Preliminary version – to be updated some day, I hope.

Git

Slides constructed from http://excess.org/article/2008/07/ogre-git-tutorial/ , Dr Sara Stoecklin's notes in SCM, and other sources.

Git

- A collection of tools developed by Linux kernel group for SCM
  - Now used by several other groups, and apparently growing in popularity
- Actually implements a replicated versioned file system
- Can be used to implement a variety of software configuration management models and workflows

Git Flavor

- A collection of <u>many</u> tools
- Evolved from scripts
- Suited to a C programmer's mentality
- Everything is exposed and accessible
- Need to understand the underlying model
- Very flexible
  - You can do anything the model permits
  - Including shooting yourself in the foot

Git has a lot of commands

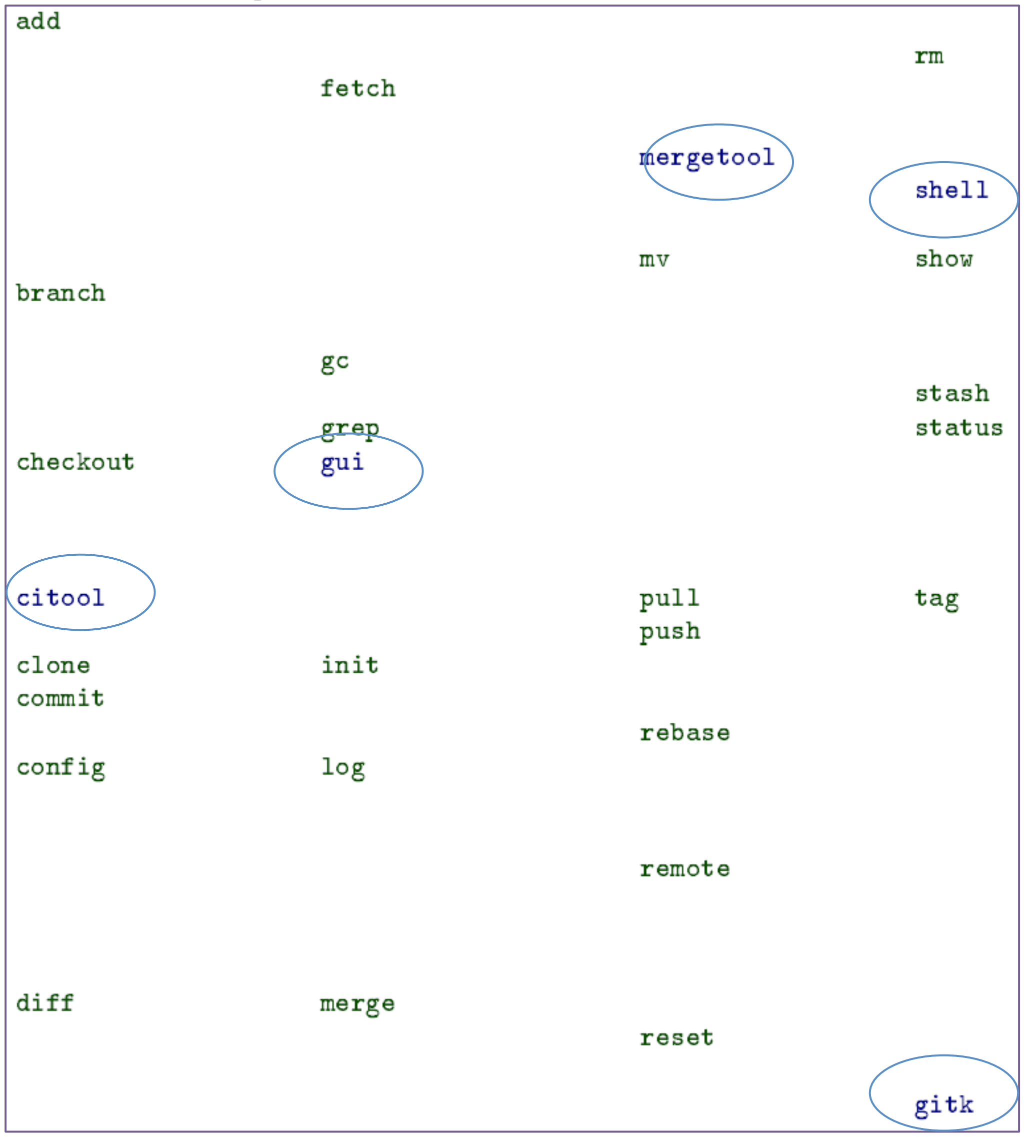
add	fast-export	merge-one-file	revert
am	fast-import	merge-resolve	rm
annotate	fetch	merge-subtree	send-email
apply	fetch-pack	merge-tree	send-pack
archimport	filter-branch	mergetool	sh-setup
archive	fmt-merge-msg	mktag	shell
bisect	for-each-ref	mktree	shortlog
blame	format-patch	mv	show
branch	fsck	name-rev	show-branch
bundle	fsck-objects	pack-objects	show-index
cat-file	gc	pack-redundant	show-ref
check-attr	get-tar-commit-id	pack-refs	stash
check-ref-format	grep	parse-remote	status
checkout	gui	patch-id	stripspace
checkout-index	hash-object	peek-remote	submodule
cherry	http-fetch	prune	svn
cherry-pick	http-push	prune-packed	symbolic-ref
citool	imap-send	pull	tag
clean	index-pack	push	tar-tree
clone	init	quiltimport	unpack-file
commit	init-db	read-tree	unpack-objects
commit-tree	instaweb	rebase	update-index
config	log	receive-pack	update-ref
count-objects	lost-found	reflog	update-server-info
cvsexportcommit	ls-files	relink	upload-archive
cvsimport	ls-remote	remote	upload-pack
cvsserver	ls-tree	repack	var
daemon	mailinfo	repo-config	verify-pack
describe	mailsplit	request-pull	verify-tag
diff	merge	rerere	whatchanged
diff-files	merge-base	reset	write-tree
diff-index	merge-file	rev-list	
diff-tree	merge-index	rev-parse	gitk

but you can get by with a subset for everyday use

`add`
`branch`
`checkout`
`clone`
`commit`
`config`
`diff`
`fetch`
`gc`
`grep`
`init`
`log`
`merge`
`mv`
`pull`
`push`
`rebase`
`remote`
`reset`
`rm`
`show`
`stash`
`status`
`tag`

and maybe a few more gui tools

- gitk
- mergetool
- gui
- citool
- shell



or maybe a few more occasionally

```
add                                             revert
am                                              rm
annotate        fetch                           send-email
apply
                                mergetool
archive                                         shell
bisect                                          shortlog
blame           format-patch    mv              show
branch                                          show-branch
bundle
                gc
                                                stash
                grep                            status
checkout        gui
                                                submodule
cherry                                          svn
cherry-pick
citool                          pull            tag
clean                           push
clone           init            quiltimport
commit
                instaweb        rebase
config          log
                                reflog          update-server-info

                                remote



describe
diff            merge                           whatchanged
                                reset

                                                gitk
```

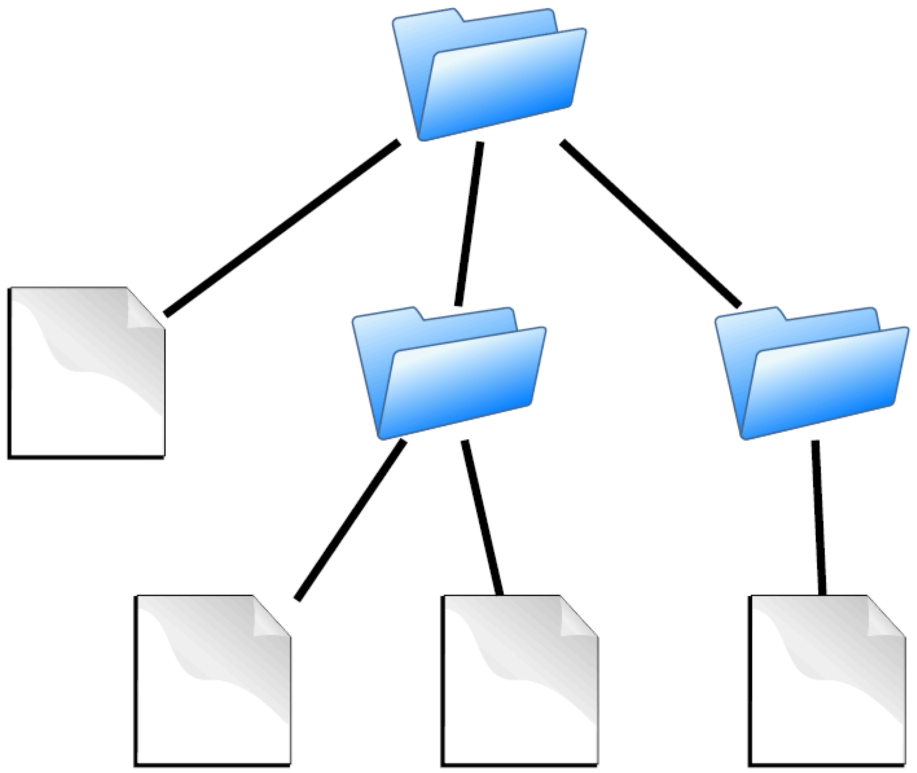
Groups of Git operations

- Setup and branch-switching
  - init, checkout, switch branch
- Modification
  - add, delete, rename, commit
- Getting information
  - status, diff, log
- Create reference points
  - tag, branch

Source code

contains

- Directories
- Files

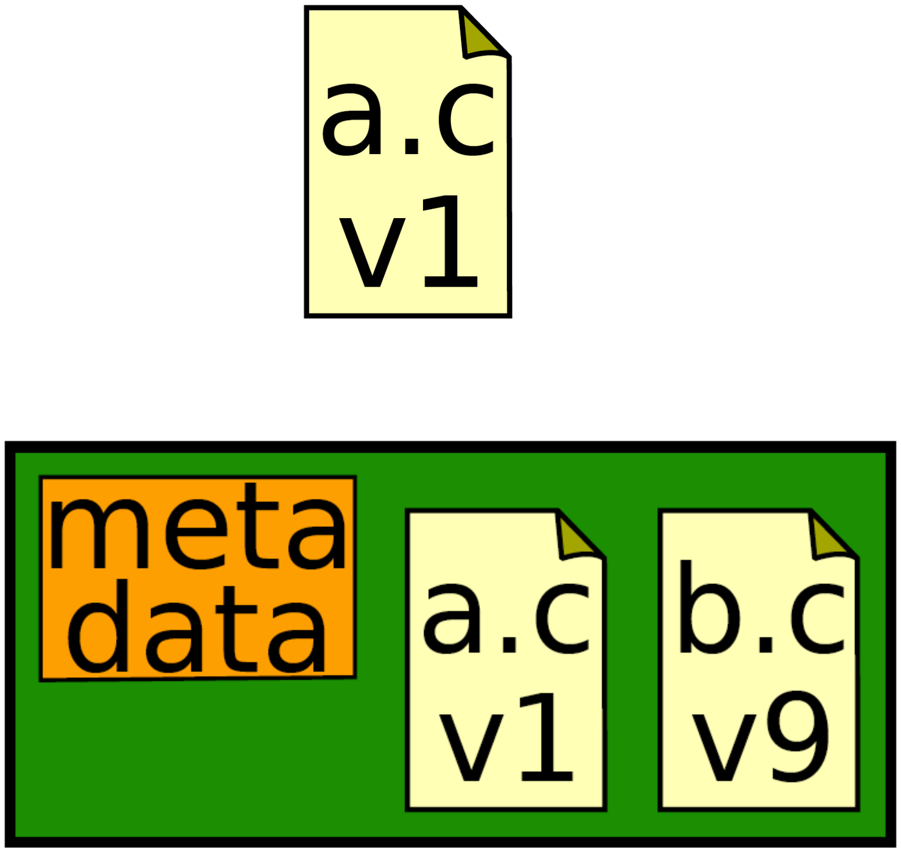


is the substance of a software configuration

Repository

Contains

- files
- commits

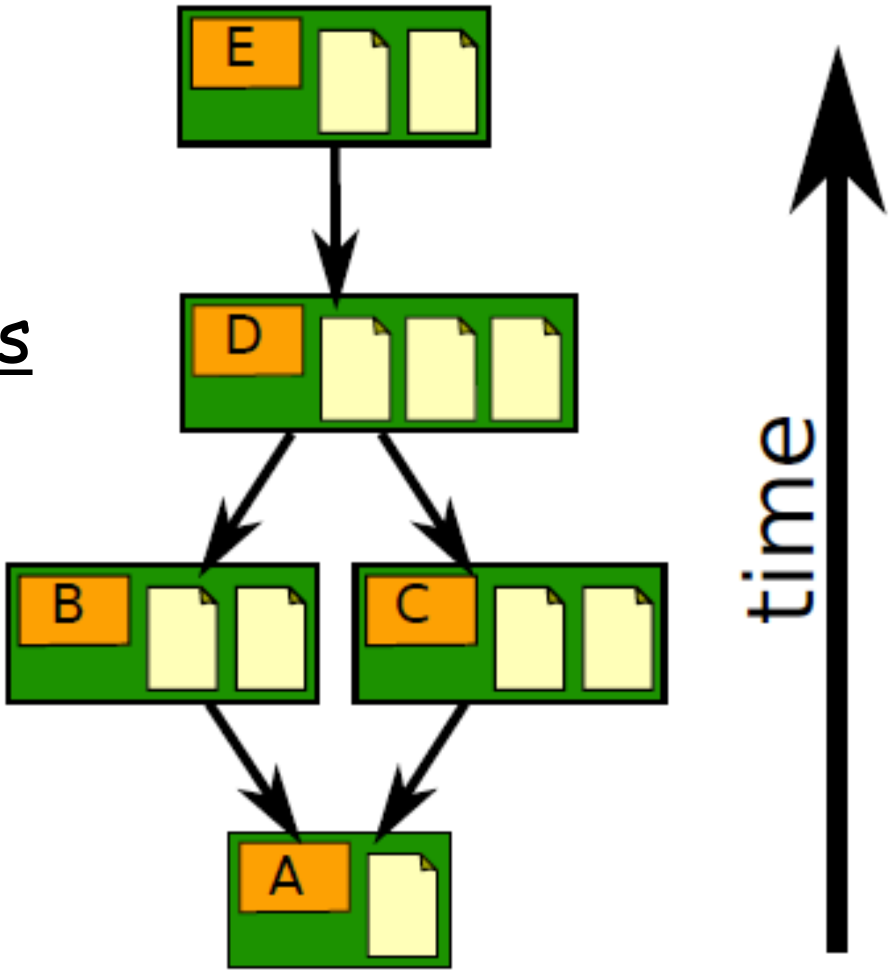


records history of changes to configuration

Repository

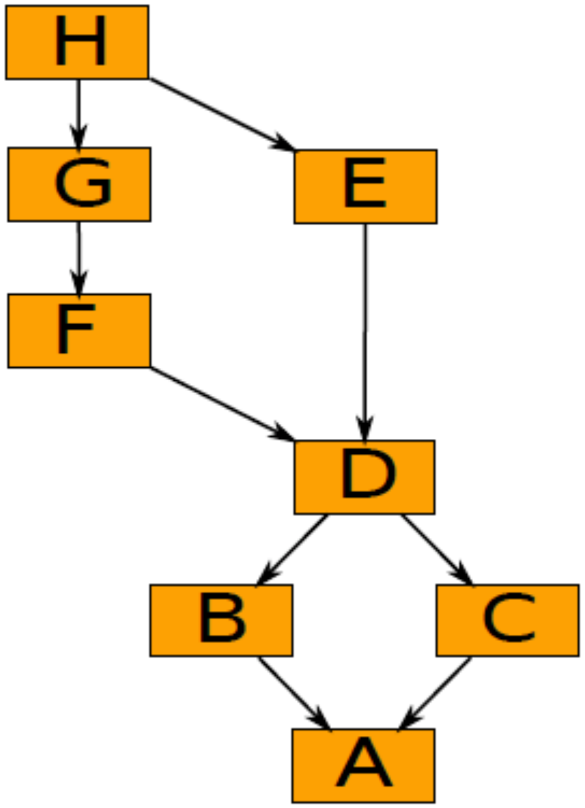
Contains

- files
- commits
- <u>ancestry relationships</u>



Ancestry relationships

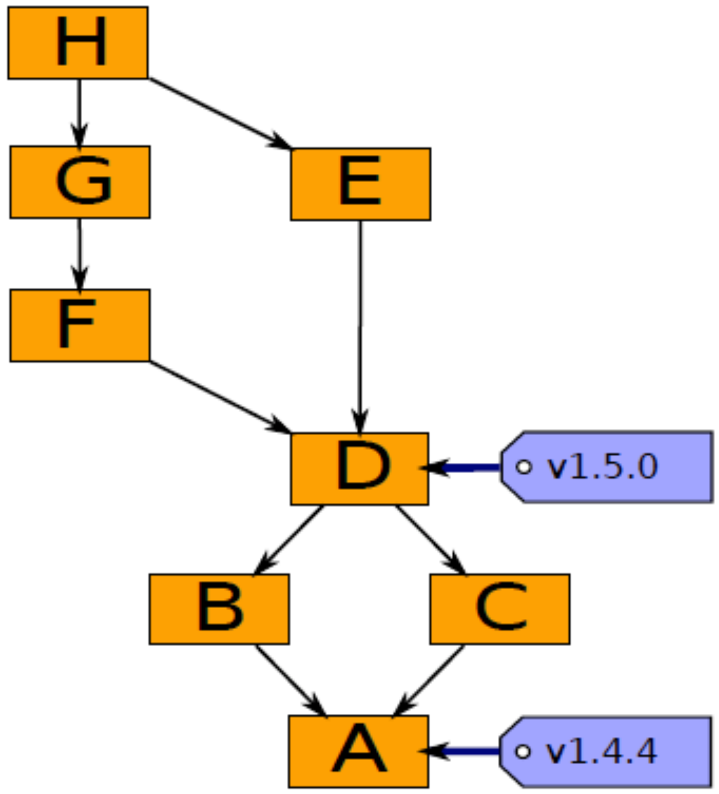
form a directed acyclic graph (DAG)



Ancestry graph features

Tags

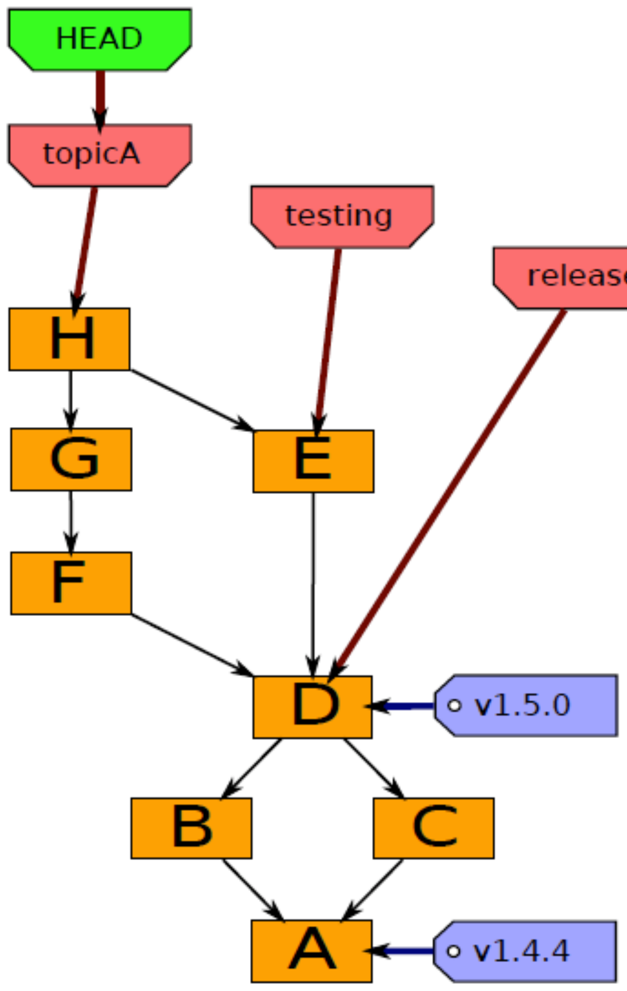
- identify versions of interest
- including "releases"



Ancestry graph features

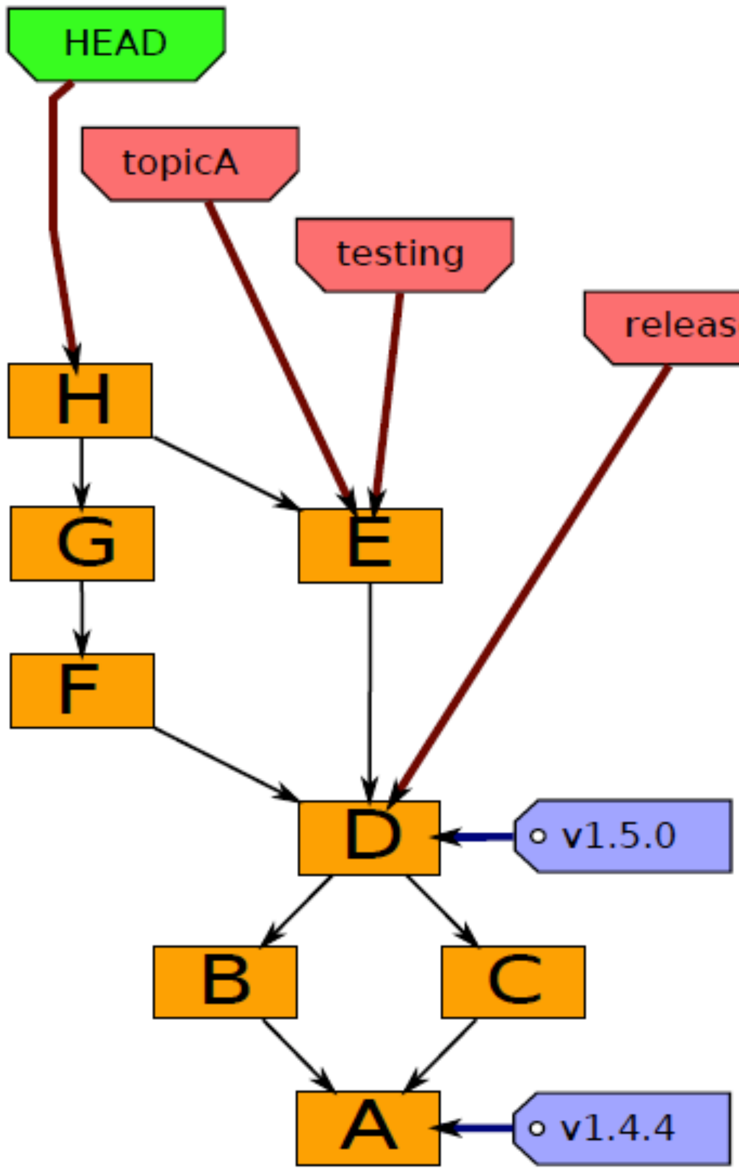
HEAD

- is current checkout
- usually points to a branch



Head may point to any commit

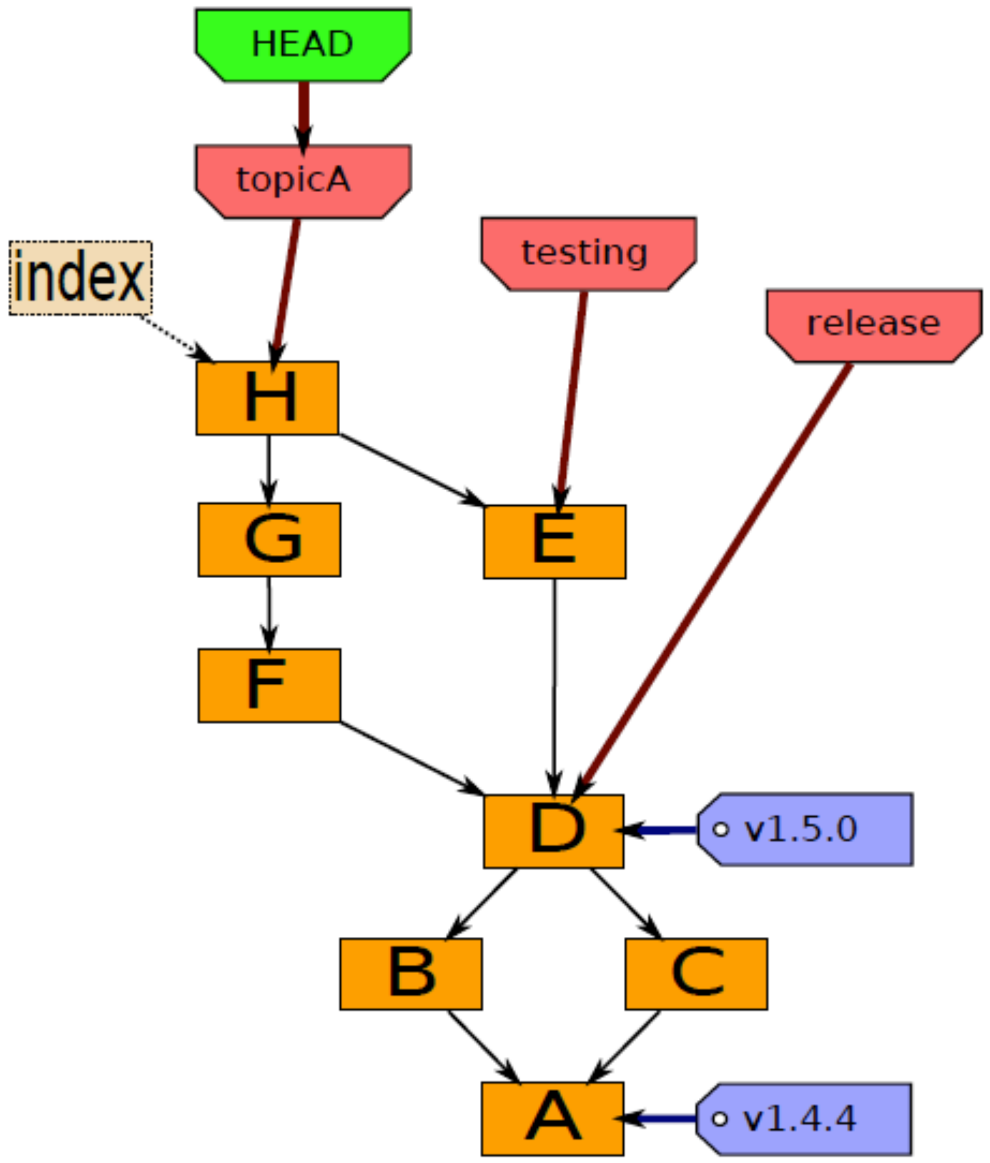
In this case it is
said to be detached.



Git components

Index

– "staging area"
– what is to be committed

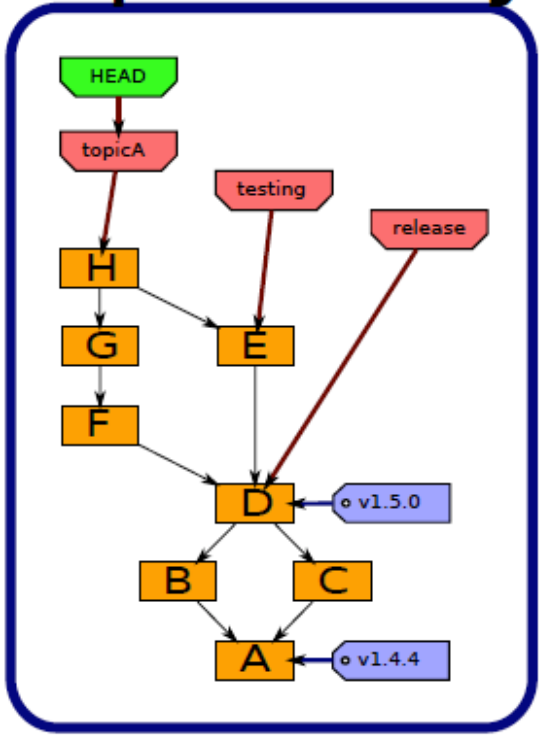


Working directory, Index, and Repository

Three top-level abstractions

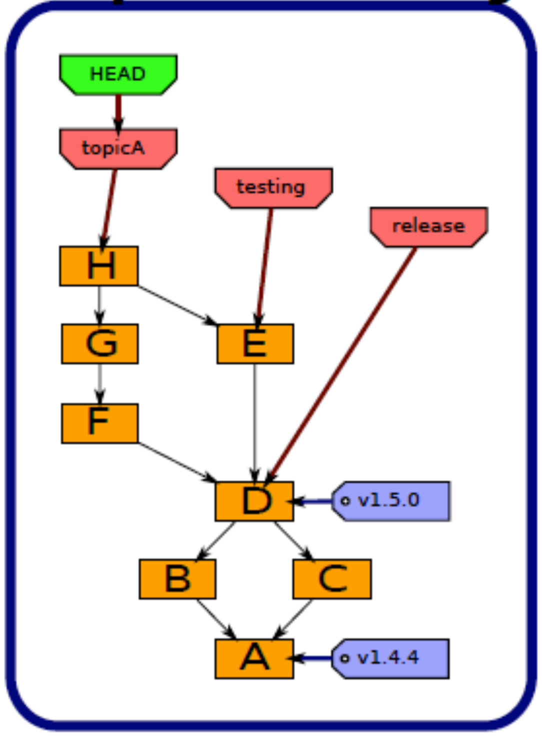
History

repository



Staging area

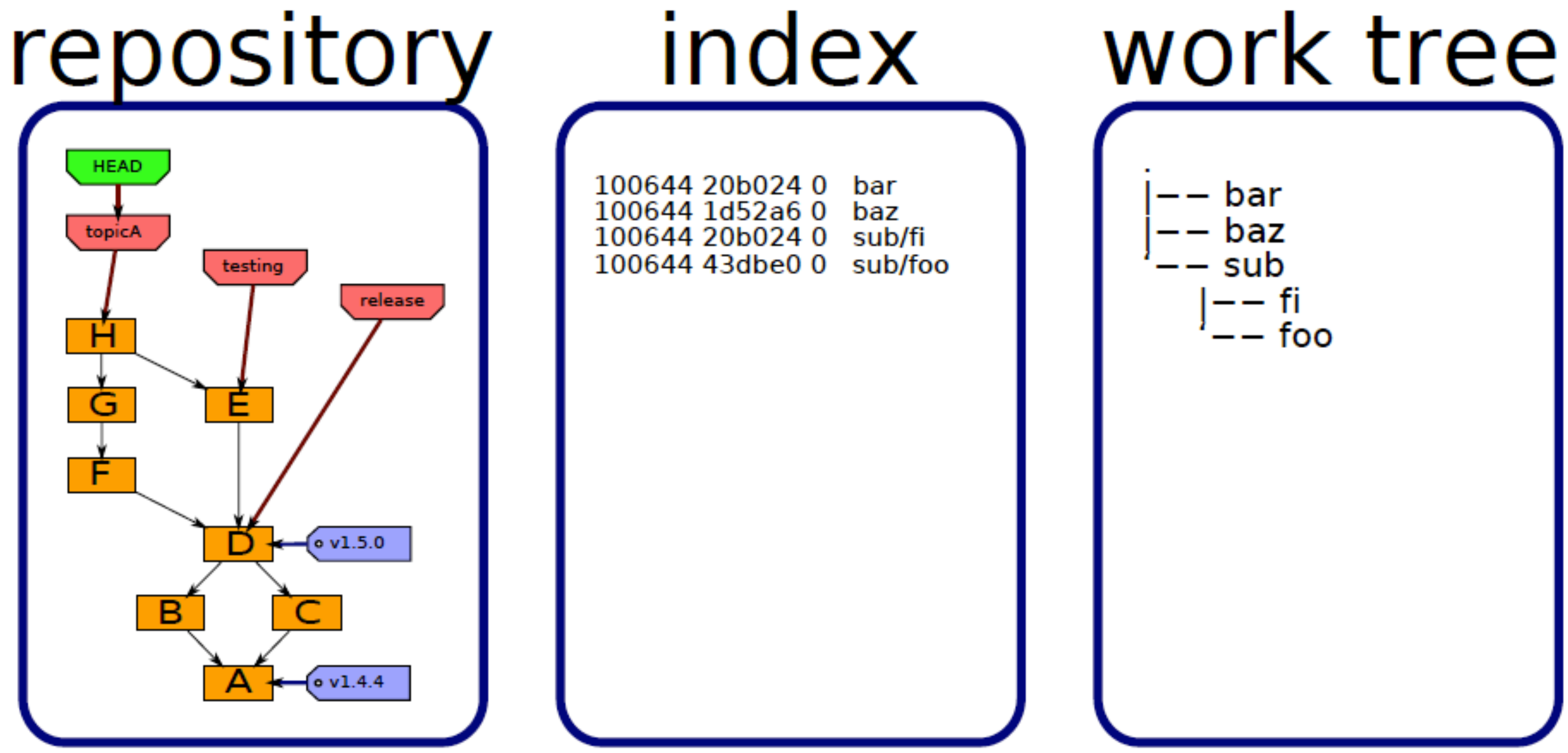
repository



index

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

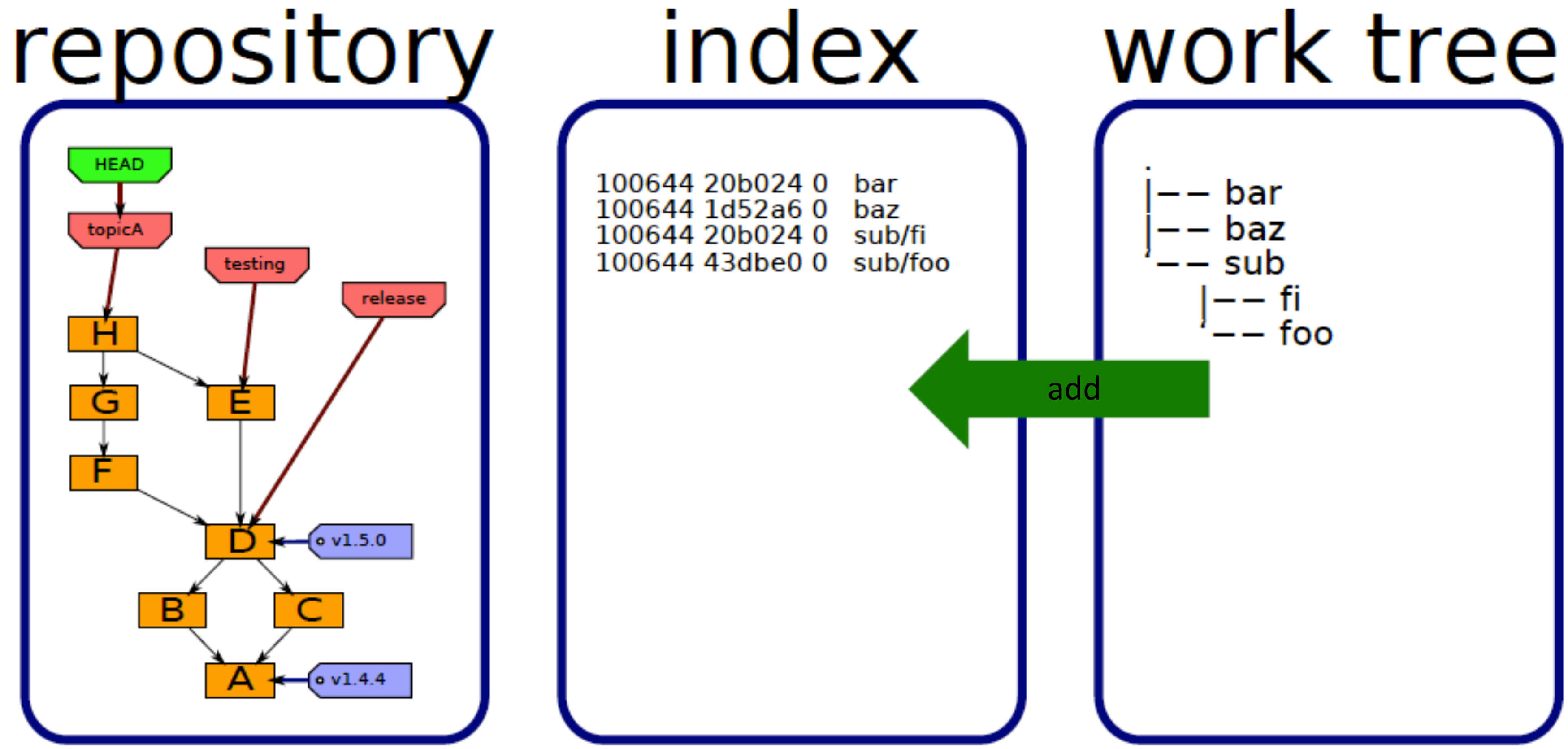
Files you edit



```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

```
.
|-- bar
|-- baz
`-- sub
    |-- fi
    `-- foo
```

Staging

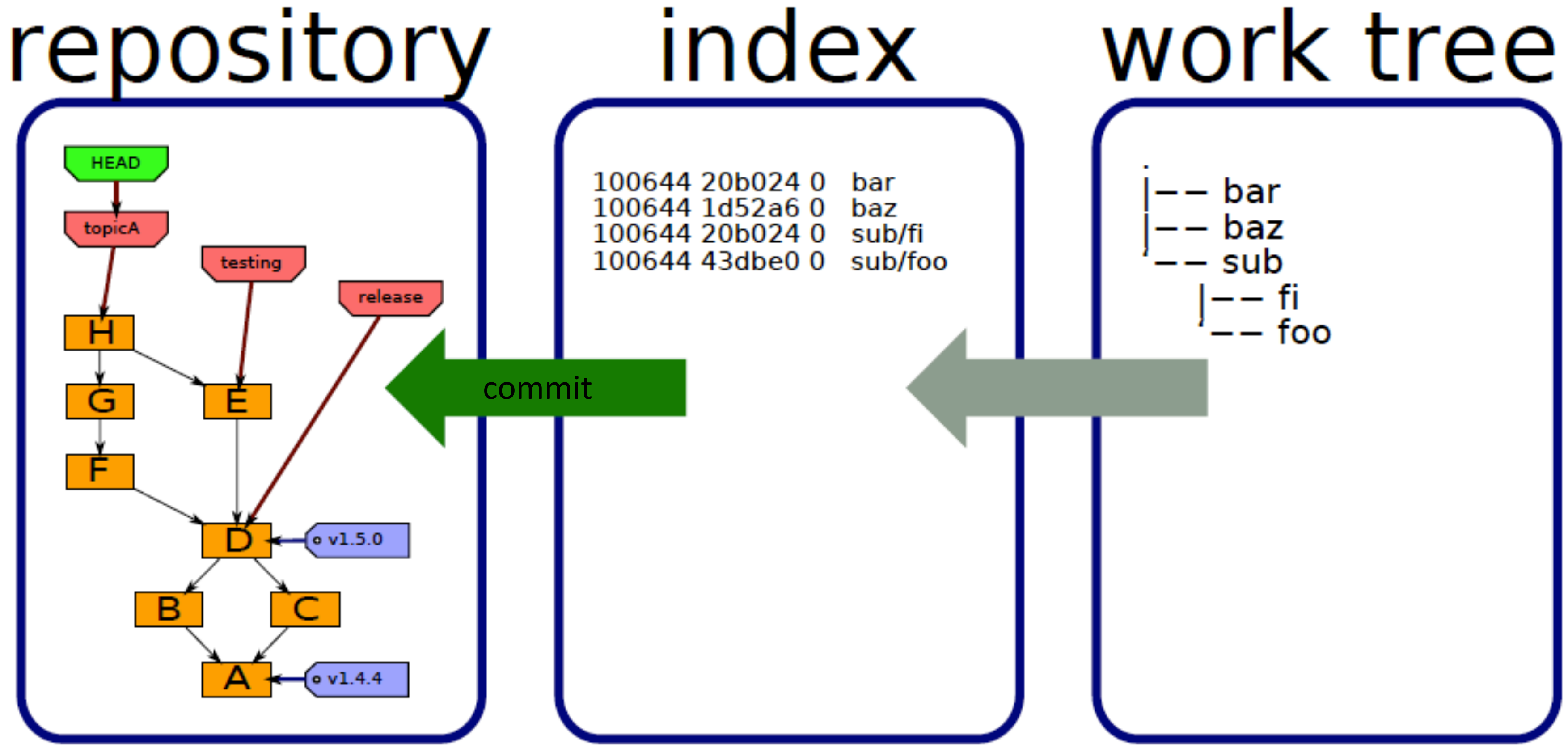


```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

```
.
|-- bar
|-- baz
`-- sub
    |-- fi
    `-- foo
```

`add, remove, rename`

Committing

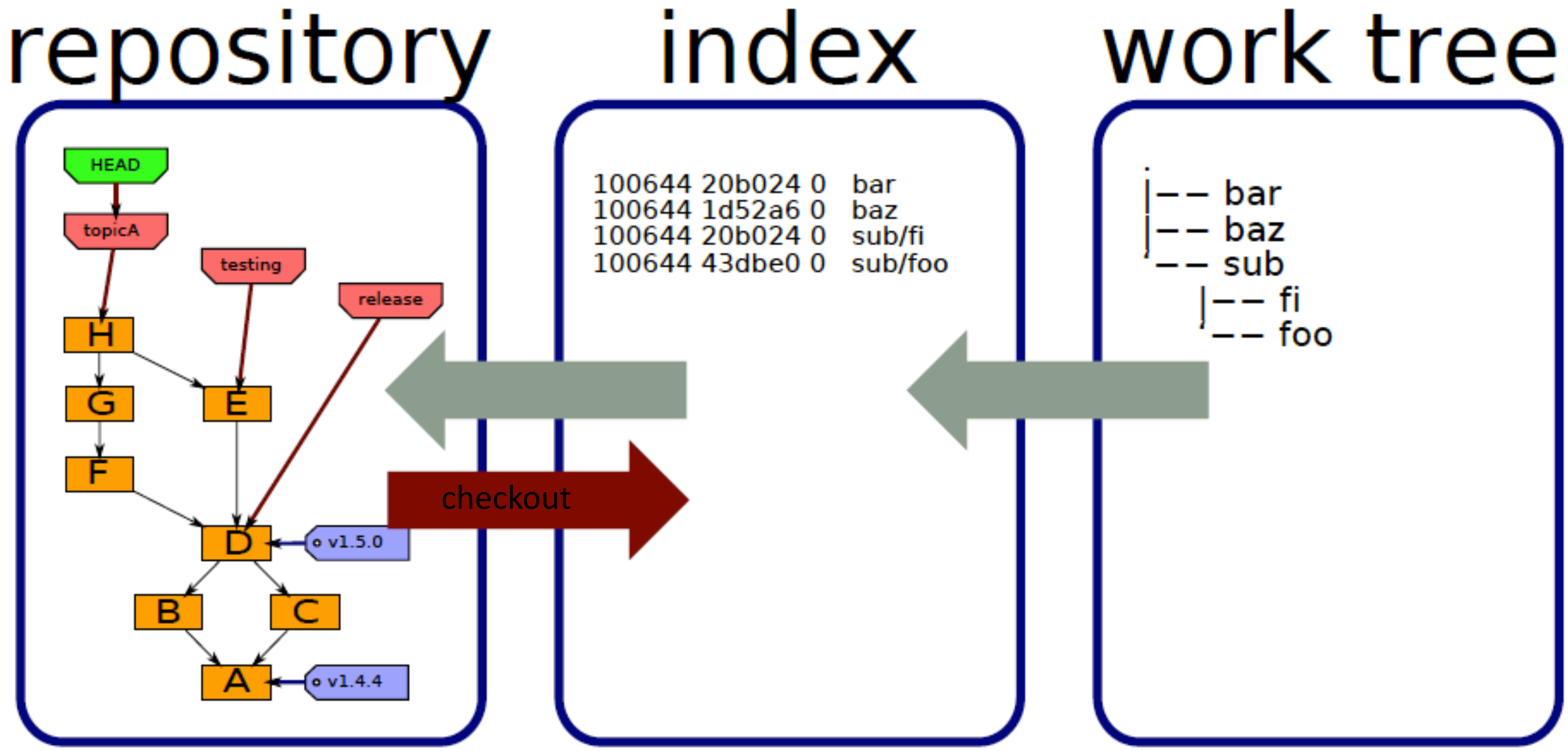


```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

```
.
|-- bar
|-- baz
`-- sub
    |-- fi
    `-- foo
```

`commit`

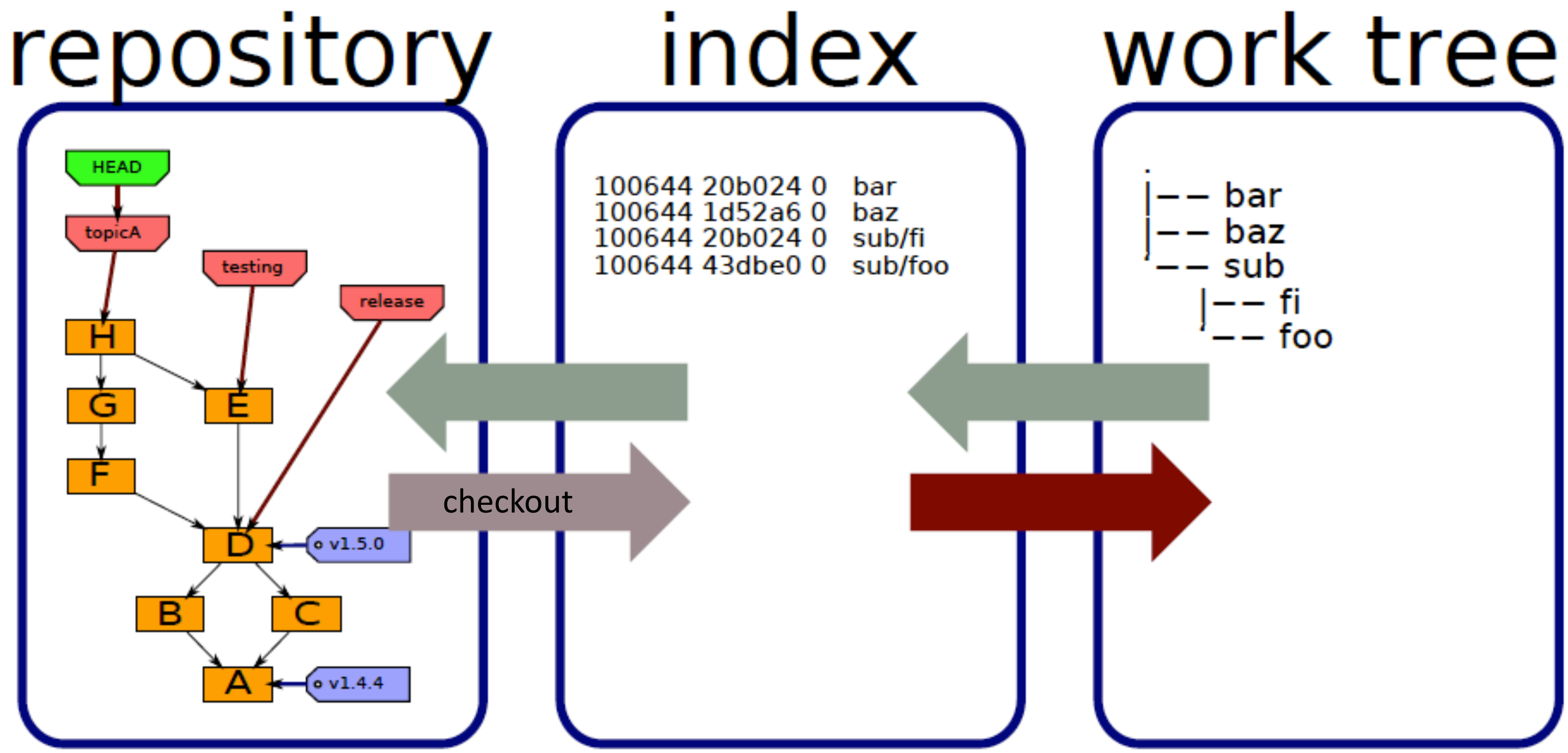
Reading tree



```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

checkout, read-tree, reset

Checking out



`checkout, checkout-index, reset`

The repository

```
.git
|-- HEAD                      current checkout reference
|-- config                    repo private config
|-- description               repo description
|-- hooks
|   `-- ...                   hooking scripts
|-- index                     changes to commit
|-- info
|   |-- exclude               repo private
|   `-- refs                  refs?
|-- logs
|   `-- ...                   "reflog" data
|-- objects
|   |-- XX
|   |   `-- ...               loose objects
|   |-- info
|   |   `-- packs             info about packs
|   `-- pack
|       `-- ...               packs and indexes
`-- refs
    |-- heads
    |   `-- master            master branch
    `-- tags
        `-- ...               tags
```

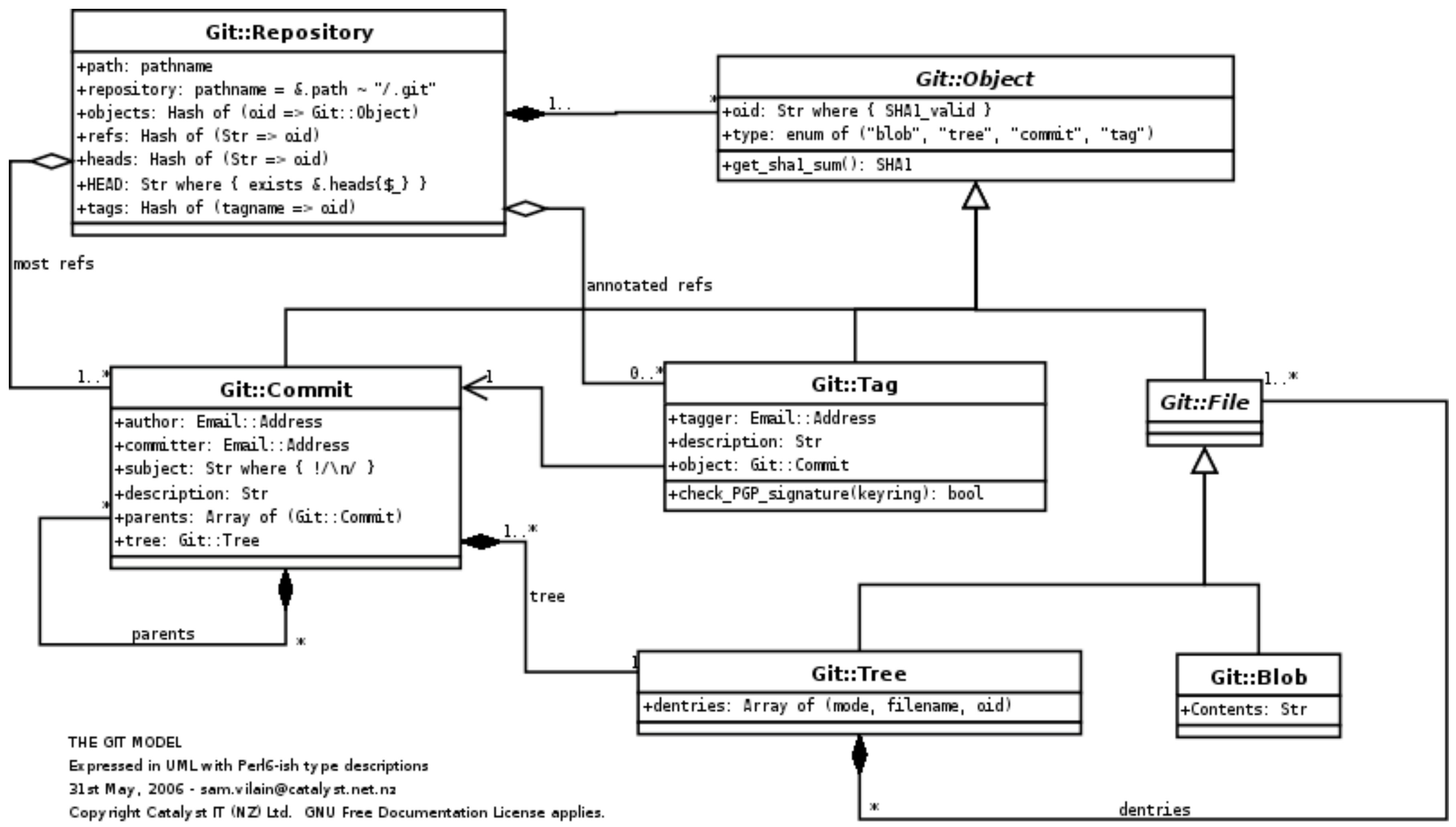
Repository files

- .git/config
- .git/description – used by gitweb
- .git/info/exclude – files to ignore

.git/objects

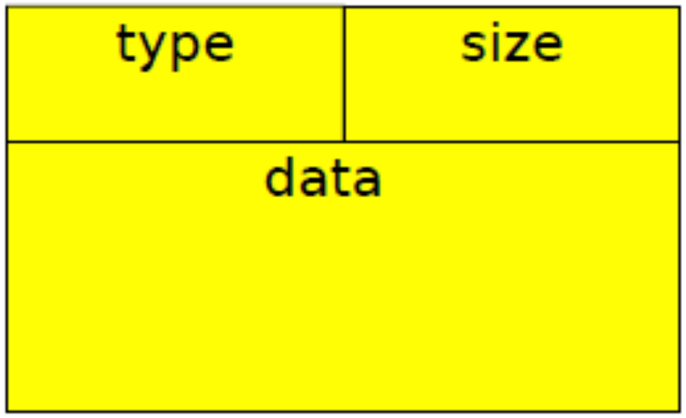
```
|-- 23
|   '-- d4bd826aba9e29aaace9411cc175b784edc399
|-- 76
|   '-- 49f82d40a98b1ba59057798e47aab2a99a11d3
|-- c4                                              loose
|   '-- aaefaa8a48ad4ad379dc1002b78f1a3e4ceabc
|-- e7
|   '-- 4be61128eef713459ca4e32398d689fe80864e
|-- info
|   '-- packs
'-- pack
    |-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.idx
    '-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.pack
```

Git object model

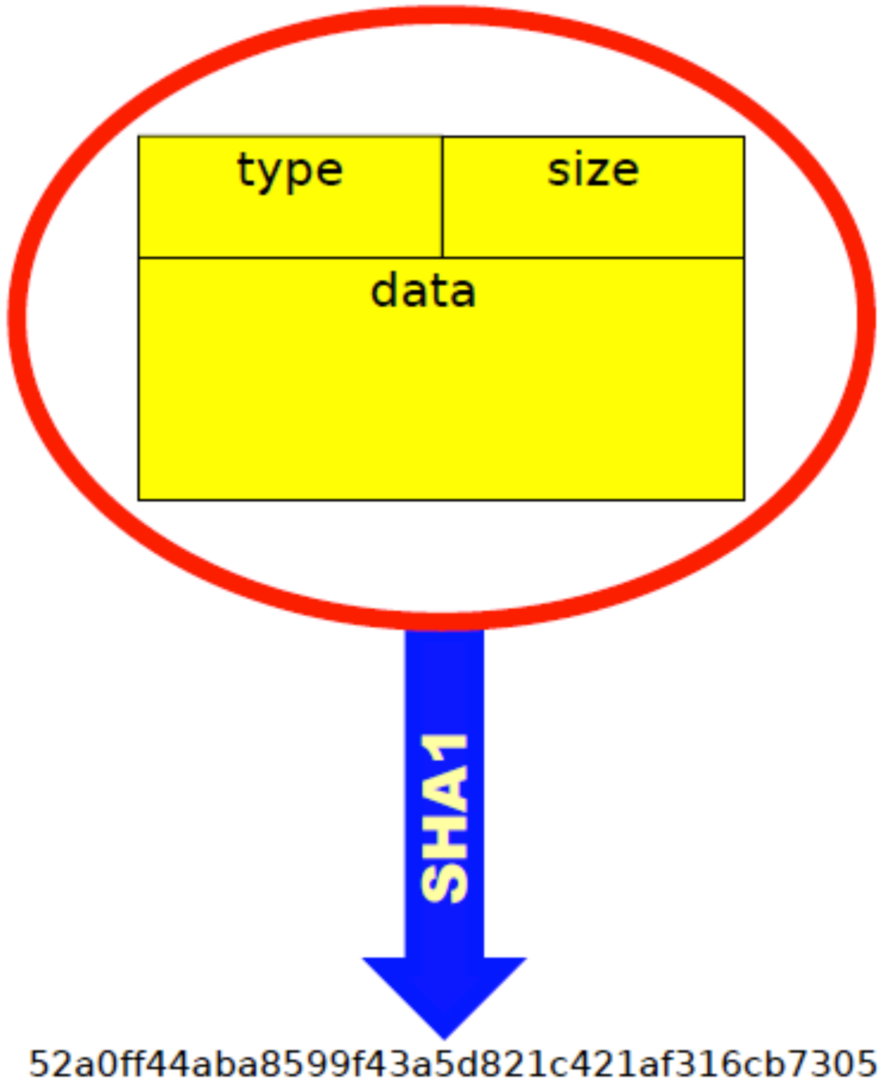

Git::Repository
+path: pathname
+repository: pathname = &.path ~ "/.git"
+objects: Hash of (oid => Git::Object)
+refs: Hash of (Str => oid)
+heads: Hash of (Str => oid)
+HEAD: Str where { exists &.heads{$_} }
+tags: Hash of (tagname => oid)
Git::Object
+oid: Str where { SHA1_valid }
+type: enum of ("blob", "tree", "commit", "tag")
+get_sha1_sum(): SHA1
1..
*
most refs
annotated refs
Git::Commit
+author: Email::Address
+committer: Email::Address
+subject: Str where { !/\n/ }
+description: Str
+parents: Array of (Git::Commit)
+tree: Git::Tree
1..*
1
0..*
Git::Tag
+tagger: Email::Address
+description: Str
+object: Git::Commit
+check_PGP_signature(keyring): bool
Git::File
1..*
parents
tree
Git::Tree
+dentries: Array of (mode, filename, oid)
Git::Blob
+Contents: Str
dentries
THE GIT MODEL
Expressed in UML with Perl6-ish type descriptions
31st May, 2006 - sam.vilain@catalyst.net.nz
Copyright Catalyst IT (NZ) Ltd. GNU Free Documentation License applies.

Repository object naming convention

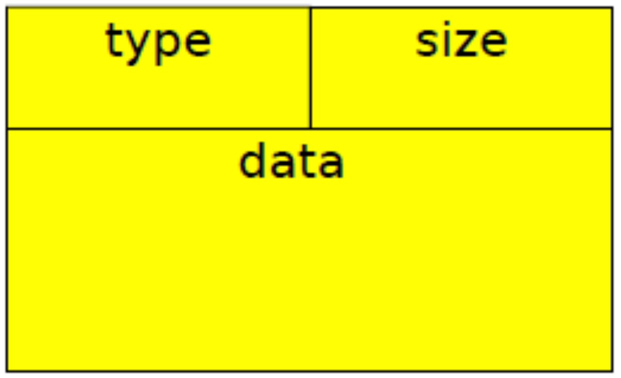
"content addressable" (hashed)

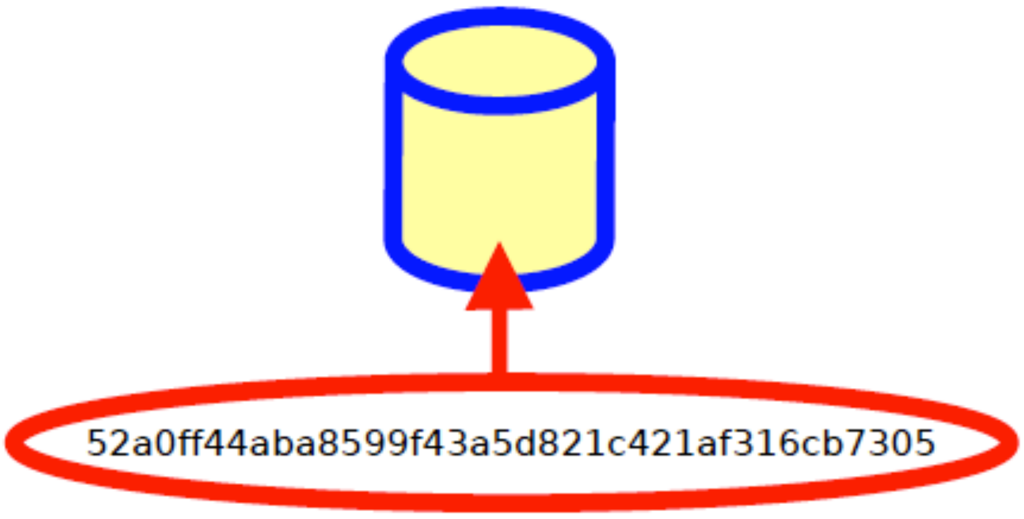


Data values determine hash

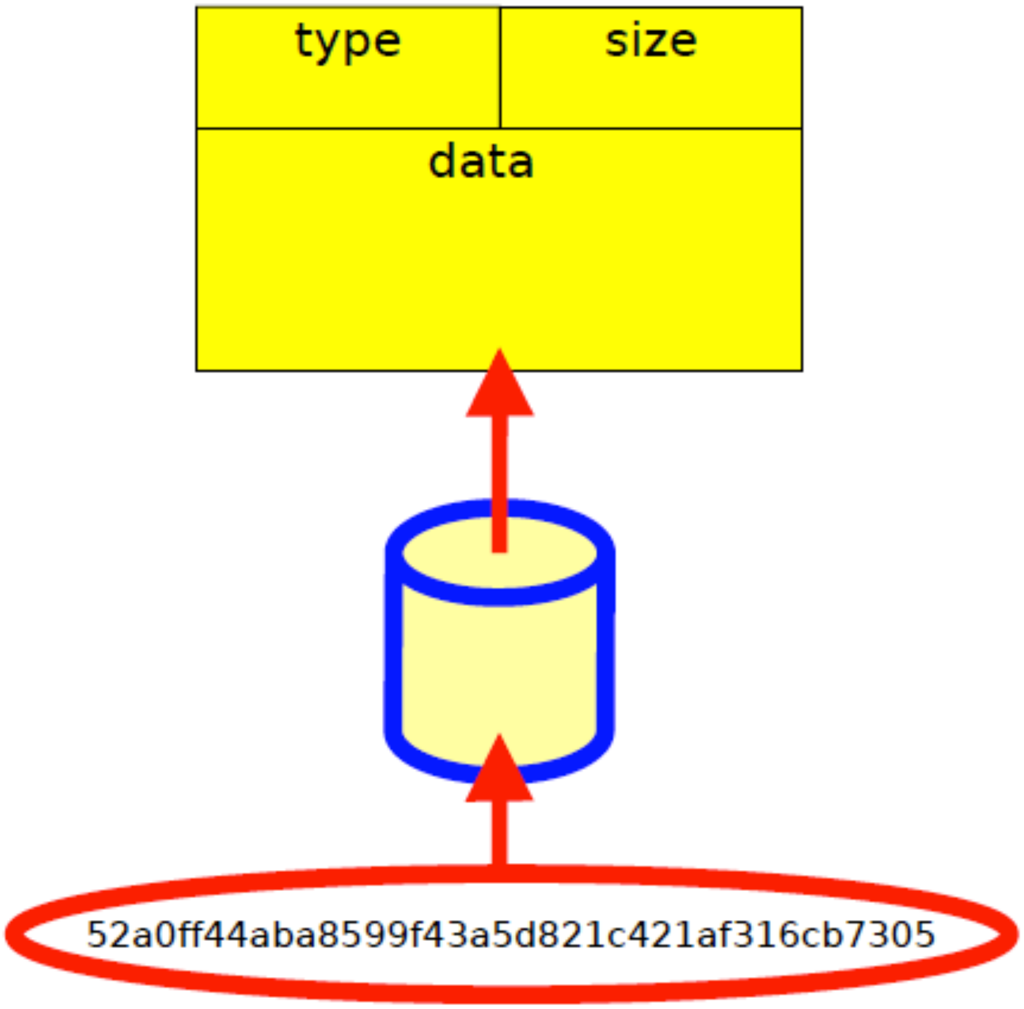


Hash value is filename





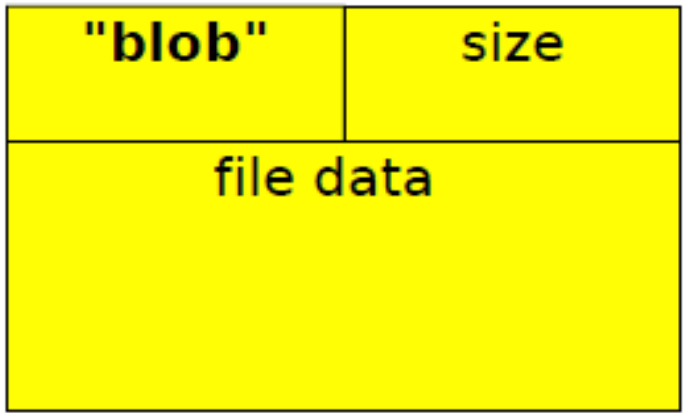
File contains data



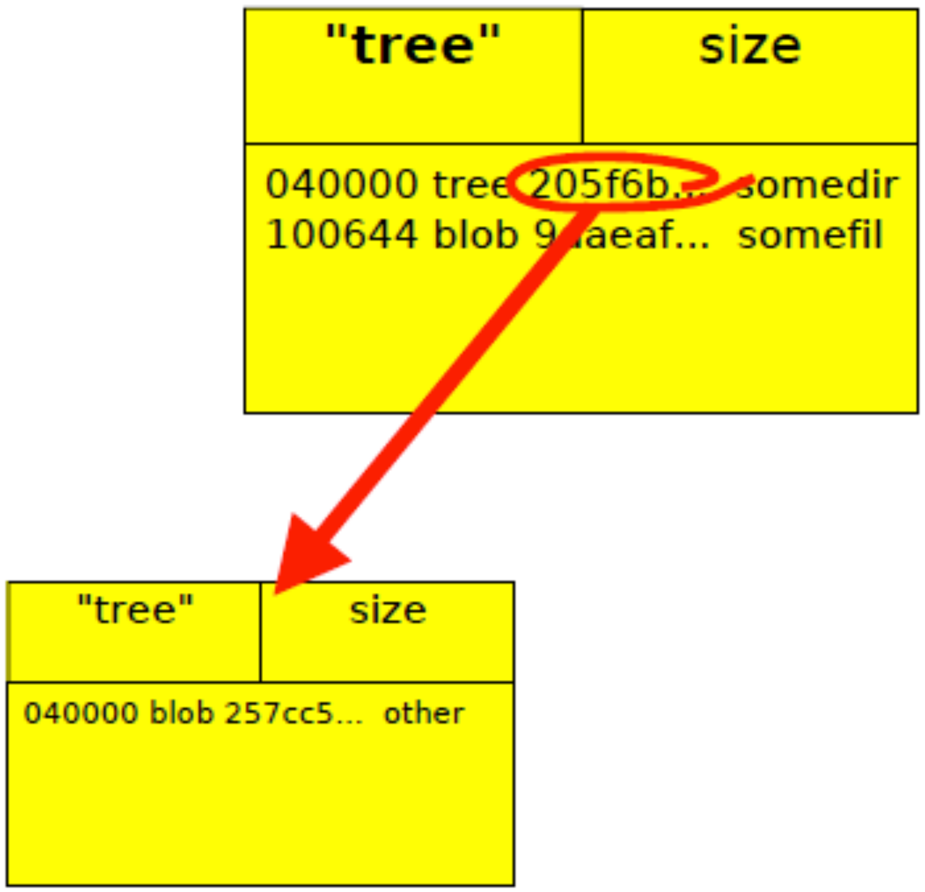
Object types

- Blobs
- Trees
- Commits
- Tags

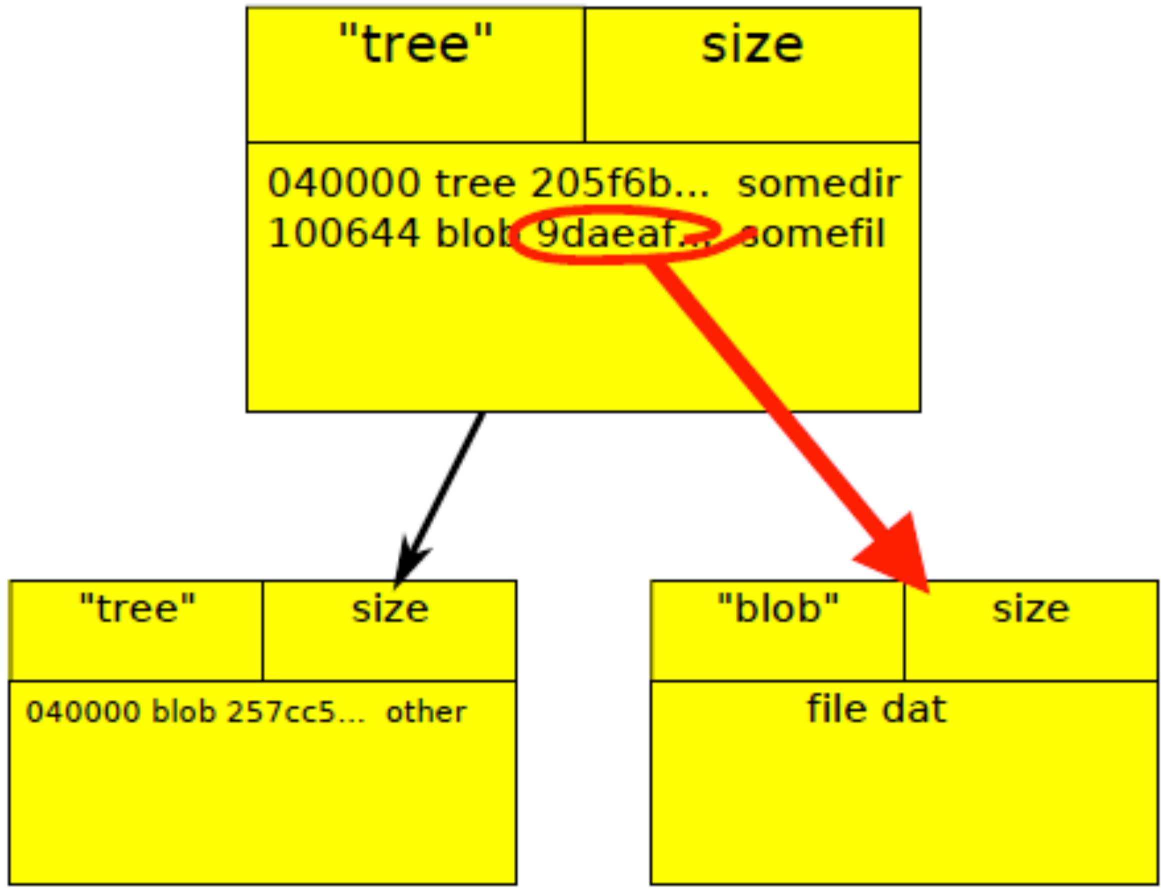
Blobs



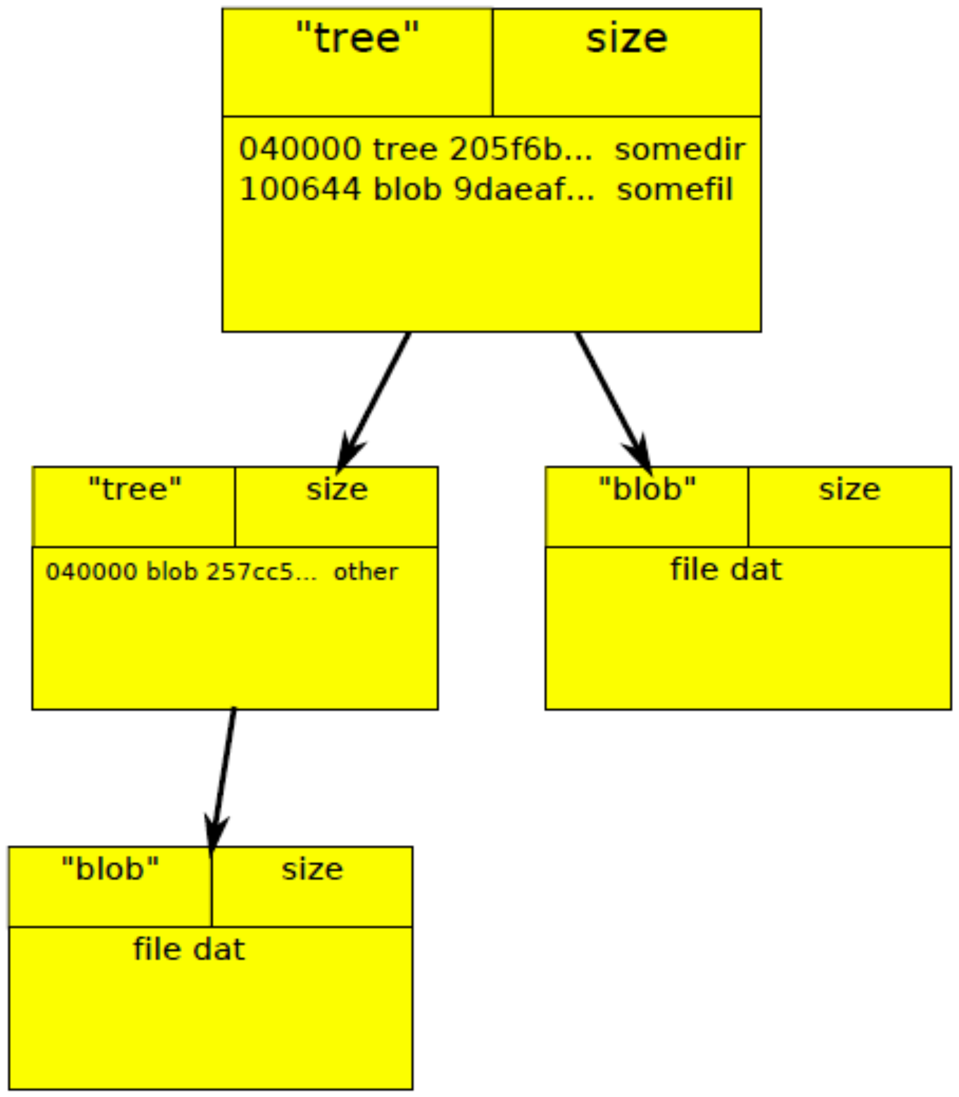
Trees


"tree"
size
040000 tree 205f6b... somedir
100644 blob 9&aeaf... somefil
"tree"
size
040000 blob 257cc5... other

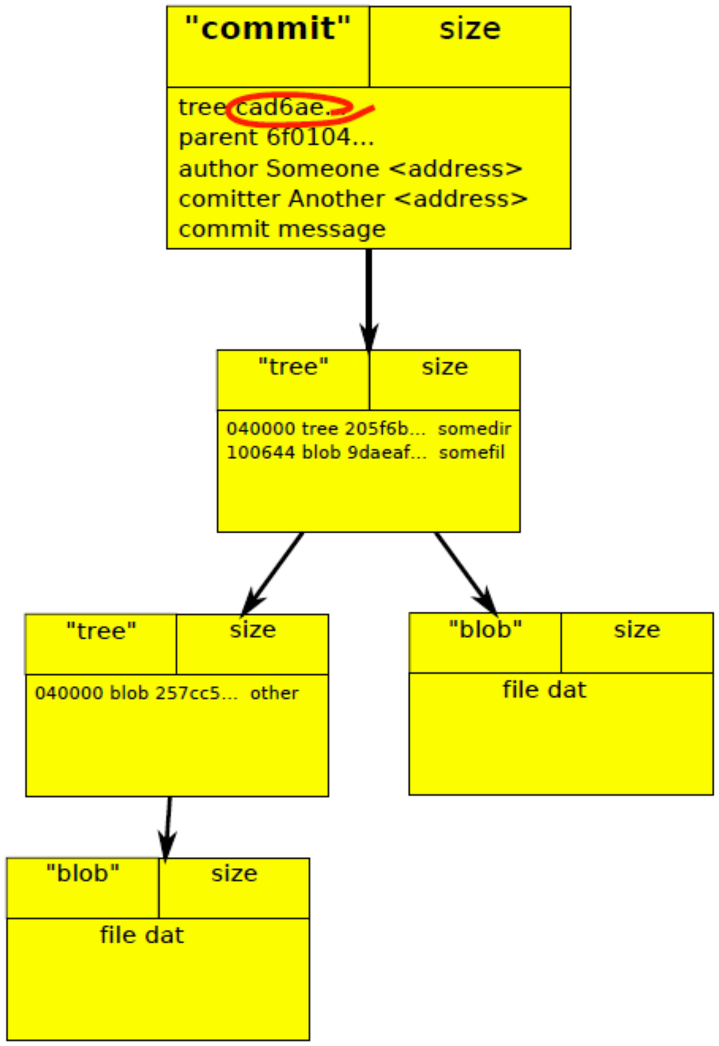
Trees


"tree"
size
040000 tree 205f6b... somedir
100644 blob 9daeaf... somefil
"tree"
size
040000 blob 257cc5... other
"blob"
size
file dat

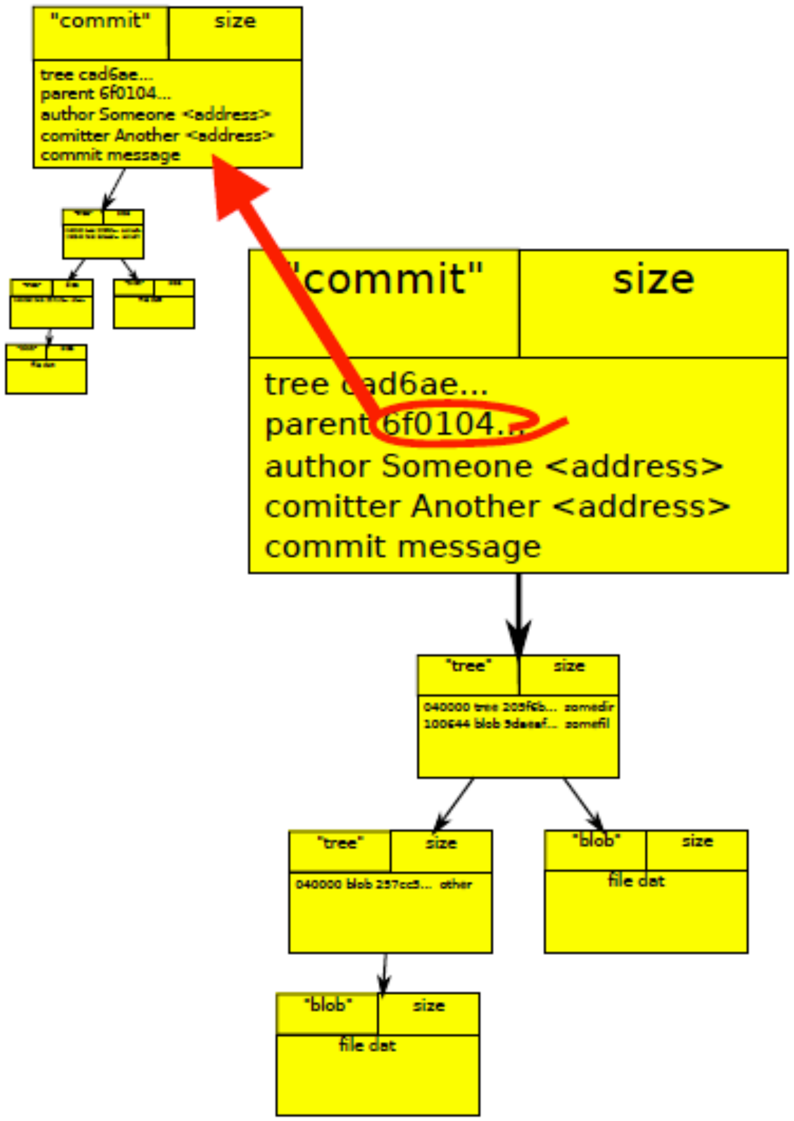
Trees


"tree"
size
040000 tree 205f6b... somedir
100644 blob 9daeaf... somefil
"tree"
size
040000 blob 257cc5... other
"blob"
size
file dat
"blob"
size
file dat

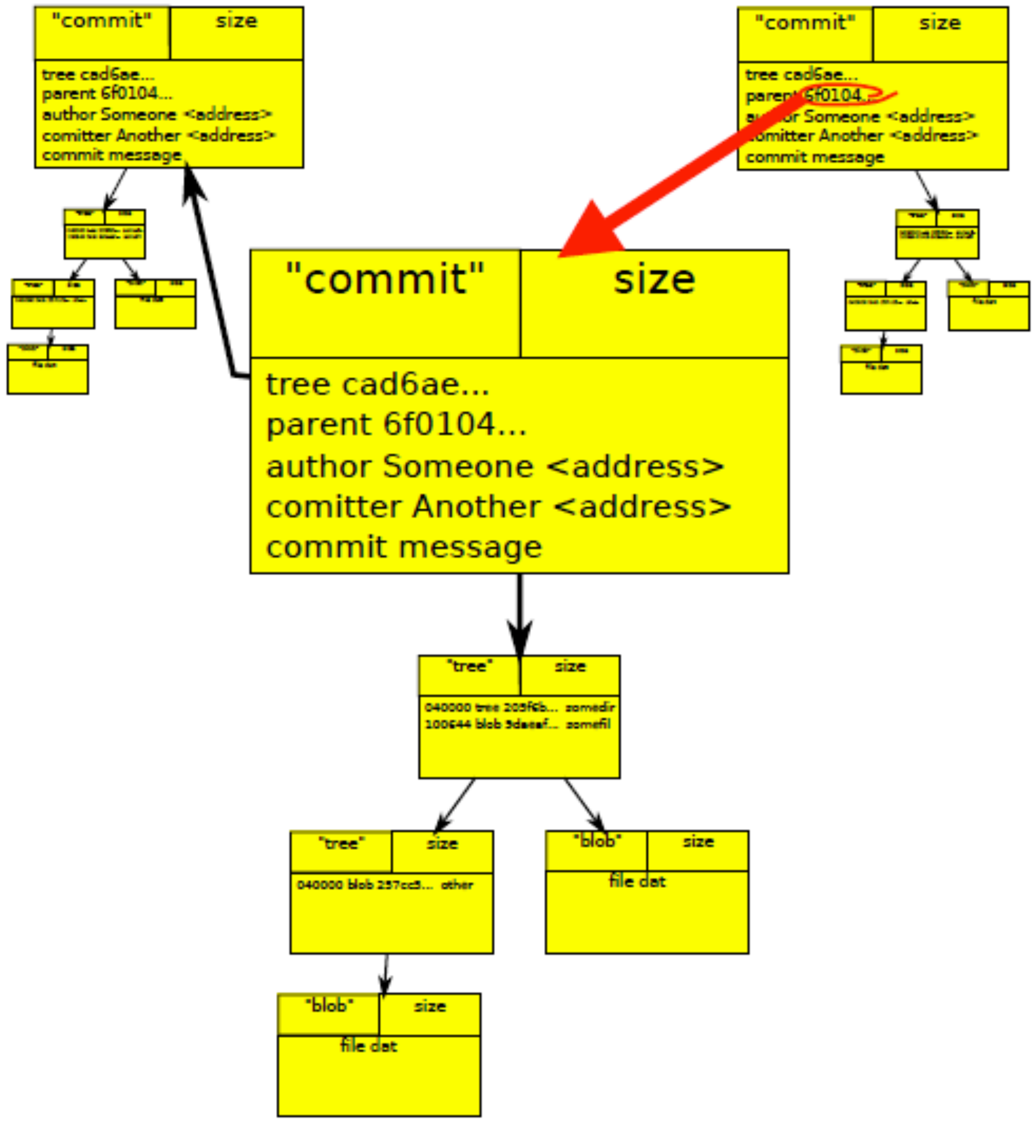
Commits


"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"tree"
size
040000 tree 205f6b... somedir
100644 blob 9daeaf... somefil
"tree"
size
040000 blob 257cc5... other
"blob"
size
file dat
"blob"
size
file dat

Commits

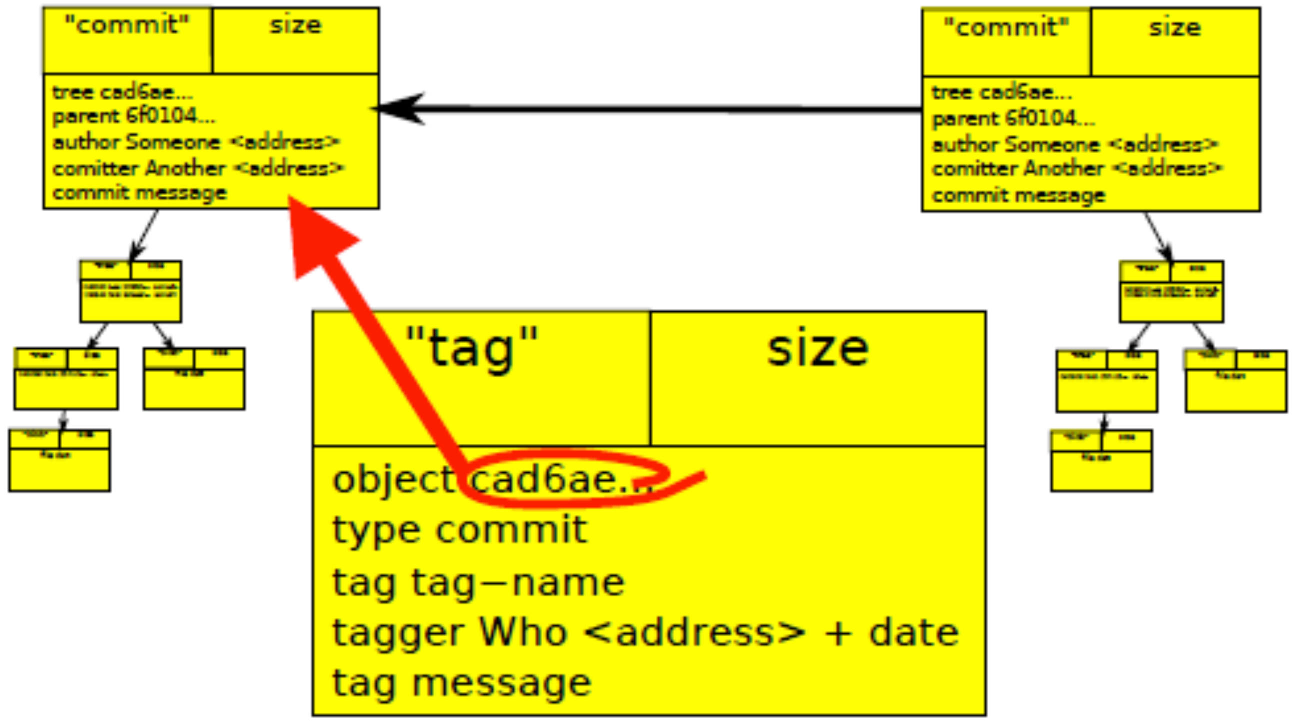

"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"tree"
size
040000 tree 205f6b... somedir
100644 blob 5dacaf... somefil
"tree"
size
040000 blob 257cc5... other
"blob"
size
file dat
"blob"
size
file dat

Commits

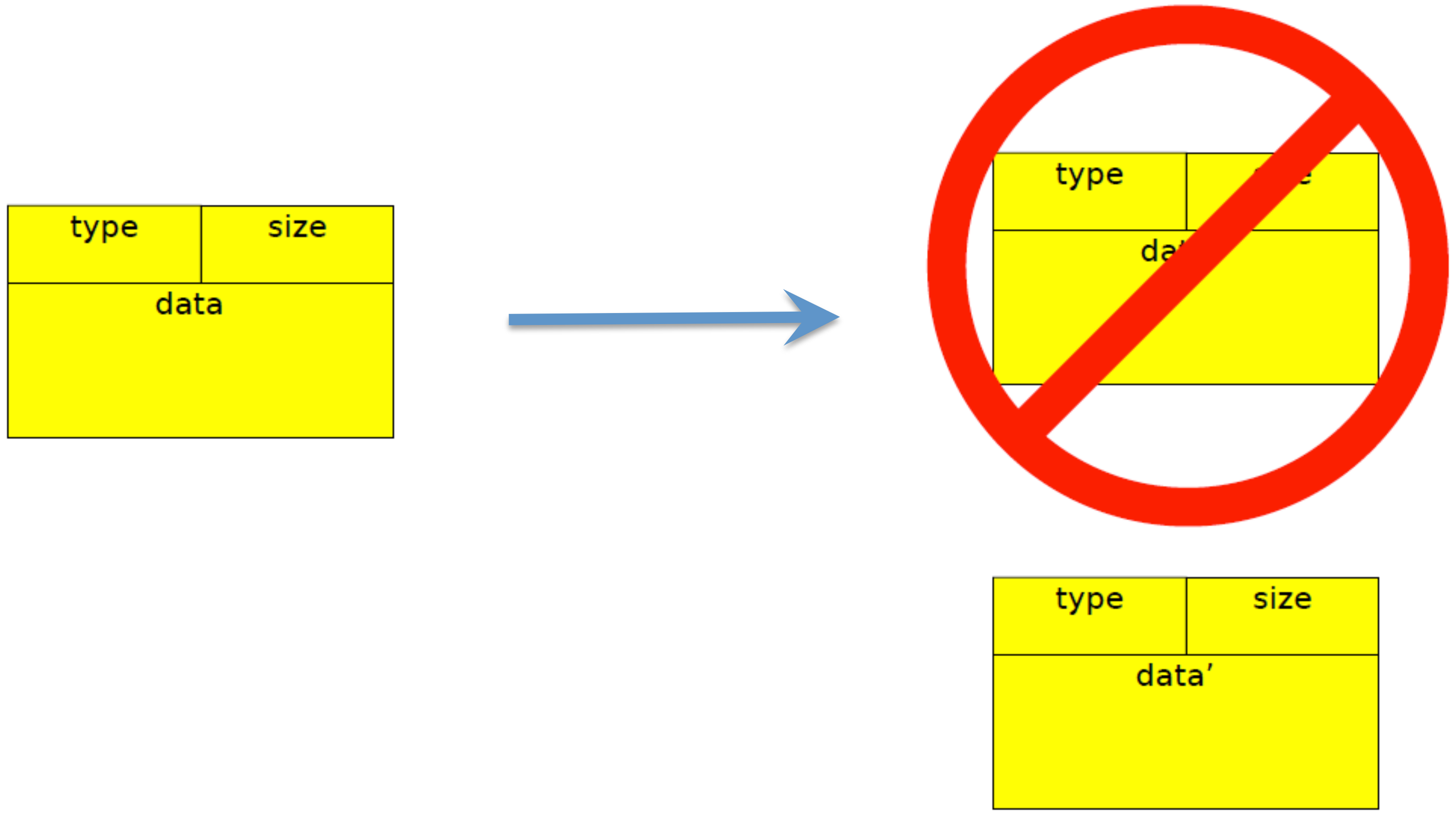


"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"tree"
size
040000 tree 205f6b... somedir
100644 blob 3daeaf... somefil
"tree"
size
040000 blob 257cc5... other
"blob"
size
file dat

Commits


"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"commit"
size
tree cad6ae...
parent 6f0104...
author Someone <address>
comitter Another <address>
commit message
"tag"
size
object cad6ae...
type commit
tag tag−name
tagger Who <address> + date
tag message

Objects are immutable



Basic command format

```
git <options> <command> <options>
```

Online help

- list of common commands

```
git help
```

- Brief per-command help

```
git command –h
```

- `man pages`

```
man git-<command>
git help <command>
git <command> --help
```

Configuration

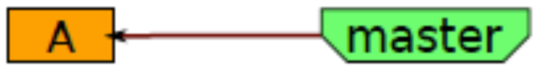
```
$HOME/.gitconfig

git config –global user.name “Ted Baker”
git config –global user.email baker@cs.fsu.edu
git config –global color.pager true
git config –global cour.ui auto
```

A typical developer story

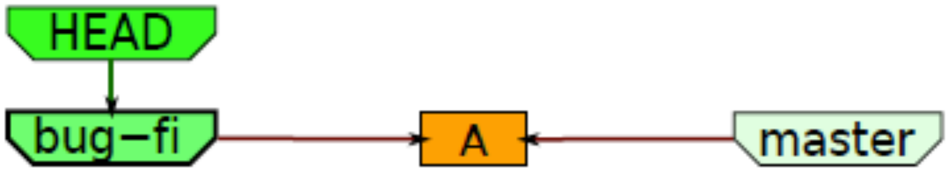
Showing how various commands are used, in context.

Working on branches

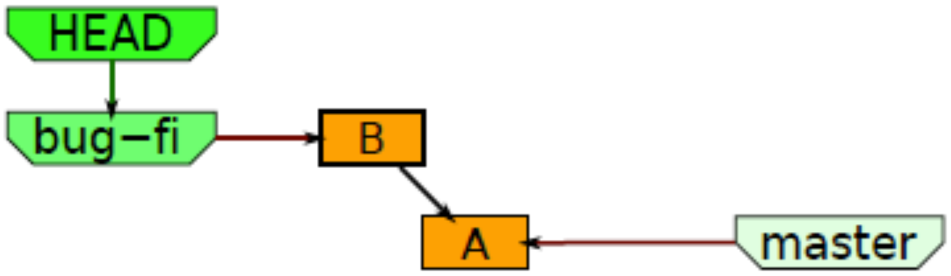
Start with some tree



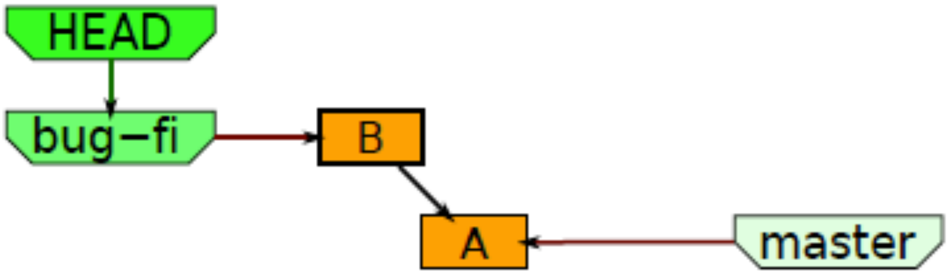
```
git checkout –b bug-fix
```



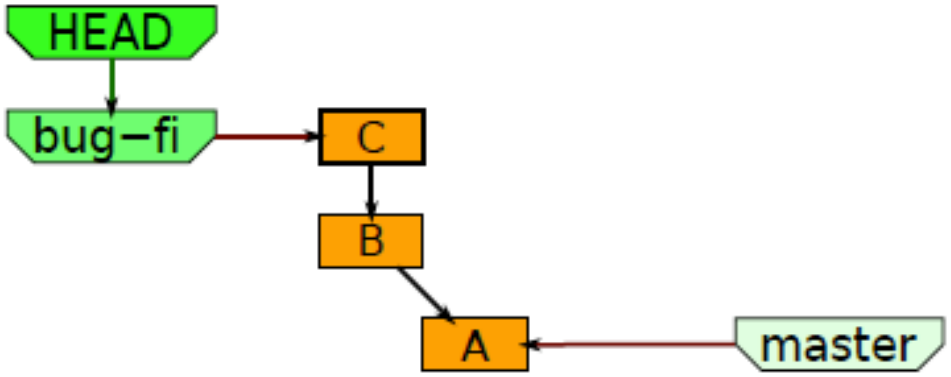
```
git commit –a –m"B"
```



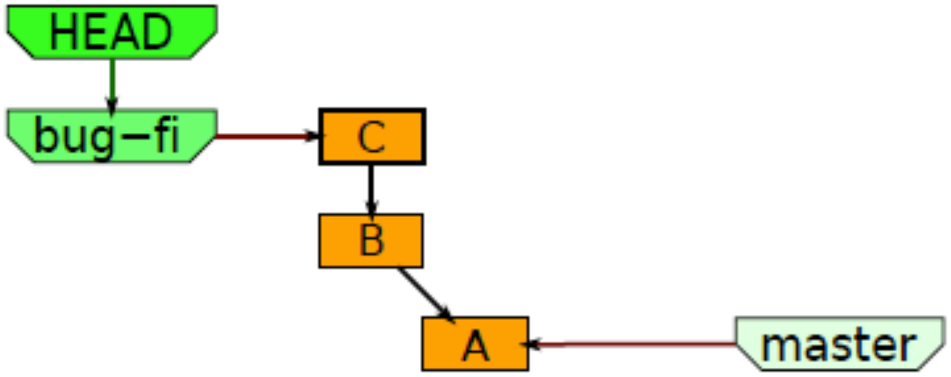
Continue making changes



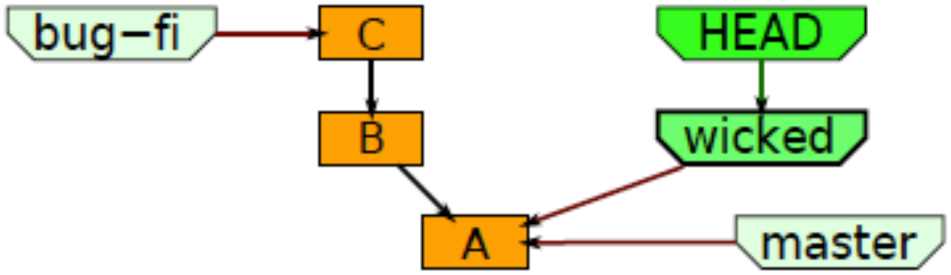
```
git commit –a –m"C" –b
```



Decide to try out a “wicked” alternate idea.

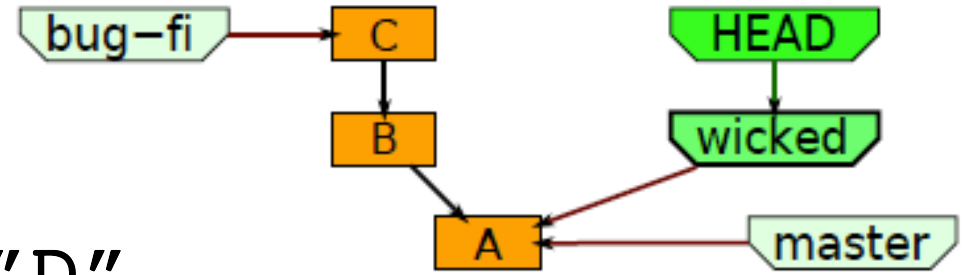


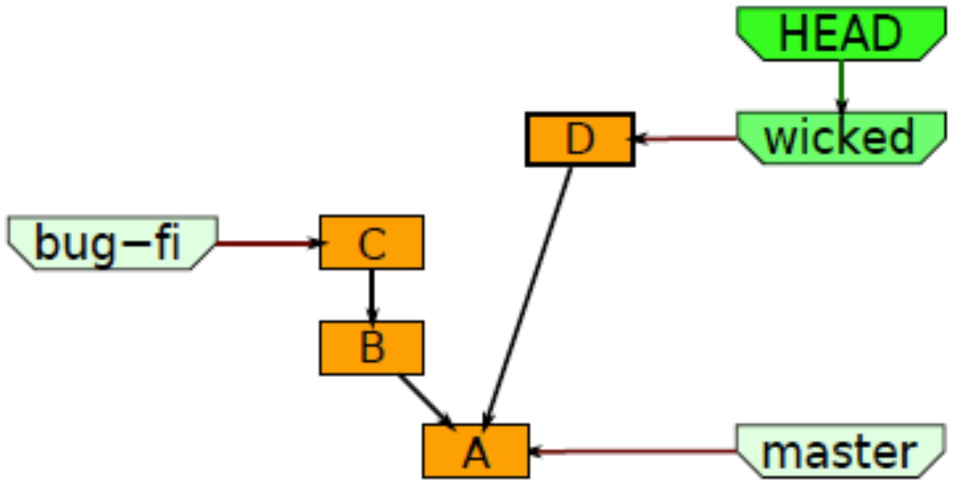
```
git checkout –b wicked master
```



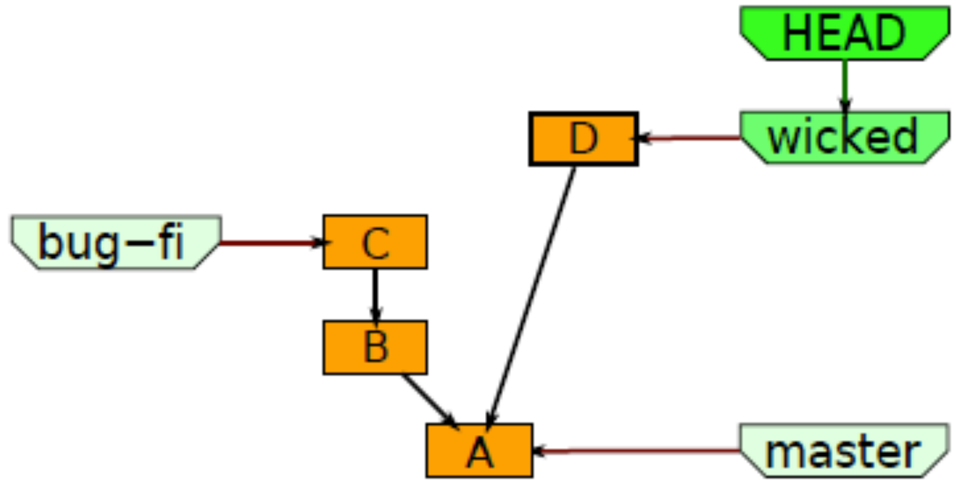
Do some work on this alternate branch.

```
git commit –a –m"D"
```

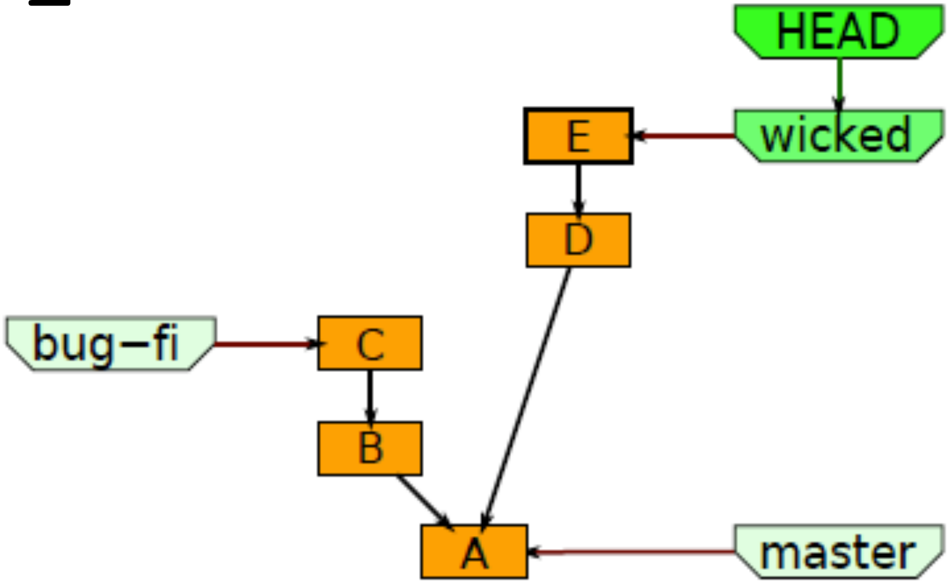




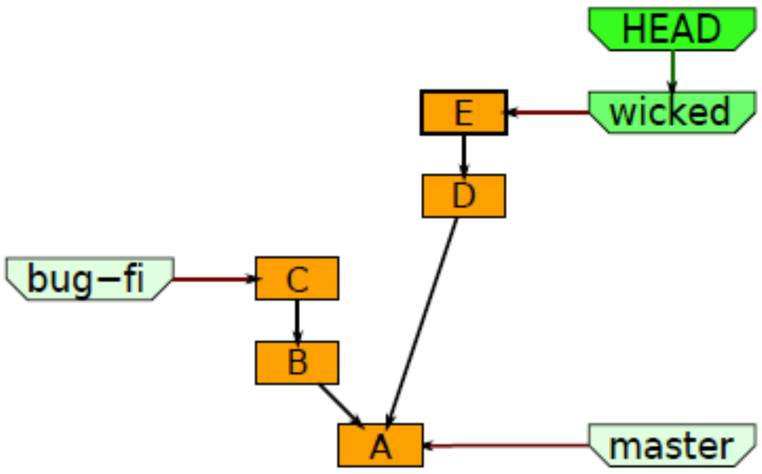
And some more work.



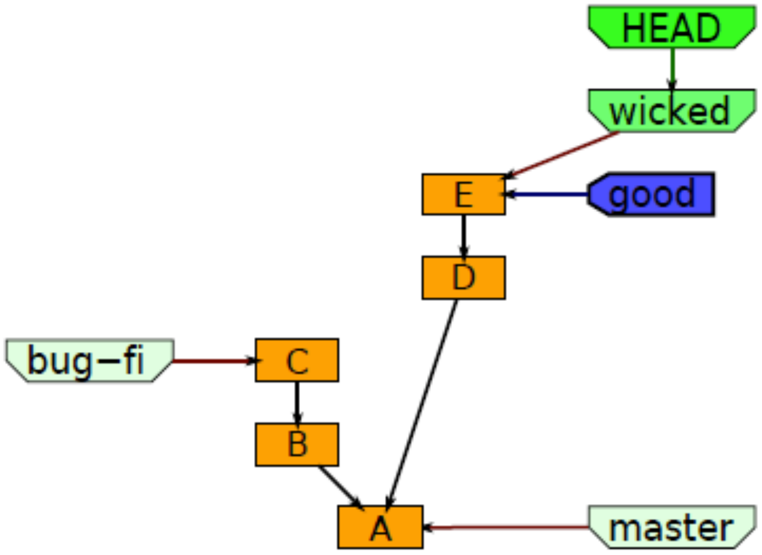
```
git commit –a –m"E"
```



You have gotten to a good point.

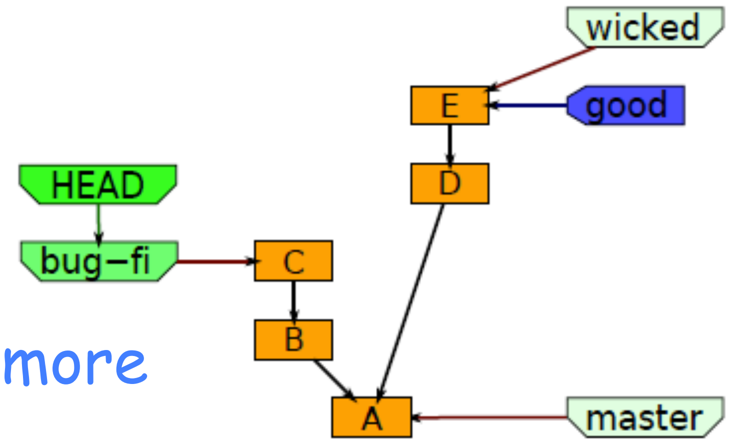


```
git tag –a –m"got somewhere" good
```



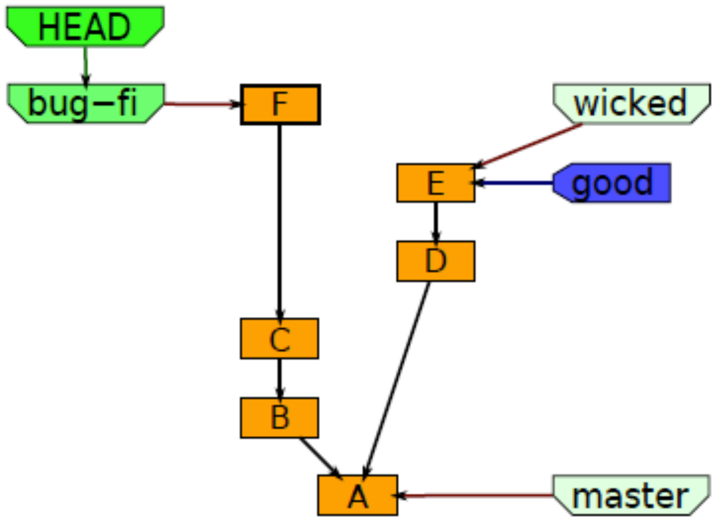
Manager asks about the bug

```
git checkout bug-fix
```



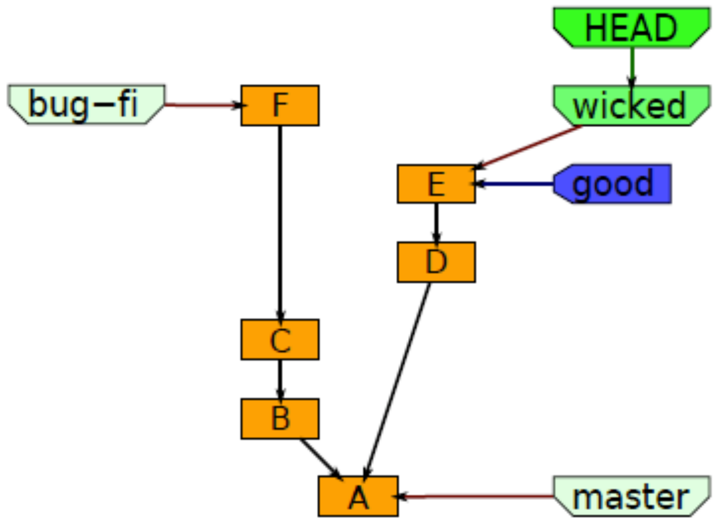
So you go back to work on it some more

```
git commit –a –m “F”
```



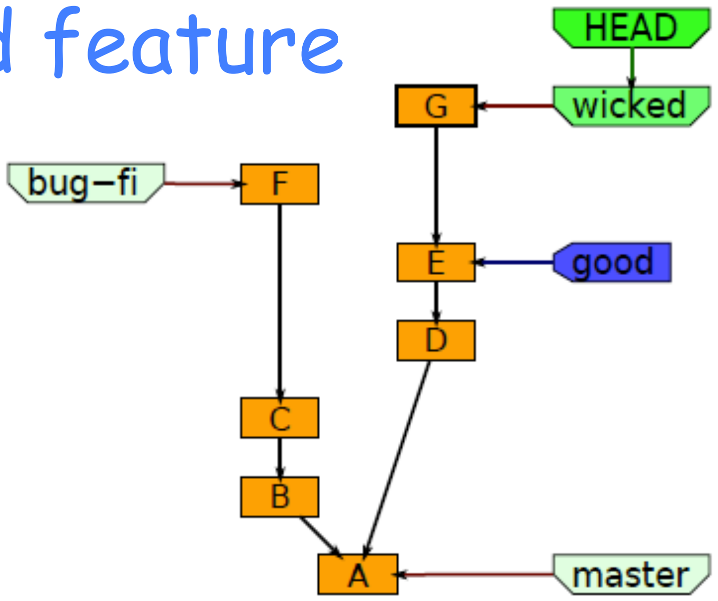
But your mind is elsewhere

```
git checkout wicked
```

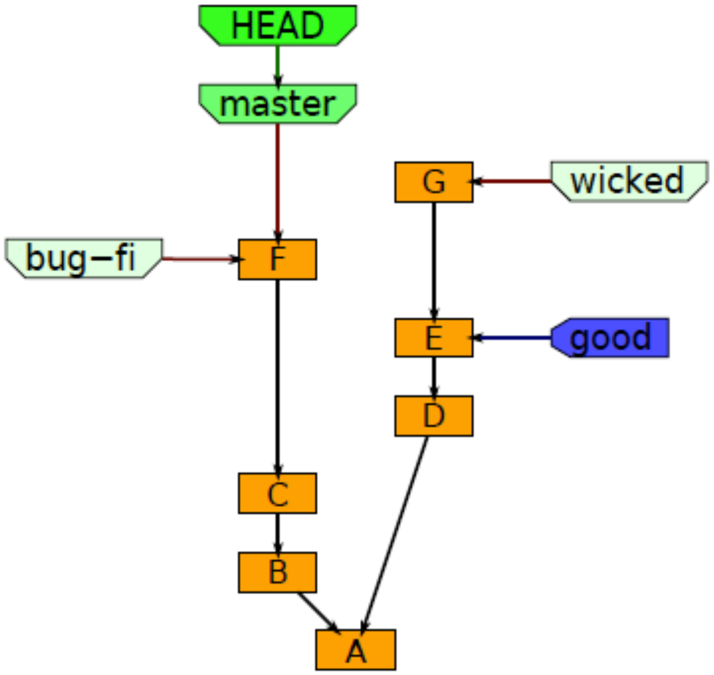


so you finish off the wicked feature

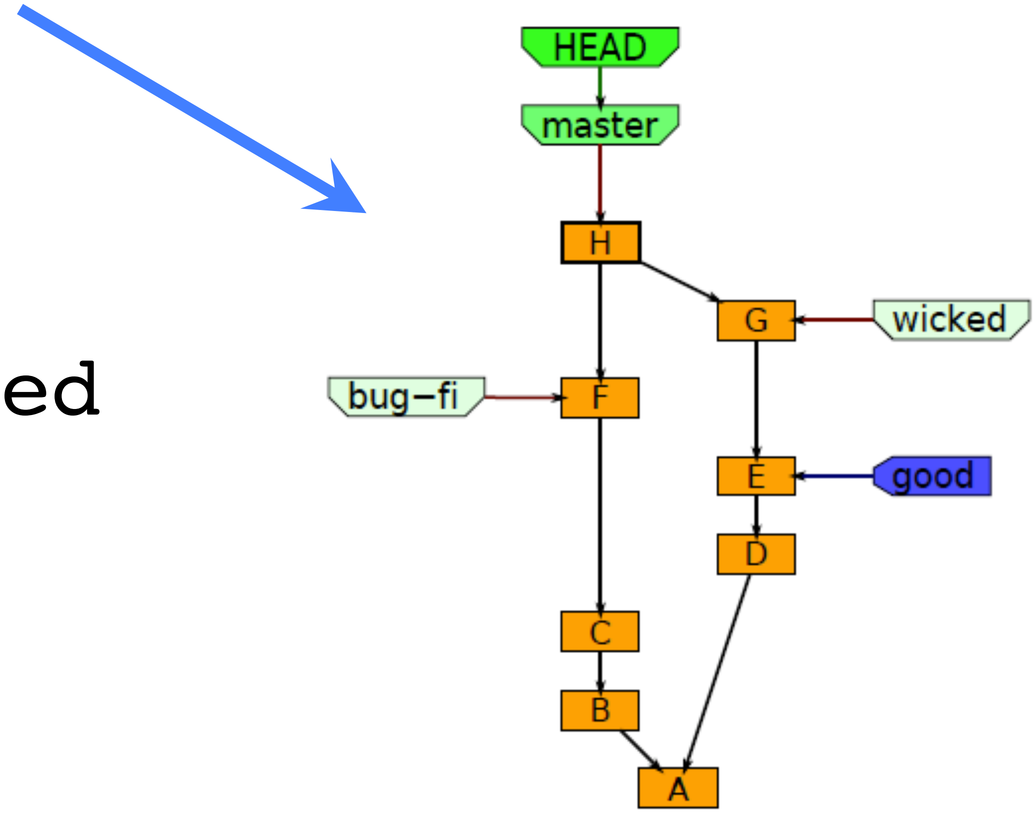
```
git commit –a –m"G"
```

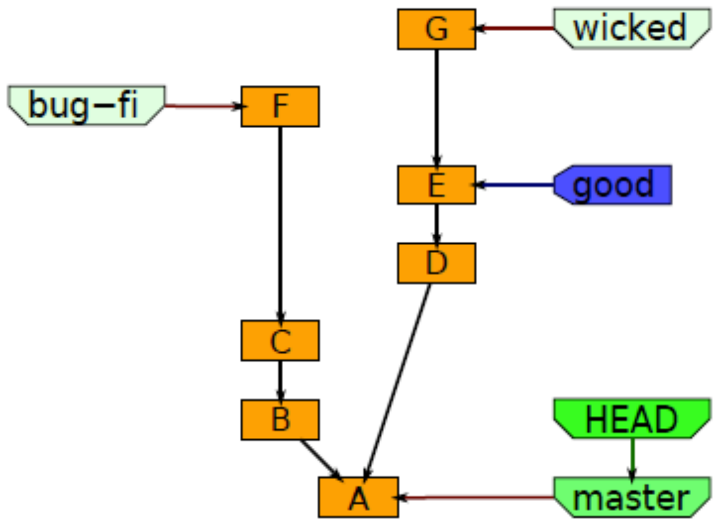


Then merge in the new feature.



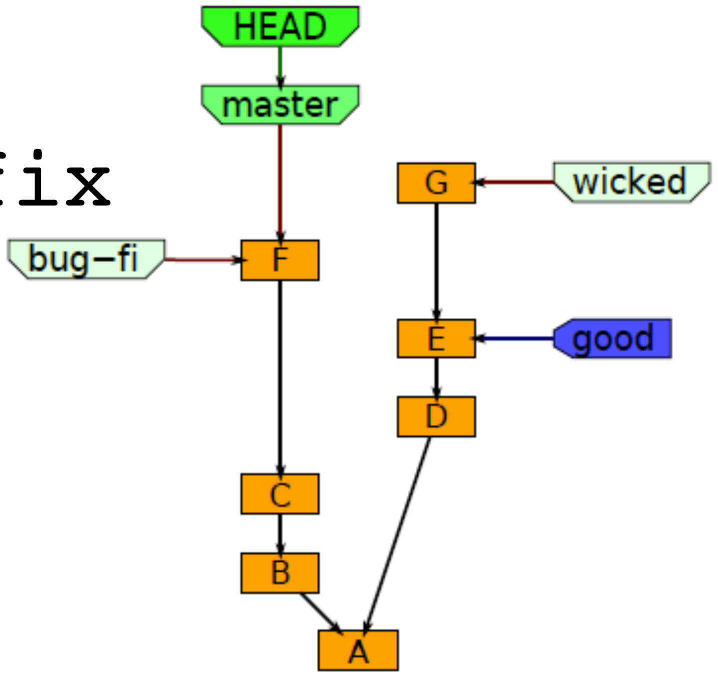
```
git merge wicked
```

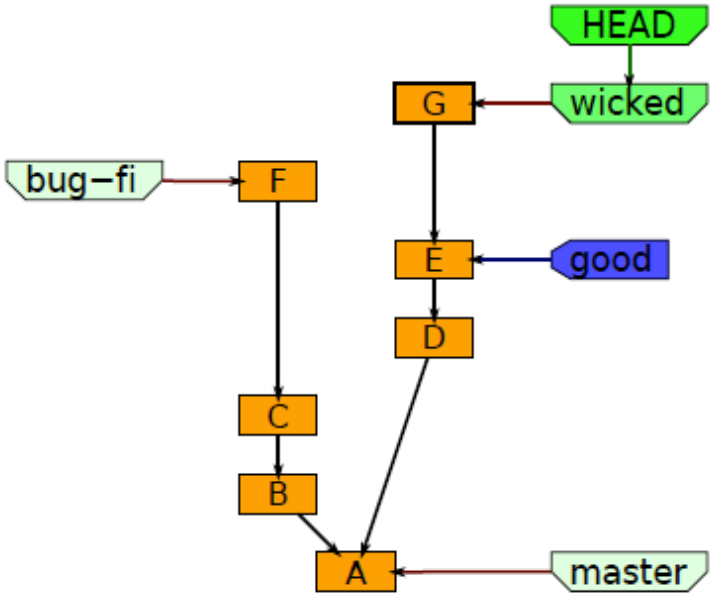




First advance the the master to include the bug fix.

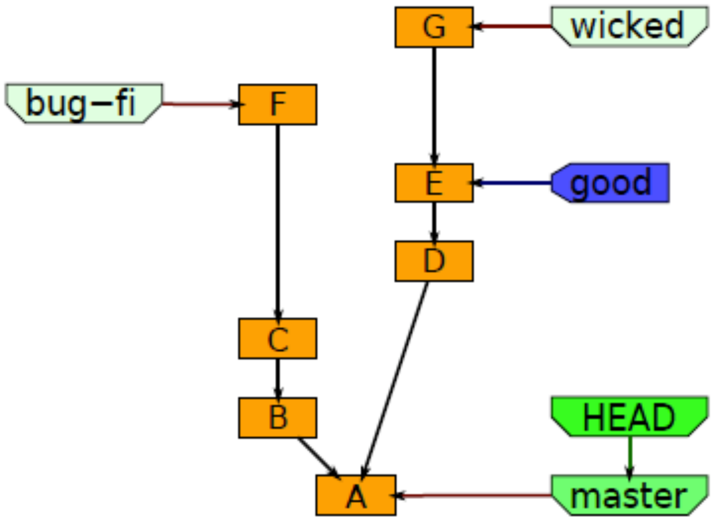
```
git reset --hard bug-fix
```





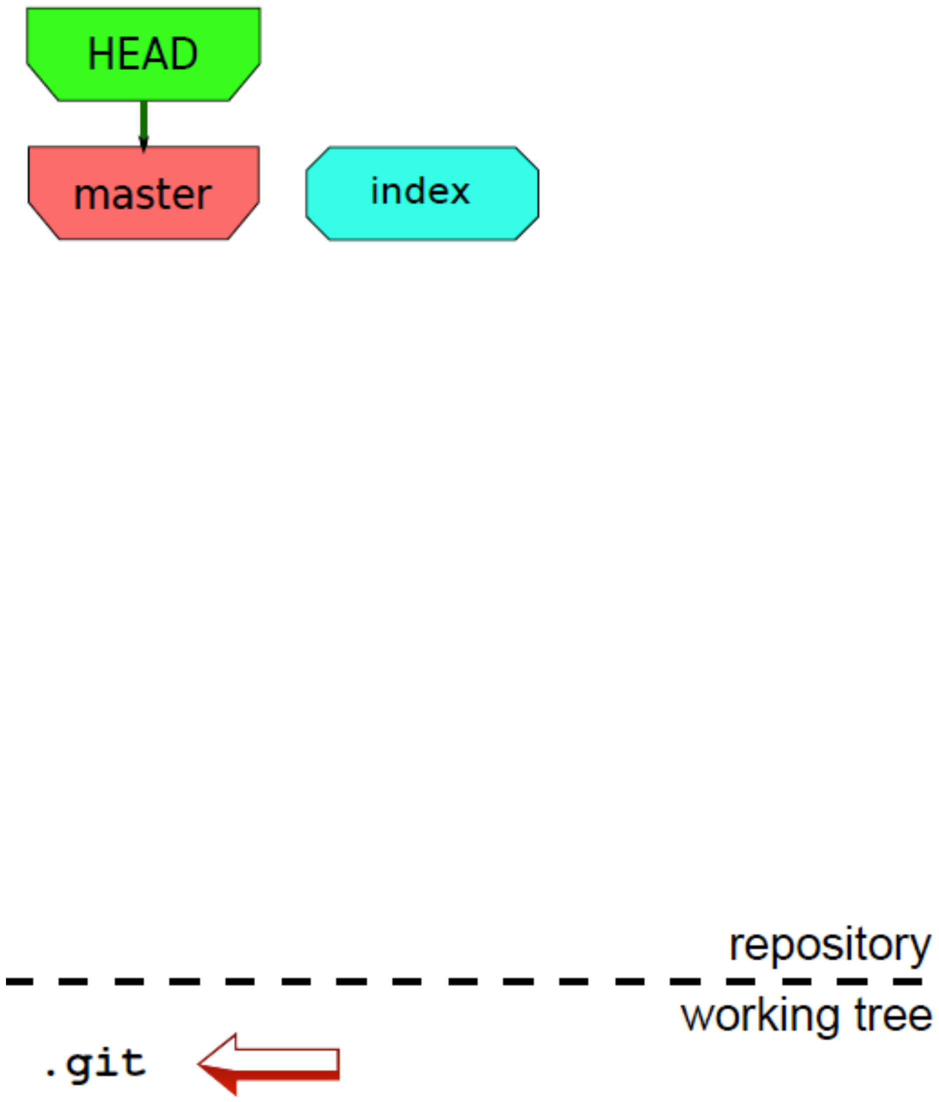
```
git checkout master
```

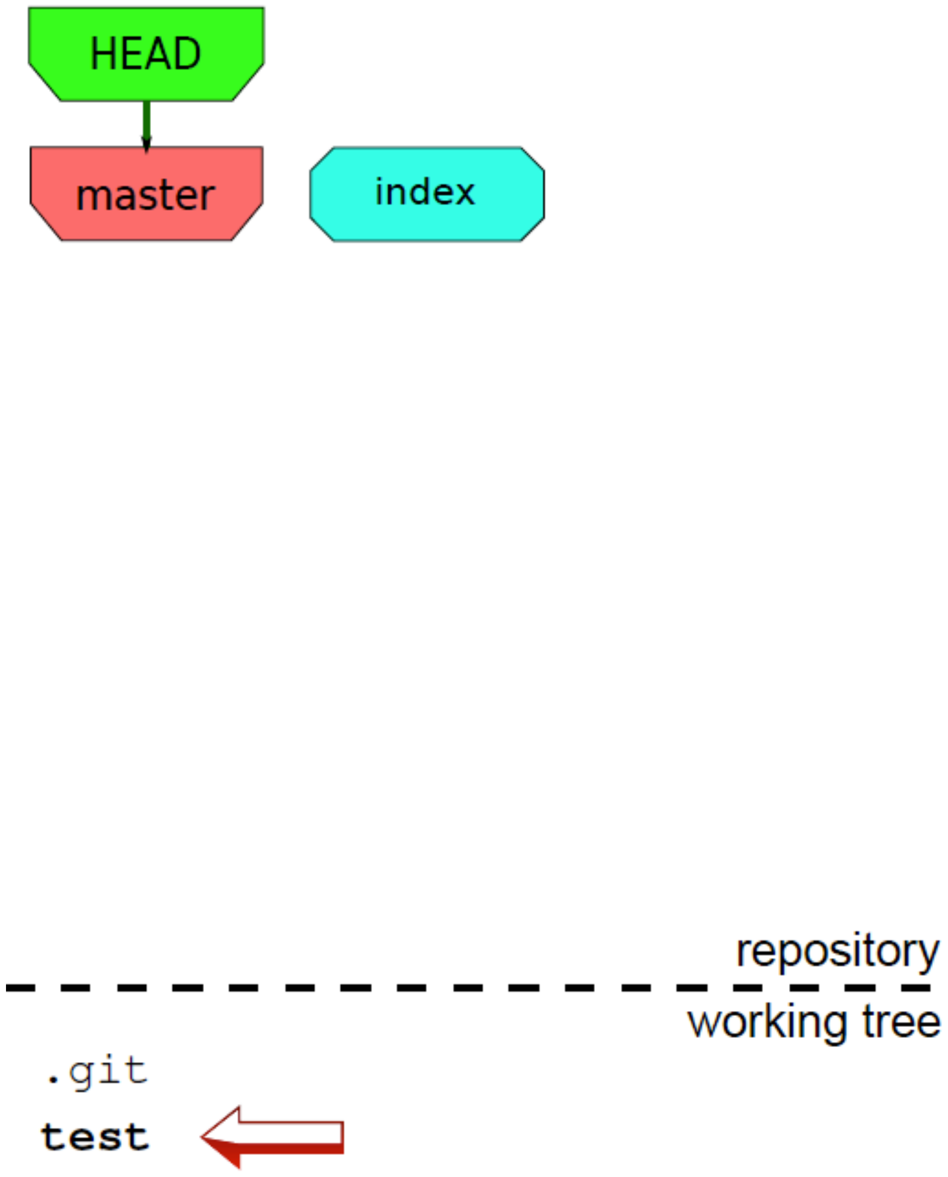
Bug fix and wicked new feature are both done, so it's time to merge.

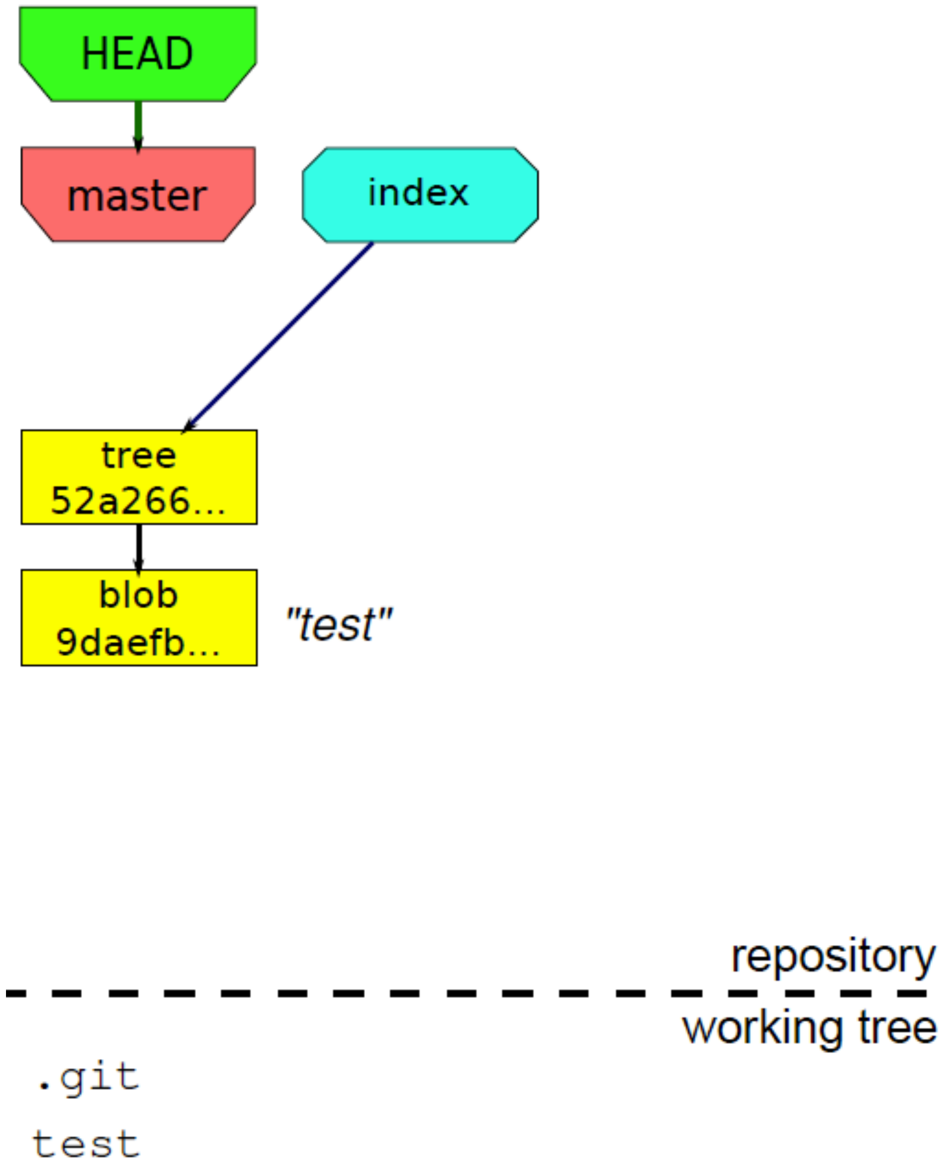


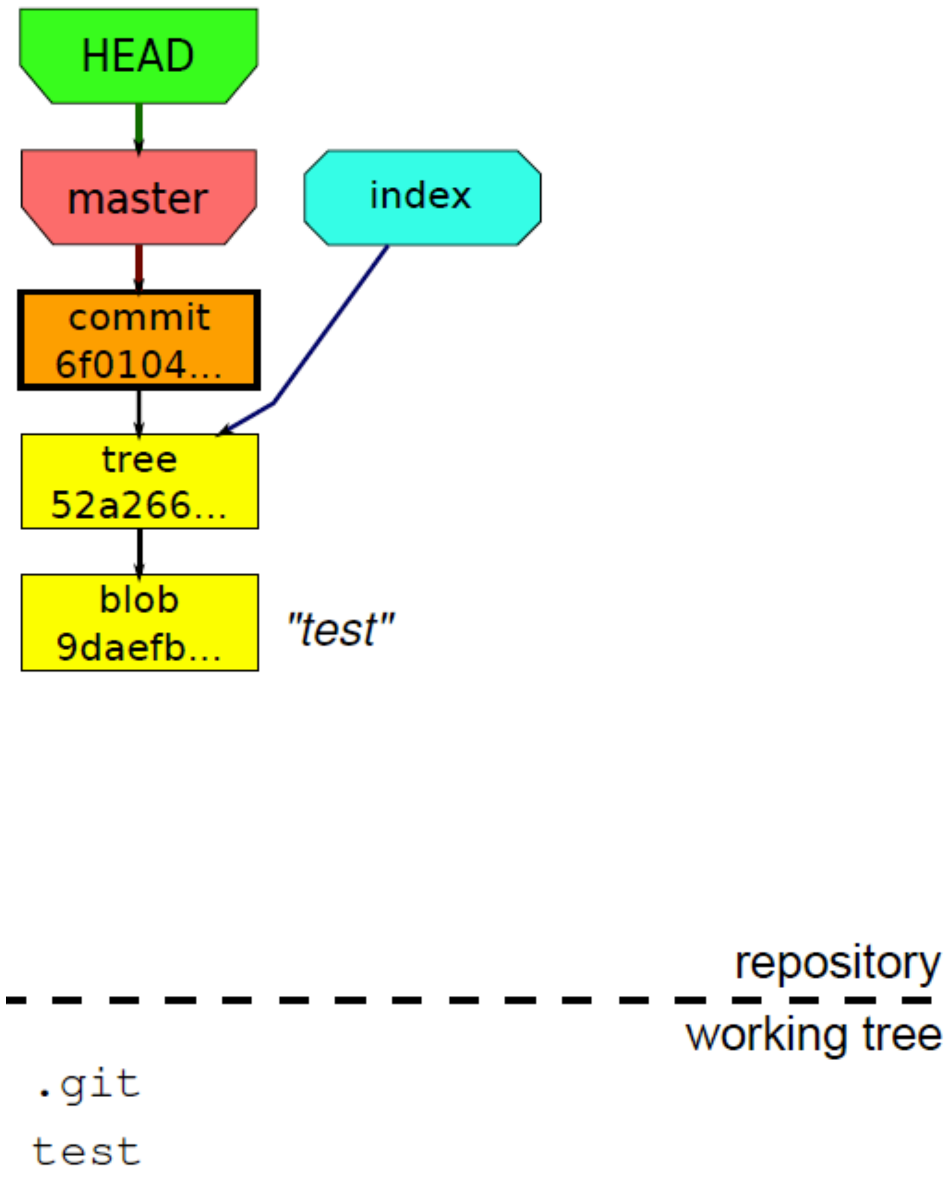
Another story

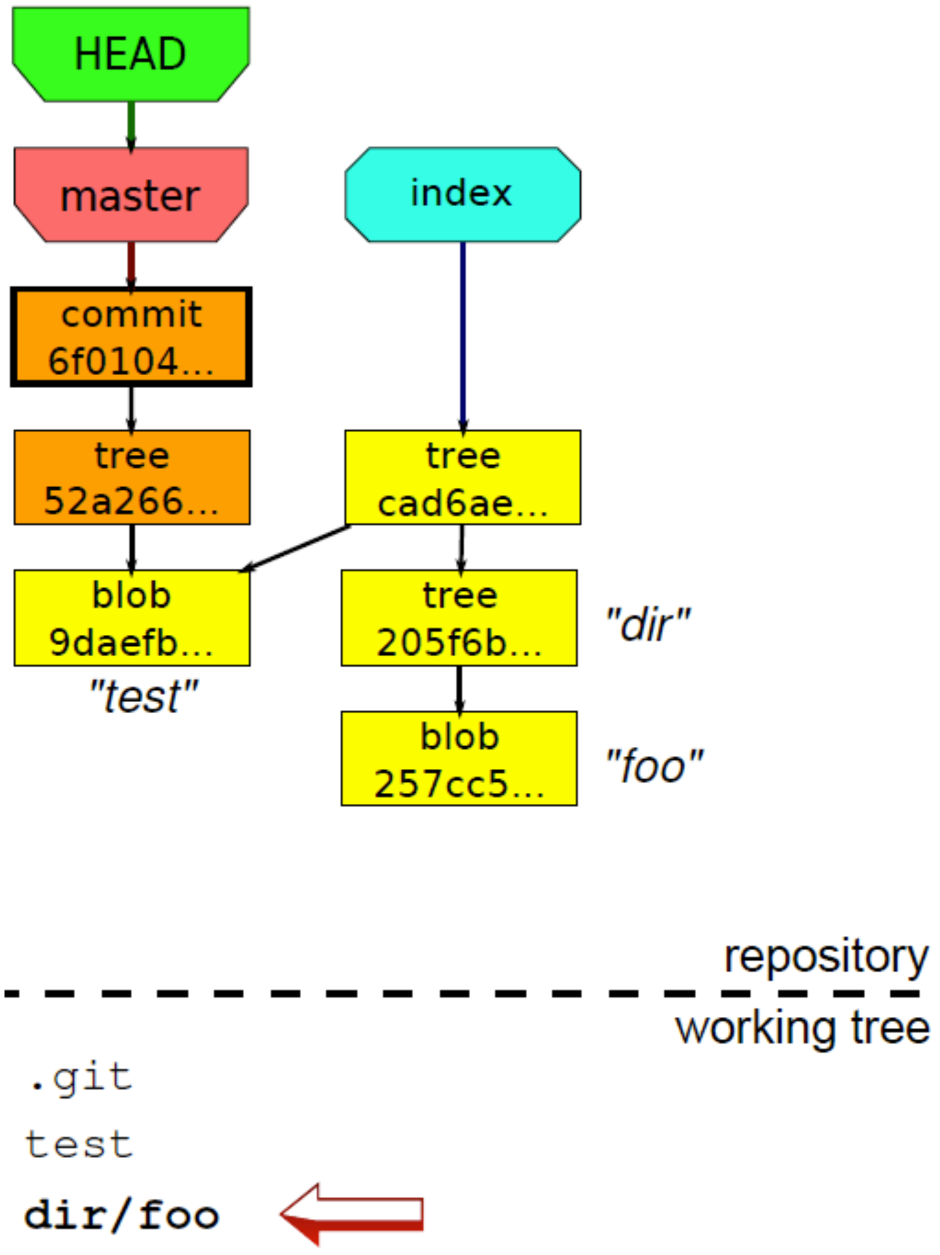
This set of slides has not yet been completely transcribed from the original web tutorial.

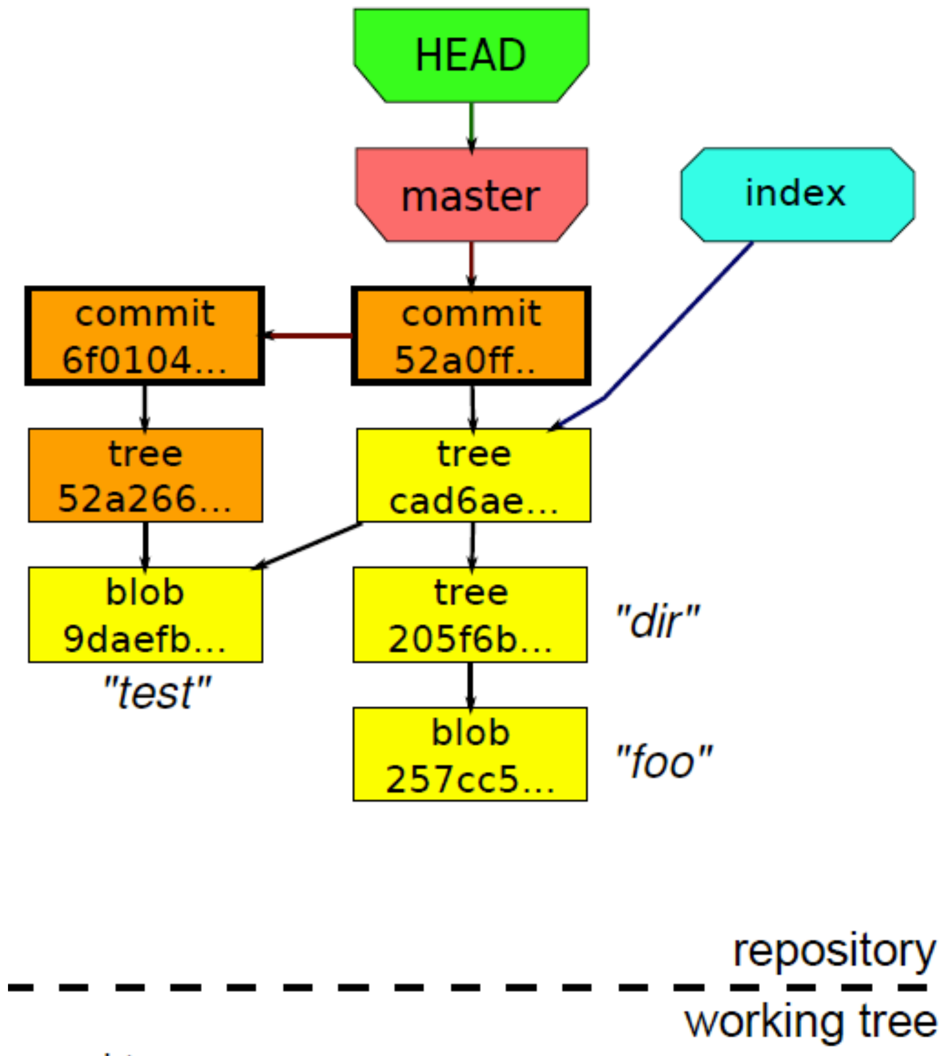

HEAD
master
index
repository
working tree
.git


HEAD
master
index
repository
working tree
.git
test

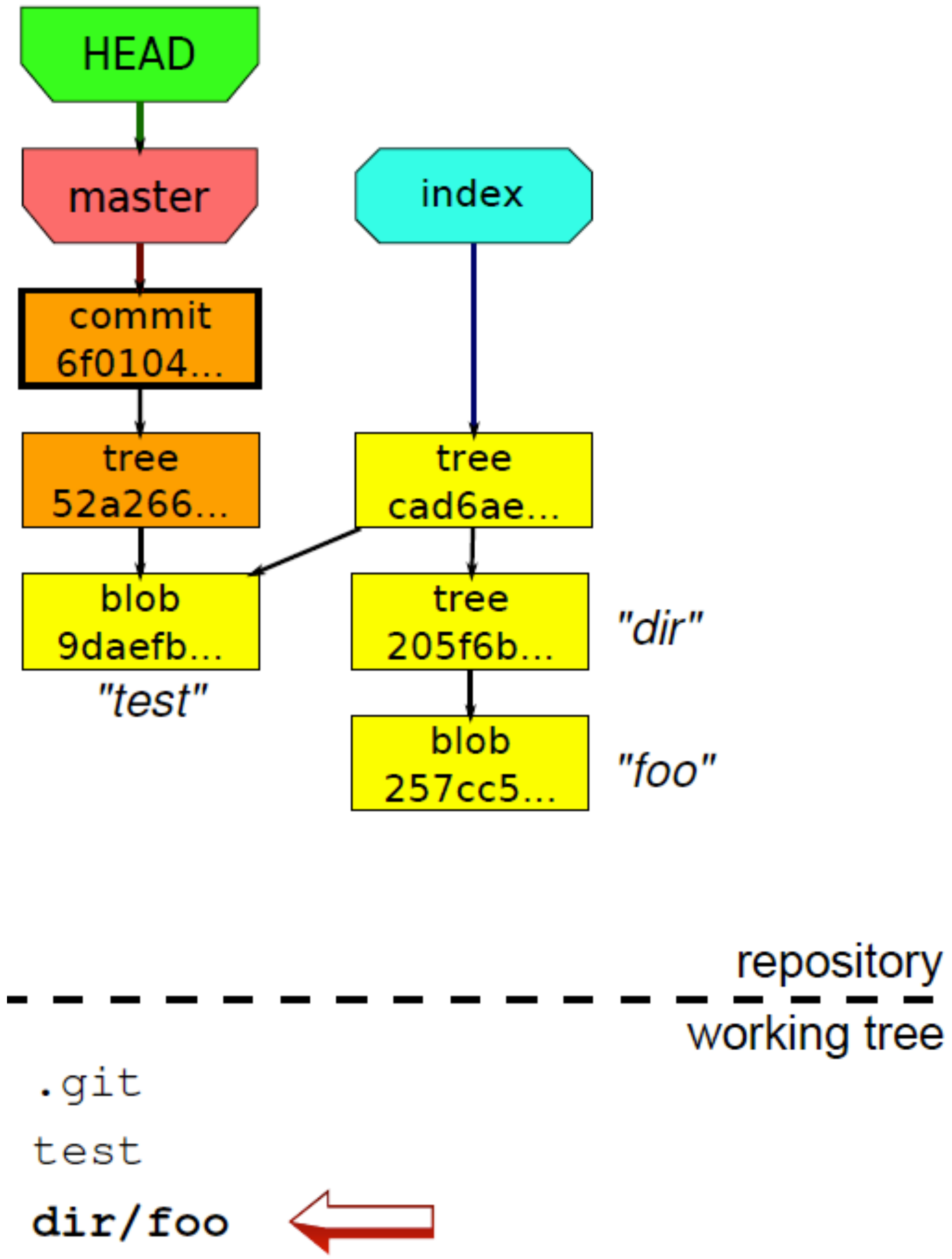

HEAD
master
index
tree
52a266...
blob
9daefb...
"test"
repository
working tree
.git
test

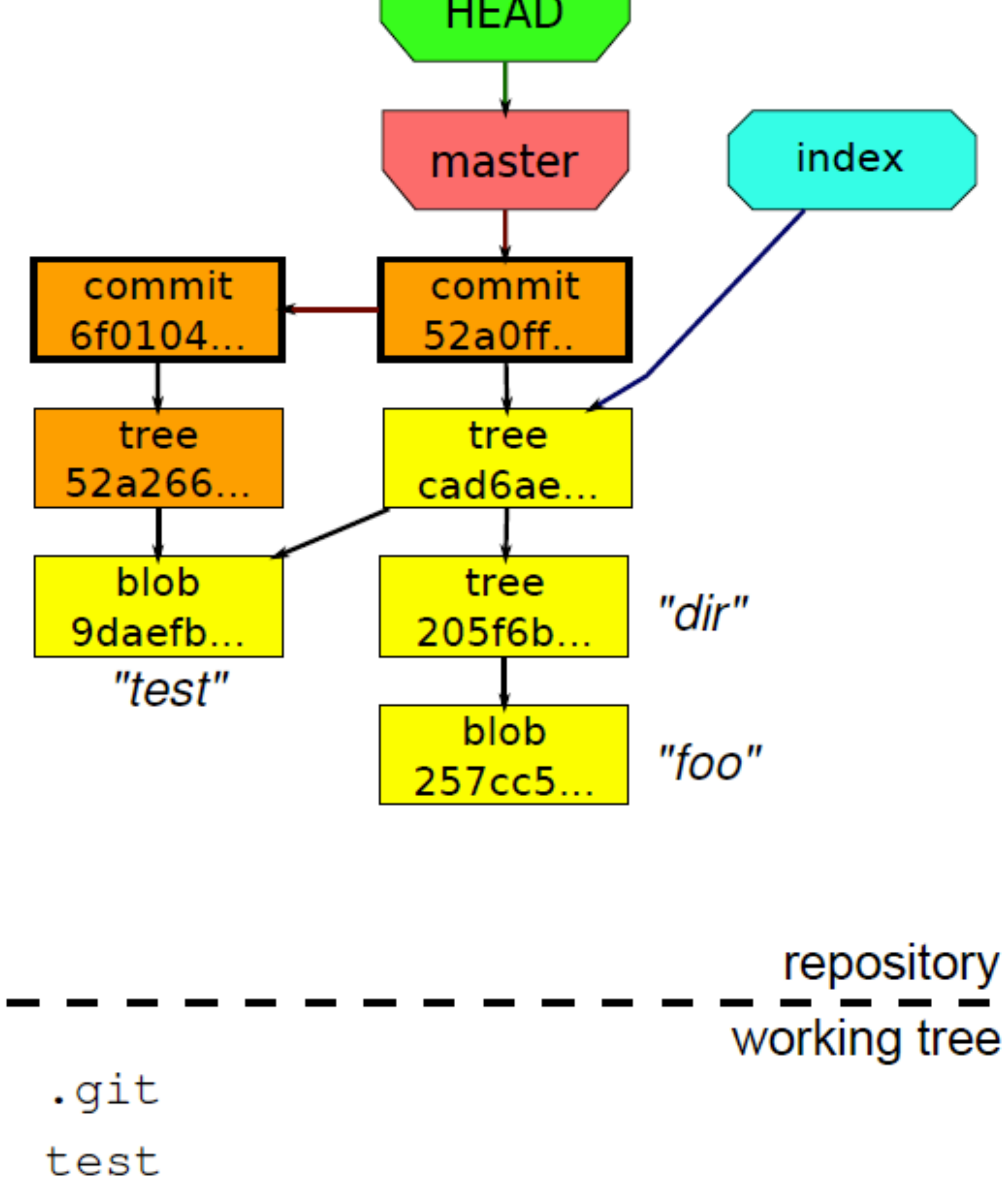

HEAD
master
index
commit
6f0104...
tree
52a266...
blob
9daefb...
"test"
repository
working tree
.git
test


HEAD
master
index
commit 6f0104...
tree 52a266...
tree cad6ae...
blob 9daefb...
"test"
tree 205f6b...
"dir"
blob 257cc5...
"foo"
repository
working tree
.git
test
dir/foo



```
.git
test
dir/foo
```

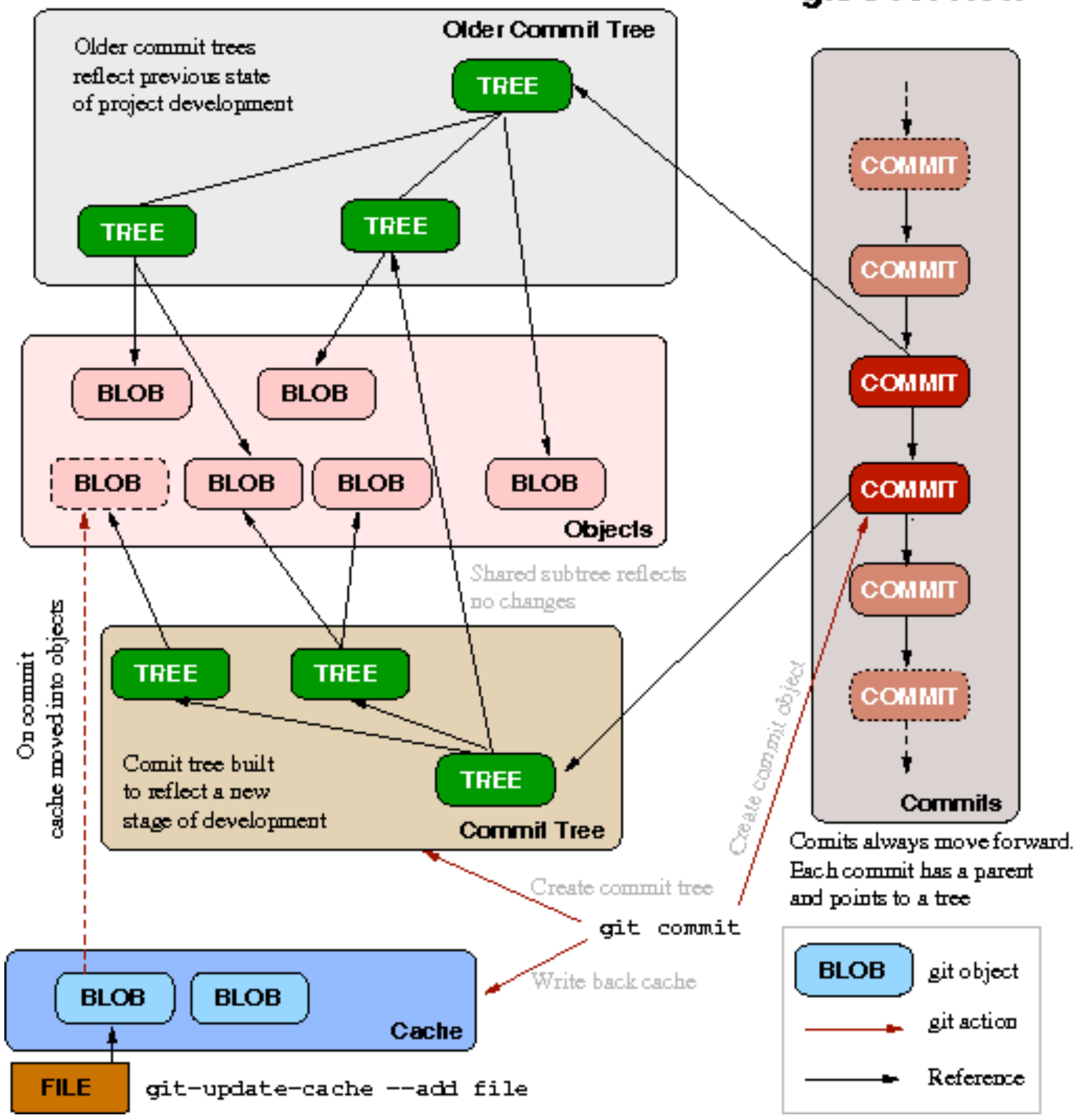

HEAD
master
index
commit 6f0104...
tree 52a266...
tree cad6ae...
blob 9daefb...
"test"
tree 205f6b...
"dir"
blob 257cc5...
"foo"
repository
working tree
.git
test
dir/foo

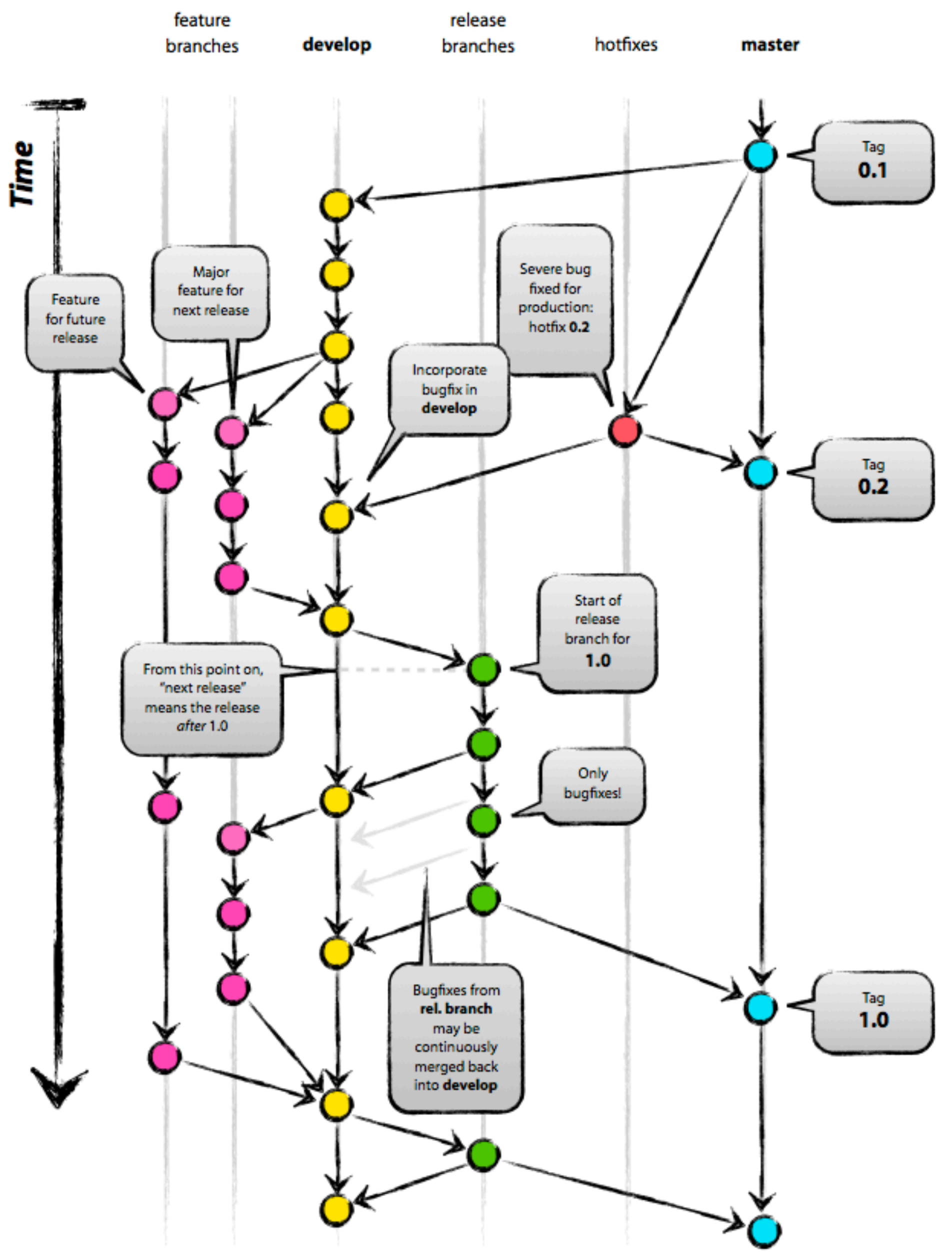

HEAD
master
index
commit 6f0104...
commit 52a0ff..
tree 52a266...
tree cad6ae...
blob 9daefb...
"test"
tree 205f6b...
"dir"
blob 257cc5...
"foo"
repository
working tree
.git
test

Nice diagrams

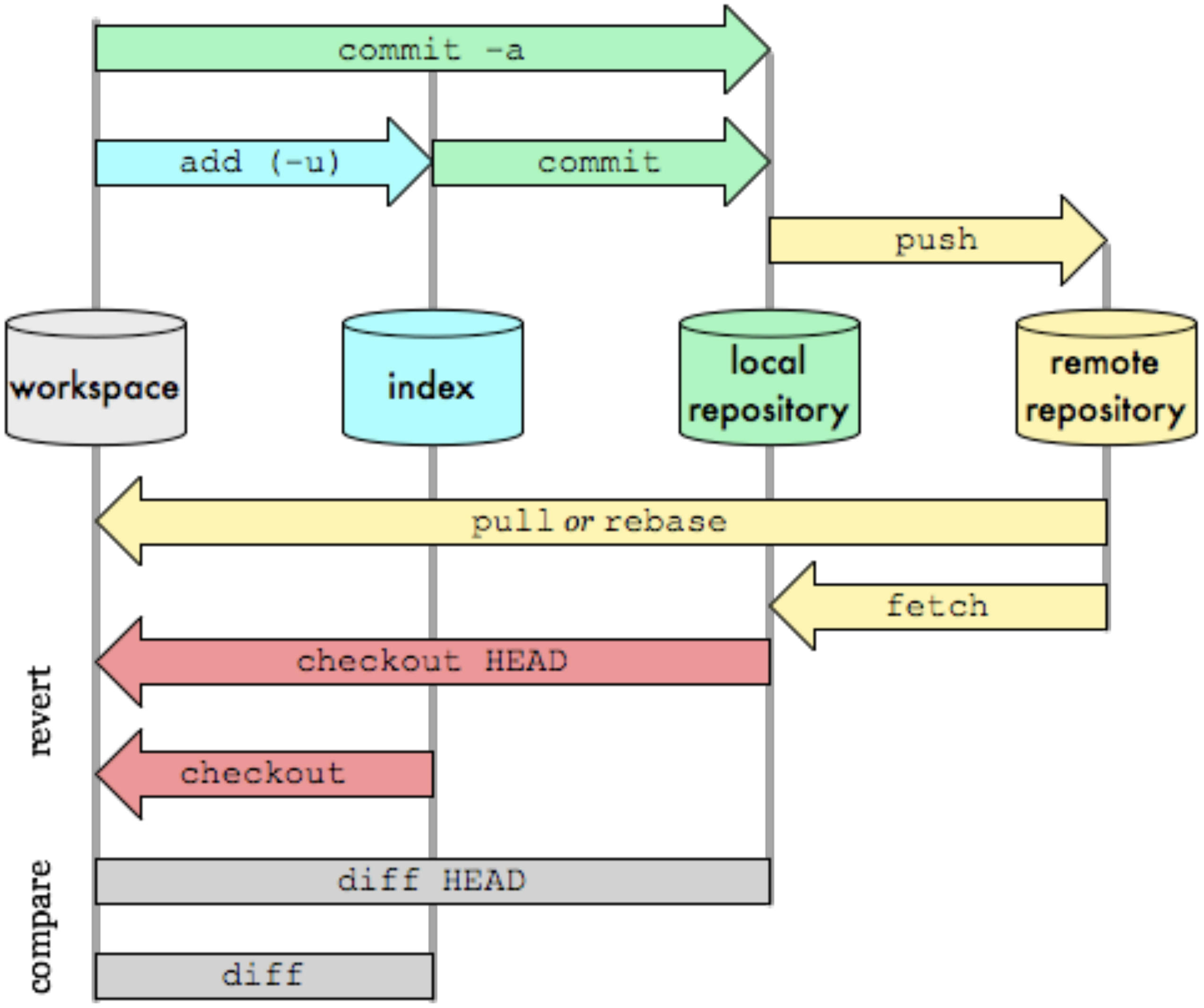
Some helpful diagrams collected from the Web.

git overview

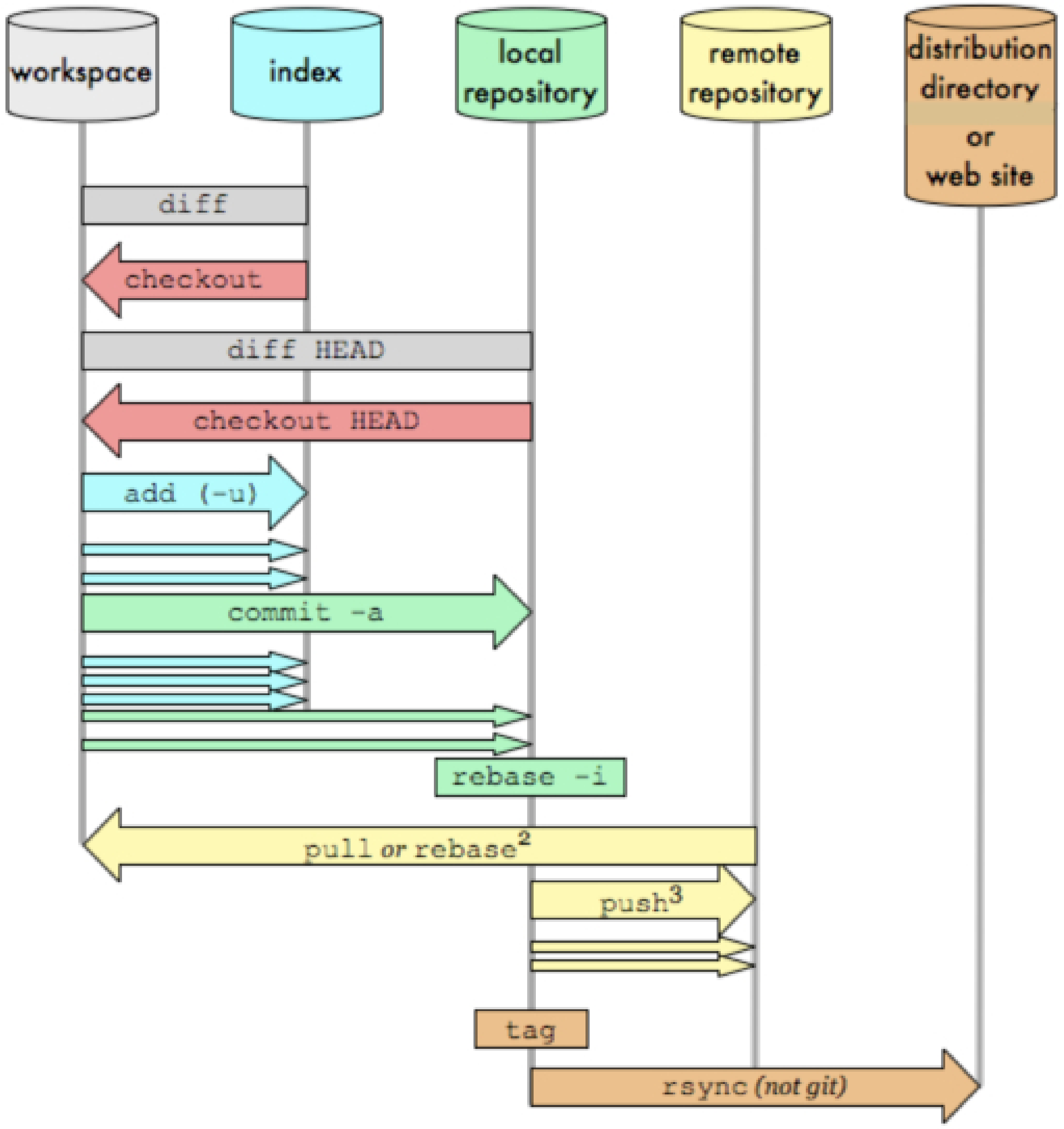

Older Commit Tree
Older commit trees reflect previous state of project development
TREE
TREE
TREE
BLOB
BLOB
BLOB
BLOB
BLOB
BLOB
Objects
Shared subtree reflects no changes
On commit cache moved into objects
TREE
TREE
Comit tree built to reflect a new stage of development
TREE
Commit Tree
Create commit tree
git commit
Write back cache
Create commit object
BLOB
BLOB
Cache
FILE
git-update-cache --add file
COMMIT
COMMIT
COMMIT
COMMIT
COMMIT
COMMIT
Commits
Comits always move forward. Each commit has a parent and points to a tree
BLOB
git object
git action
Reference


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Git transport commands



A Git workflow


workspace
index
local repository
remote repository
distribution directory or web site
diff
checkout
diff HEAD
checkout HEAD
add (-u)
commit -a
rebase -i
pull or rebase[2]
push[3]
tag
rsync (not git)