Introductory Version Control with Git

January 12, 2010

Slides by Jim Puls, jim@nondifferentiable.com

HACKER DOJO

Mountain View, CA

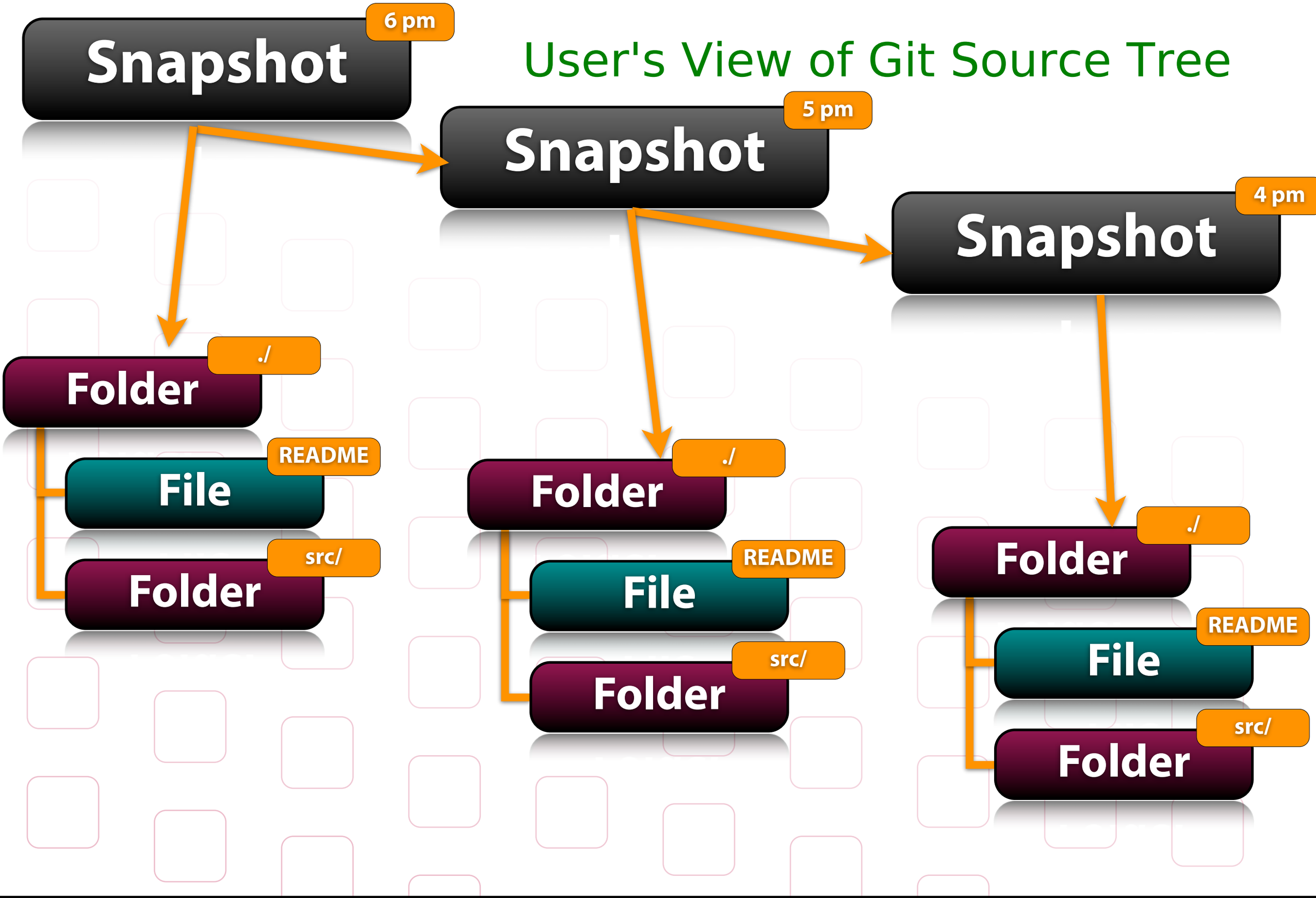
Before we start, let's install some software!

Major differences with svn, cvs, rcs:
-- git is project-based, rather than file-based;
-- git does not modify files;
-- "distributed" means that the user has a local repo over which he/she has complete control and which can be modified w/o a network connection.

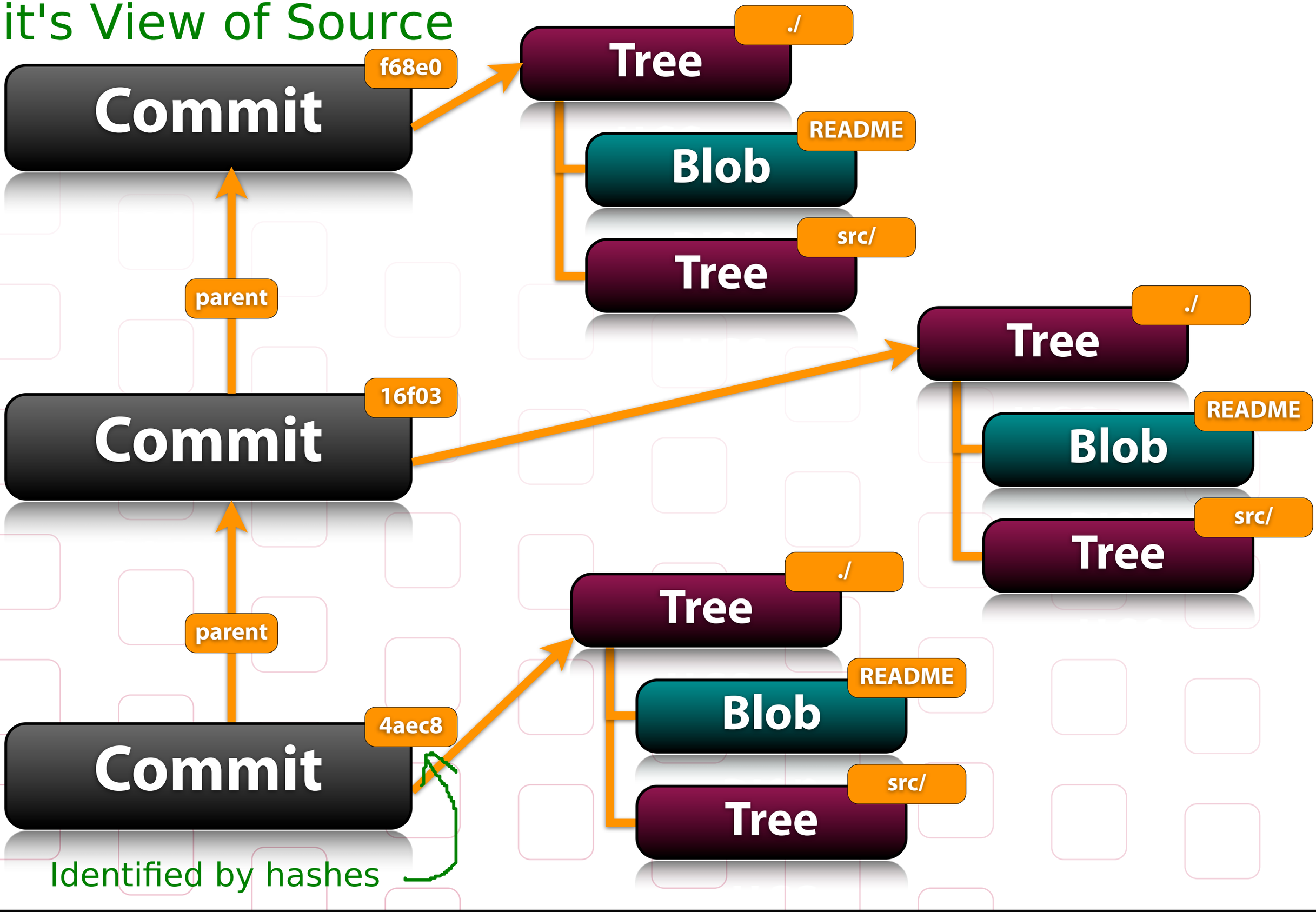
Git is an open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Potential source of confusion with git: modifying the local repo and a remote are separate steps.

User's View of Git Source Tree

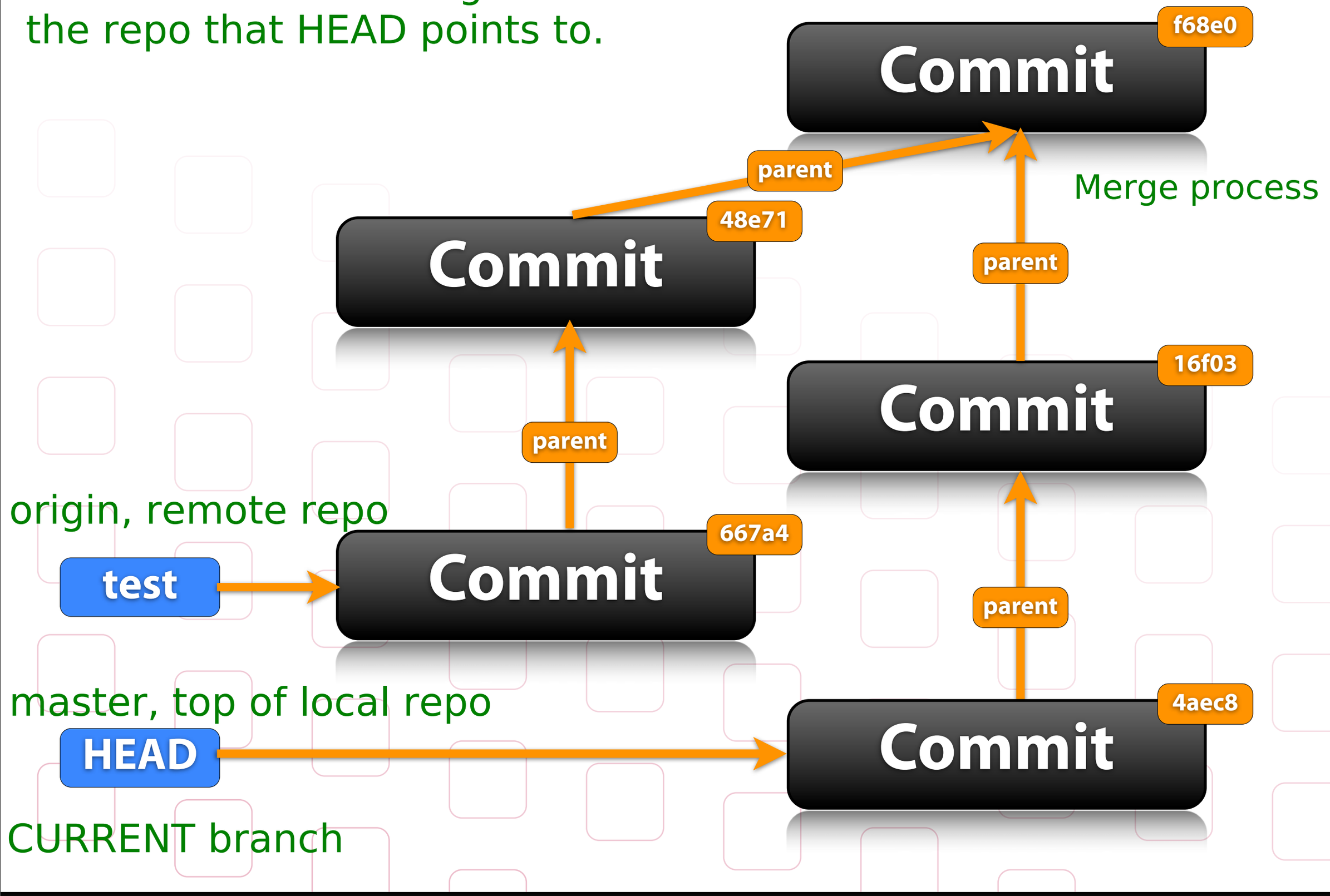


Git's View of Source



"Rebase" means change
the repo that HEAD points to.

Our current source tree



git config -l ==> settings for CWD

Part II: Basic Git Commands

Where do we begin?

```
puls@subtlefuge:~ $ git config --global user.name ‘Your Name’
puls@subtlefuge:~ $ git config --global user.email ‘your@address.com’
puls@subtlefuge:~ $
```

Goes in $HOME/.gitconfig. Also set GIT_EDITOR.

There are eight basic Git commands:

`git init`	`git status`
`git clone`	`git add`
`git commit`	`git diff`
`git log`	`git reset`

These commands address the local repository.

`git init`

"Make a brand new, empty repository that I can fill with what I'm working on now."

```
puls@subtlefuge:~/mysource $ git init
Initialized empty Git repository in /Users/puls/mysource/.git/
puls@subtlefuge:~/mysource $
```

Git is useful for keeping track of your own projects (your blog, your own scripts . . .) in a local repo -- no need to use github or gitorious.

`git status`

"Show me what has happened since the last commit."

```
puls@subtlefuge:~/mysource $ git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
puls@subtlefuge:~/mysource $
```

git status --help for options

`git clone`

"Start by making me a copy of that other repository over there so I can work on it."

git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git

```
puls@subtlefuge:~ $ git clone git://github.com/dustball/hdsignin.git
Initialized empty Git repository in /Users/puls/hdsignin/.git/
remote: Counting objects: 17, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 17 (delta 4), reused 0 (delta 0)
Receiving objects: 100% (17/17), 14.83 KiB, done.
Resolving deltas: 100% (4/4), done.
puls@subtlefuge:~ $
```

git clone git://github.com/dneary/linux-devel-tools-tutorial.git

Adds occur by-value, not by-reference: git takes snapshots!

`git add`

"Stage these files for the next commit that I'm about to create."

```
puls@subtlefuge:~/mysource $ echo "Hello world" > README
puls@subtlefuge:~/mysource $ git add README
puls@subtlefuge:~/mysource $ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#     new file:   README
#
puls@subtlefuge:~/mysource $
```

`git commit`

"Take the stuff I just staged and turn it in to a recorded snapshot. Also update the HEAD and 'current branch' references."

```
puls@subtlefuge:~/mysource $ echo "Hello world" > README
puls@subtlefuge:~/mysource $ git add README
puls@subtlefuge:~/mysource $ git commit -m "I made a readme"
[master (root-commit) ddd9c89] I made a readme
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 README
puls@subtlefuge:~/mysource $
```

Changes to local repository HEAD, not to origin.

`git checkout`

"Make my source tree reflect a commit referenced by this name."

Checking out another branch allows reversion or merging of one or more files

```
puls@subtlefuge:~/mysource $ git checkout two
Switched to branch 'two'
puls@subtlefuge:~/mysource $ git checkout master
Switched to branch 'master'
puls@subtlefuge:~/mysource $
```

Short branch names set via "remote" command (below).

git merge

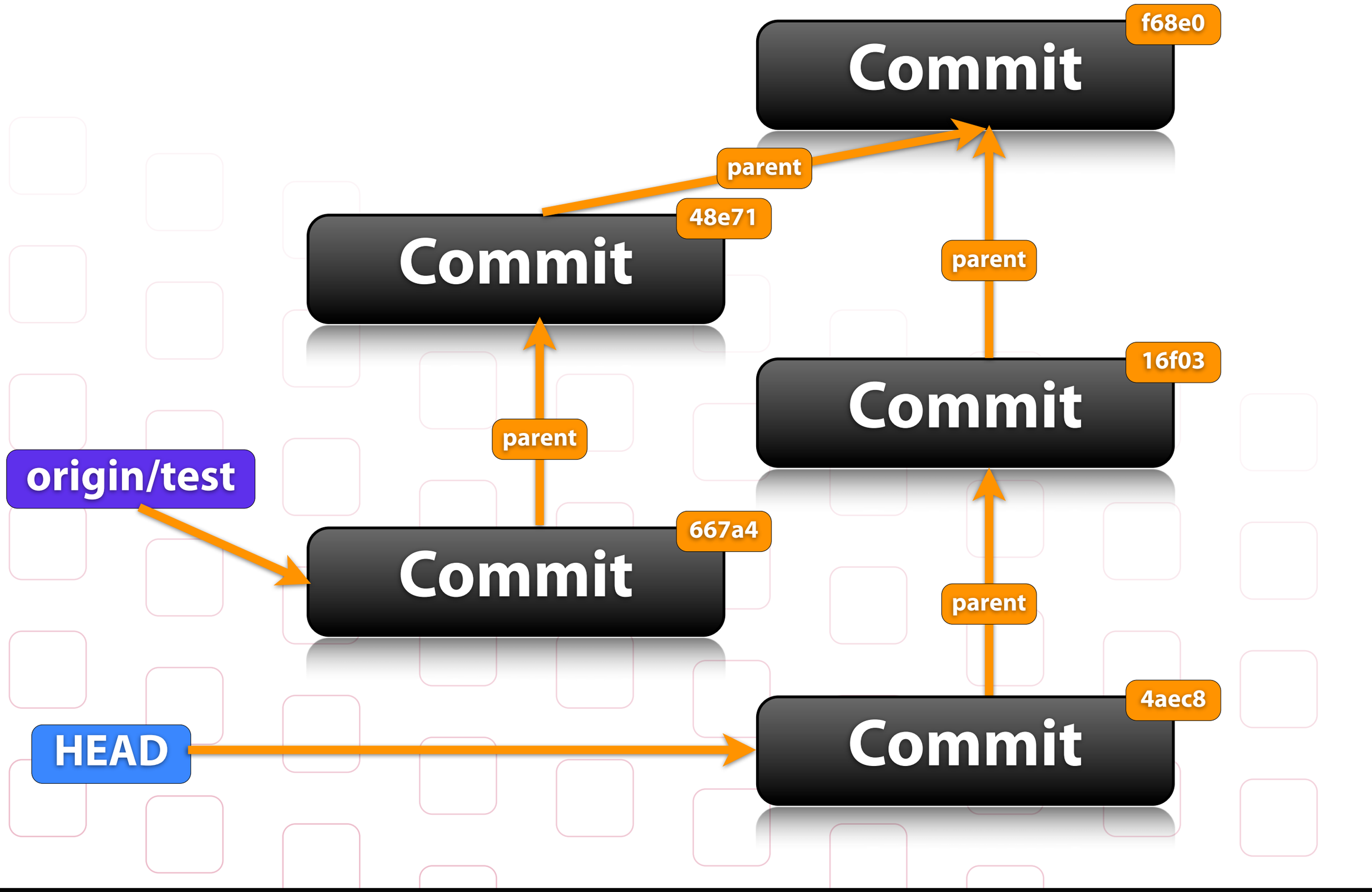
"Make a new commit that reflects all of the changes in two different commit histories."

```
puls@subtlefuge:~/mysource $ cat README
New line at the beginning
Hello world
Goodbye world
puls@subtlefuge:~/mysource $ git checkout two; cat README
Switched to branch 'two'
Hello world
Goodbye world
New line at the end
puls@subtlefuge:~/mysource $ git checkout master; git merge two
Auto-merging README
Merge made by recursive.
 README |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
puls@subtlefuge:~/mysource $
```

git merge

"Make a new commit that reflects all of the changes in two different commit histories."

```
puls@subtlefuge:~/mysource $ cat README
New line at the beginning
Hello world
Goodbye world
New line at the end
puls@subtlefuge:~/mysource $
```


Commit
f68e0
parent
Commit
48e71
parent
origin/test
parent
Commit
16f03
Commit
667a4
parent
HEAD
Commit
4aec8

Commands referring to remote repositories

Still three more commands to learn:

`git remote`
`git fetch`
`git push`

`git remote`

"Add or remove a named shorthand for a remote copy of this repository."

```
puls@subtlefuge:~/hdsignin $ git remote add puls \
git://github.com/puls/hdsignin.git
puls@subtlefuge:~/hdsignin $ git remote -v
origin  git://github.com/dustball/hdsignin.git (fetch)
origin  git://github.com/dustball/hdsignin.git (push)
puls  git://github.com/puls/hdsignin.git (fetch)
puls  git://github.com/puls/hdsignin.git (push)
puls@subtlefuge:~/hdsignin $
```

`git fetch`

"Copy over all of the objects and references I don't have already from the remote database."

```
puls@subtlefuge:~/hdsignin $ git fetch puls
From git://github.com/puls/hdsignin
 * [new branch]      master     -> puls/master
puls@subtlefuge:~/hdsignin $
```

New files in origin not fetched? If "git status" shows
"Your branch is behind origin/master," you need to rebase.
"git rebase origin" applies patches from HEAD to a local copy of remote origin.
"git fetch" then copies missing files in origin to local repo.

git push

"Send a particular reference and all of the objects it points at over to that repository ."

```
puls@subtlefuge:~/hdsignin $ git remote add mine \
git@github.com:puls/hdsignin.git
puls@subtlefuge:~/hdsignin $ git push mine
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 267 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:puls/hdsignin.git
   fa0d670..7643aac  master -> master
puls@subtlefuge:~/hdsignin $
```

git rebase

"Instead of merging, apply each change I've made as a patch over that version."

```
puls@subtlefuge:~/mysource $ git log --all --graph --pretty=oneline
* d5282460d8524c7aa1854d47d314d531ab39928a Commit on branch
| * f4a56b15d313383802cd571a5cb47c4ce14fc921 Second commit
|/
* 8be940dfb3e10e63e04e8b73990618eb9b7827e0 First commit
puls@subtlefuge:~/mysource $ git checkout top && cat README
Switched to branch 'top'
Zero
One
puls@subtlefuge:~/mysource $ git checkout master && cat README
Switched to branch 'master'
One
Two
puls@subtlefuge:~/mysource $
```

git rebase

"Instead of merging, apply each change
I've made as a patch over that version."

In other words, change the base of the local repo.

```
puls@subtlefuge:~/mysource $ git rebase top
First, rewinding head to replay your work on top of it...
Applying: Second commit
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
Auto-merging README
puls@subtlefuge:~/mysource $ git log --all --graph --pretty=oneline
* 3197886f076afd504fbd453cbadbd26bea7eb99a Second commit
* d5282460d8524c7aa1854d47d314d531ab39928a Commit on branch
* 8be940dfb3e10e63e04e8b73990618eb9b7827e0 First commit
puls@subtlefuge:~/mysource $
```

Useful tools:

http://gitref.org

git --help

gitk and gitview graphical source-tree viewers

git-completion.bash script allows prompt-string setting to match CWD

New sparkleshare cloud-based storage and sync service uses git for transferring and versioning.

Patience!

Even pros sometimes run into problems.

See "Patch submission practices" and "Messy merging" in "Zack's Kernel News" column of Nov. 2010 _Linux Pro Magazine_.