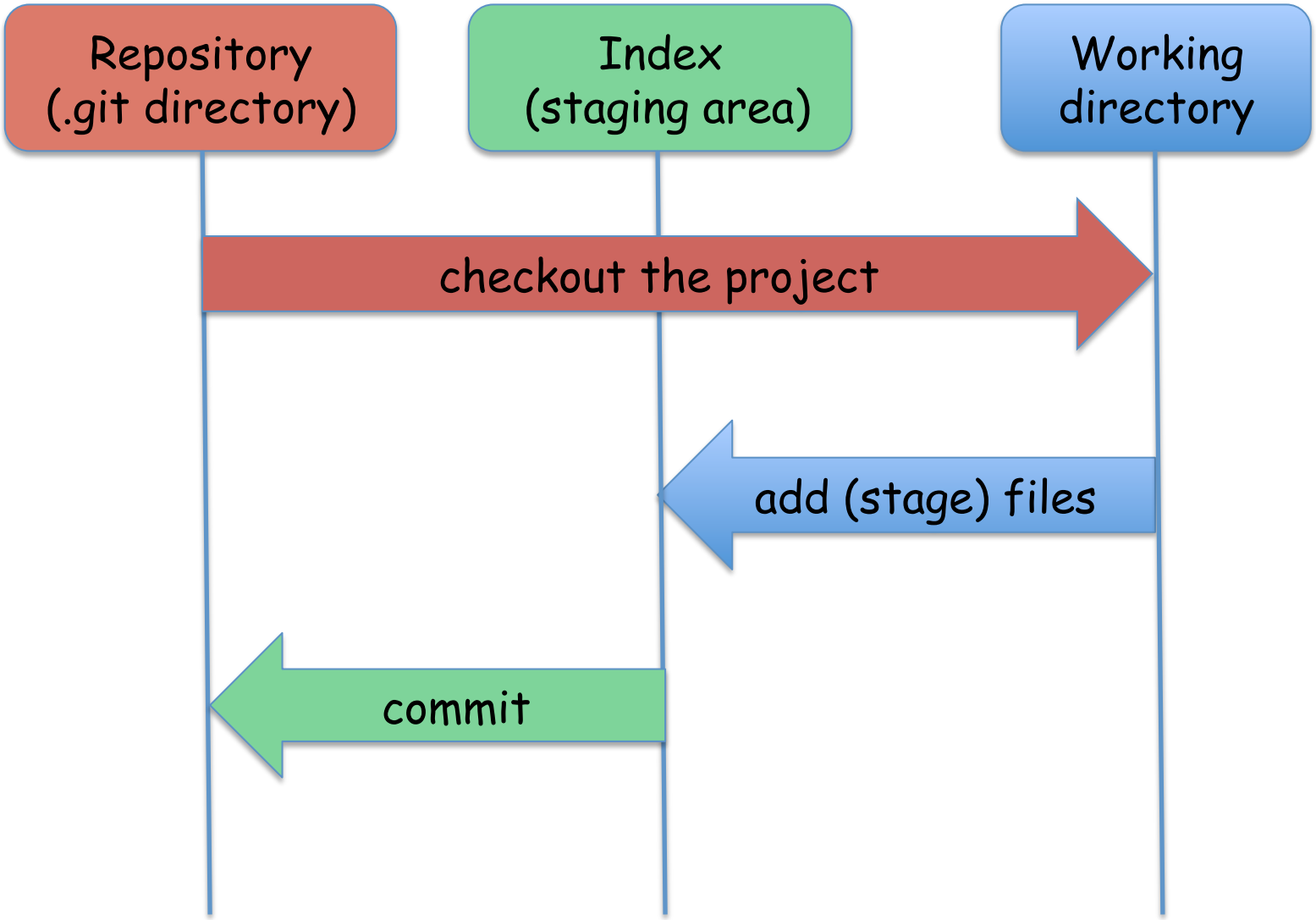
Index

- "staging area"
- what is to be committed

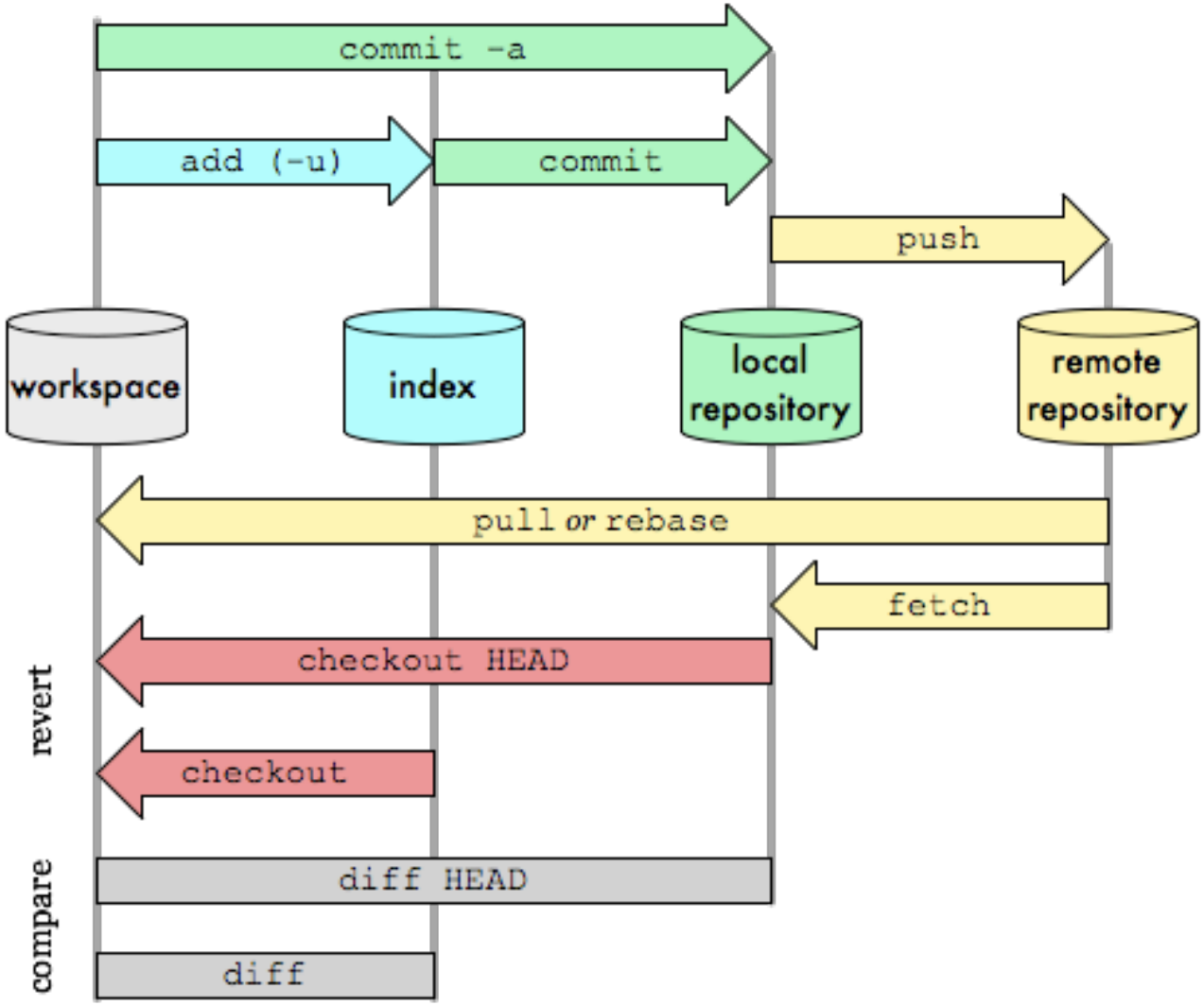


http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Local Operations



Git transport commands



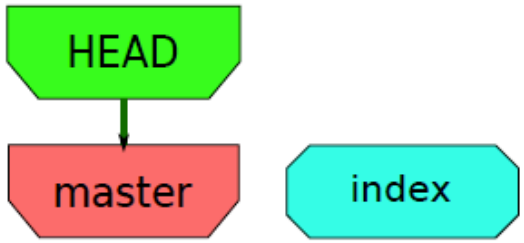
http://osteele.com/images/2008/git-transport.png

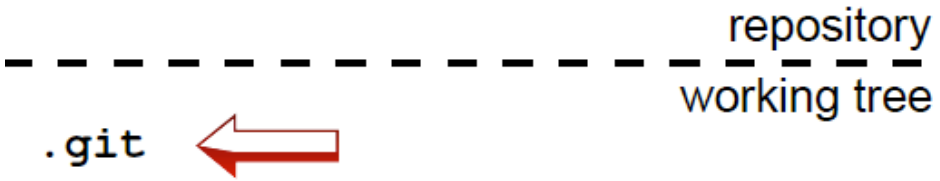
Git init, add, commit

Using basic local operations to work on a single branch.

Bootstrap

```
mkdir project
cd project
git init
```

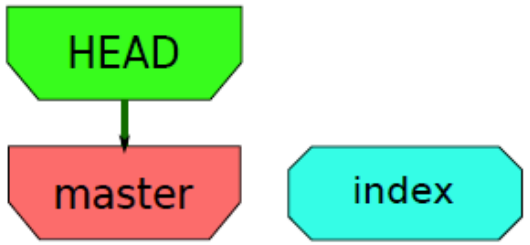


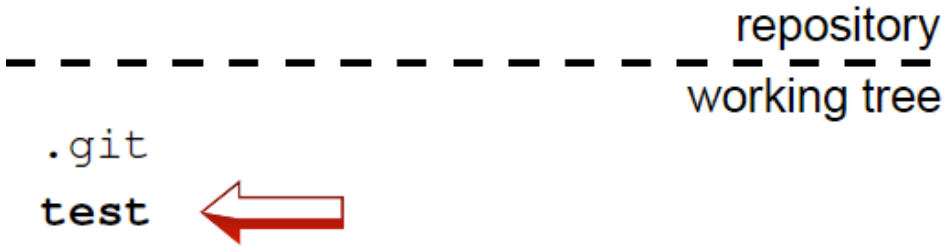


http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Work

```
touch test
```

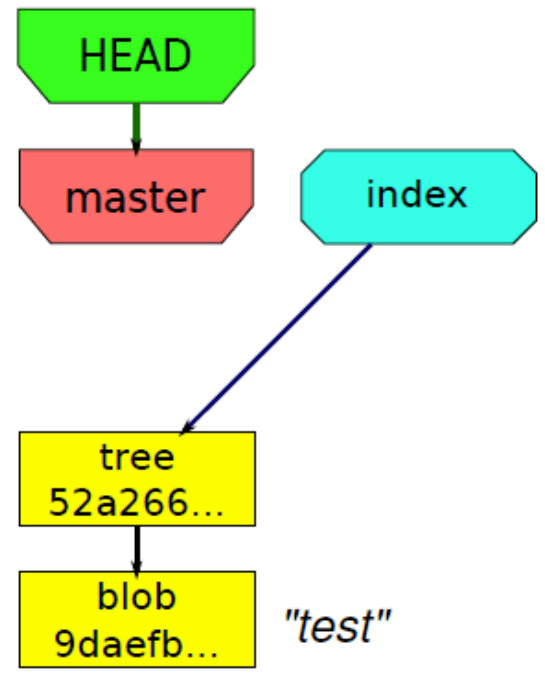


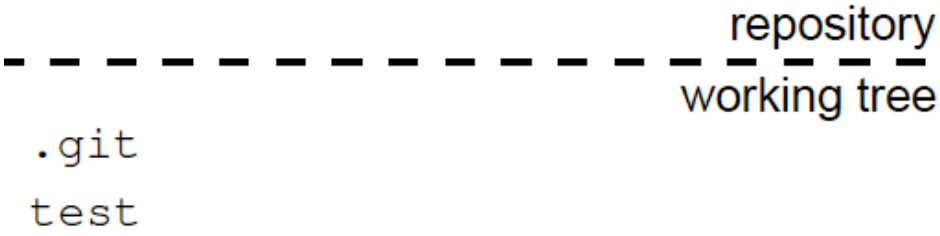


http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Stage

```
touch test
git add test
```

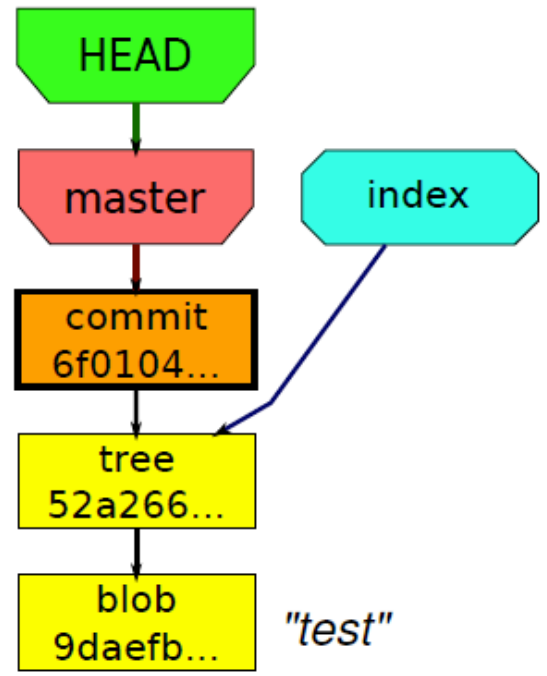


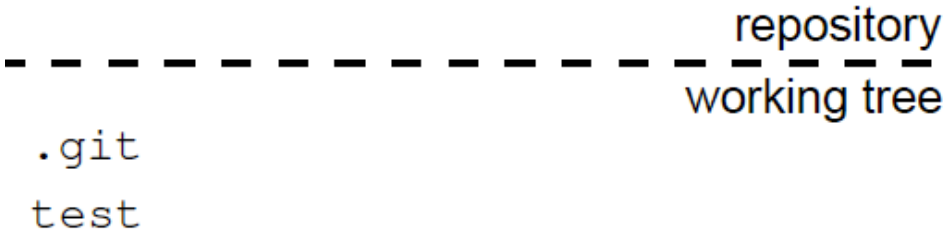


http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Commit

```
touch test
git add test
git commit –m"test"
```

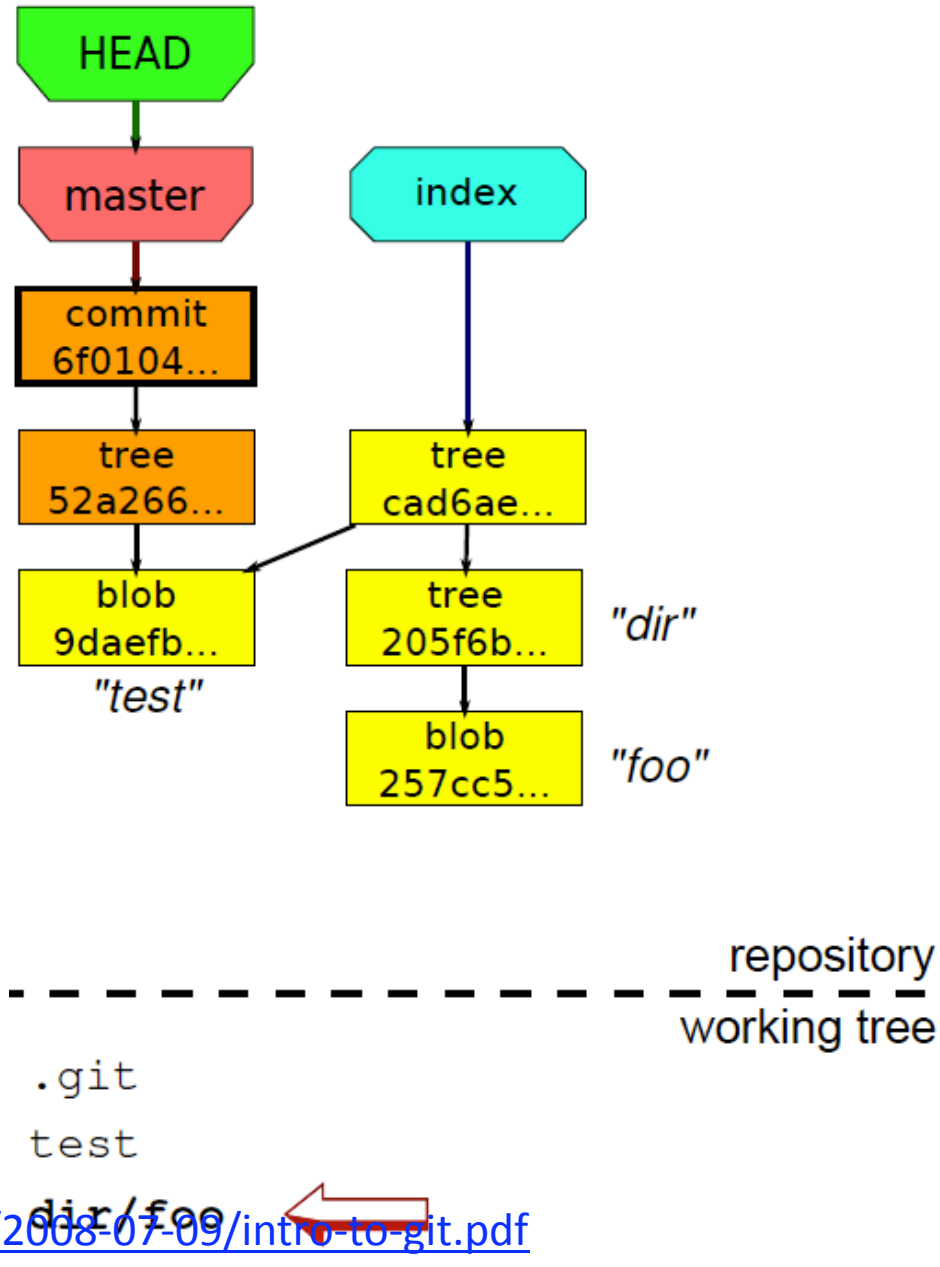




http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Work & Stage

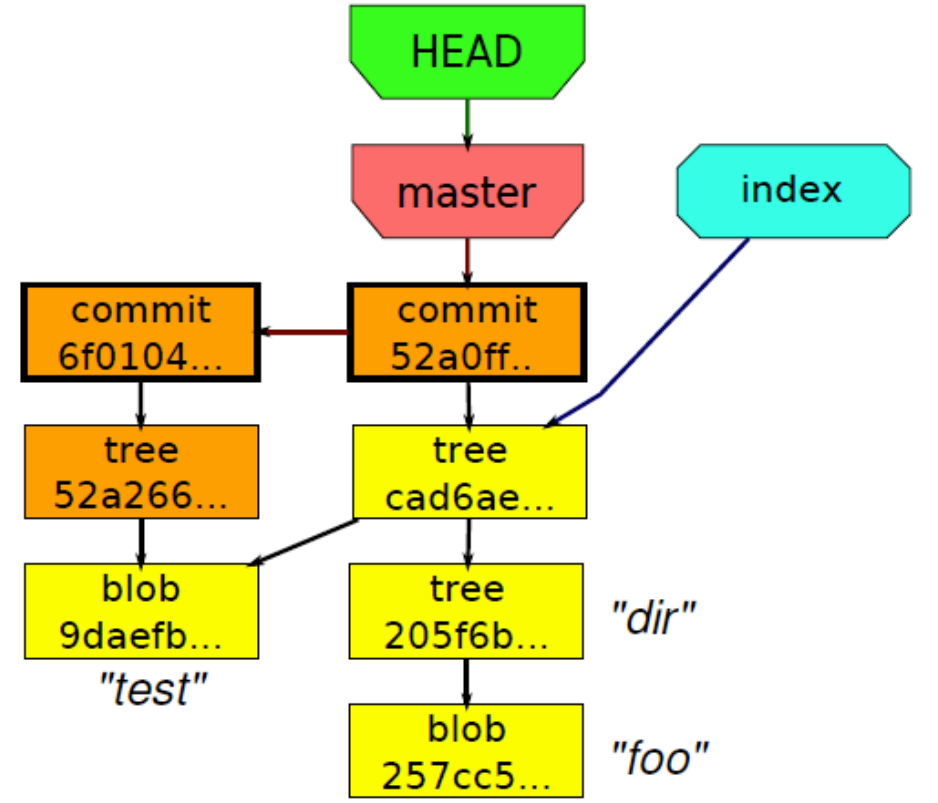
```
touch test
git add test
git commit –m”test”
mkdir dir
echo “foo” > dir/foo
git add dir/foo
```



http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Commit again

```
touch test
git add test
git commit –m”test”
mkdir dir
echo “foo” > dir/foo
git add dir/foo
git commit –m”foo”
```



http://edgyu.excess.org/git-tutorial/2008-07-09/intro-to-git.pdf

Naming

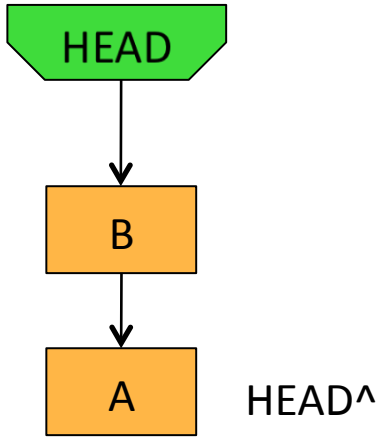
How to refer to a commit

A commit can be identified by

full hash	`6bb1270ffb60cbfef87266d2d4b44abe4218d9c68`
short hash	6bb127
tag	V1.5.6.1
local branch	master
remote branch	origin/master
message	“:/some text:
checkout	HEAD
last fetch	FETCH_HEAD
previous head	ORIG_HEAD
…	

Other naming methods

```
HEAD^, HEAD^^, …
MASTER^, MASTER^^, …
HEAD~1, HEAD~2, …
@{yesterday}, or HEAD@{yesterday}
branchname@{June.1}
master@{3}
```



Getting information

See the state of your repository, and review the history.

git status

Shows what is

- Staged
- Unstaged
- Untracked

git diff

- `git diff`
  - index vs. working files
- `git diff –staged`
  - HEAD vs. index
- `git diff HEAD`
  - HEAD vs. working files
- `git diff <commit1> <commit2>`

git show

`git show`
  - summarizes the last commit

`git show –stat`
  - shows just the statistics

`git show –name-status`
  - shows status

- Can apply to any commit

```
git show HEAD
git show master^^
…
```

- Or to a file

```
git show HEAD:file
```

git log

- Shows history of changes
- Can apply to any single commit or range

```
git log
git log tag..branch
git log HEAD~10..
git log ~10
```

- Or attributes of commits, etc. etc.

```
git log --since=“May 1” –until=“June 1”
git log --author=fred
git log --grep-”commit.*message.*text”
…
```

git grep

```
git grep –e “pattern” -- some/file
git grep –e “pattern” branch -- some/file
```

Viewing references

- Named
- Refer to commits
- Can be moved
- 3 basic types:
  - Tags
  - Local branches
  - Remote branches

Tags

To list tags: `git tag –l`

```
web2py1.74.11
web2py1.74.9
web2py1.75.2
```

Or look at the files in `.git/refs/tags/`

Local branches

To list them: `git branch –l`

```
    branch1
    branch2
  * master
```

Or look at files in `.git/refs/heads/`

Remote branches

To see them: `git branch –r`

```
origin/HEAD -> origin/master
origin/master
origin/update
```

Or look at files in `.git/refs/remotes/`