GIT

- History
- Git basics
- Using git
- Branching and merging
- Remote repositories
- Summary

SCA meeting Nov. 22, 2010

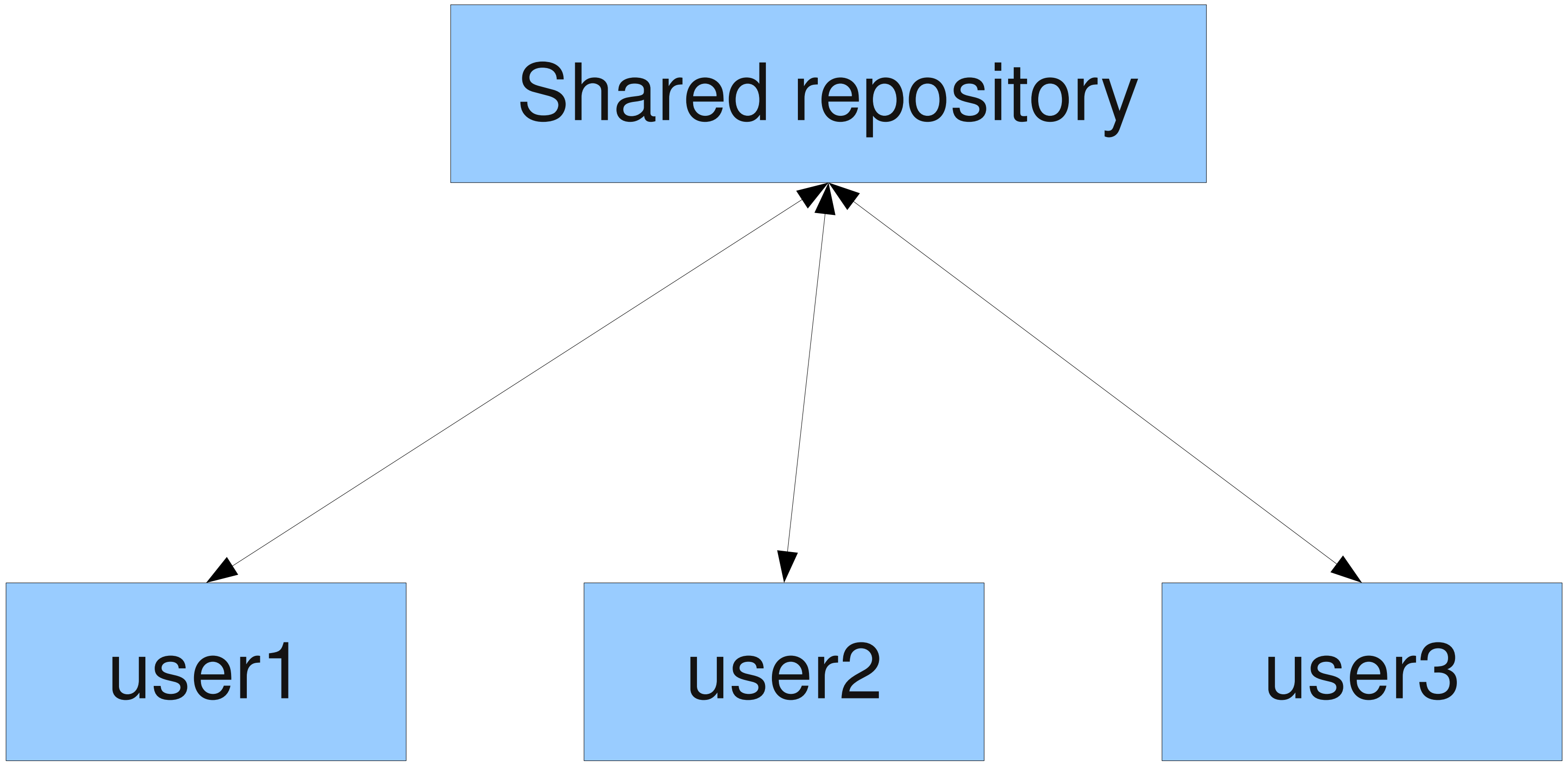
History

- -2002 Tar + patches used by Linux kernel
- 2002-2005 agreement to use commercial VCS BitKeeper free of charge.
- 2005 free usage was revoked. Development of git started. Design goals:
  - Fast
  - Distributed
  - Scalable to large projects (like linux kernel)
  - Strong support for branching

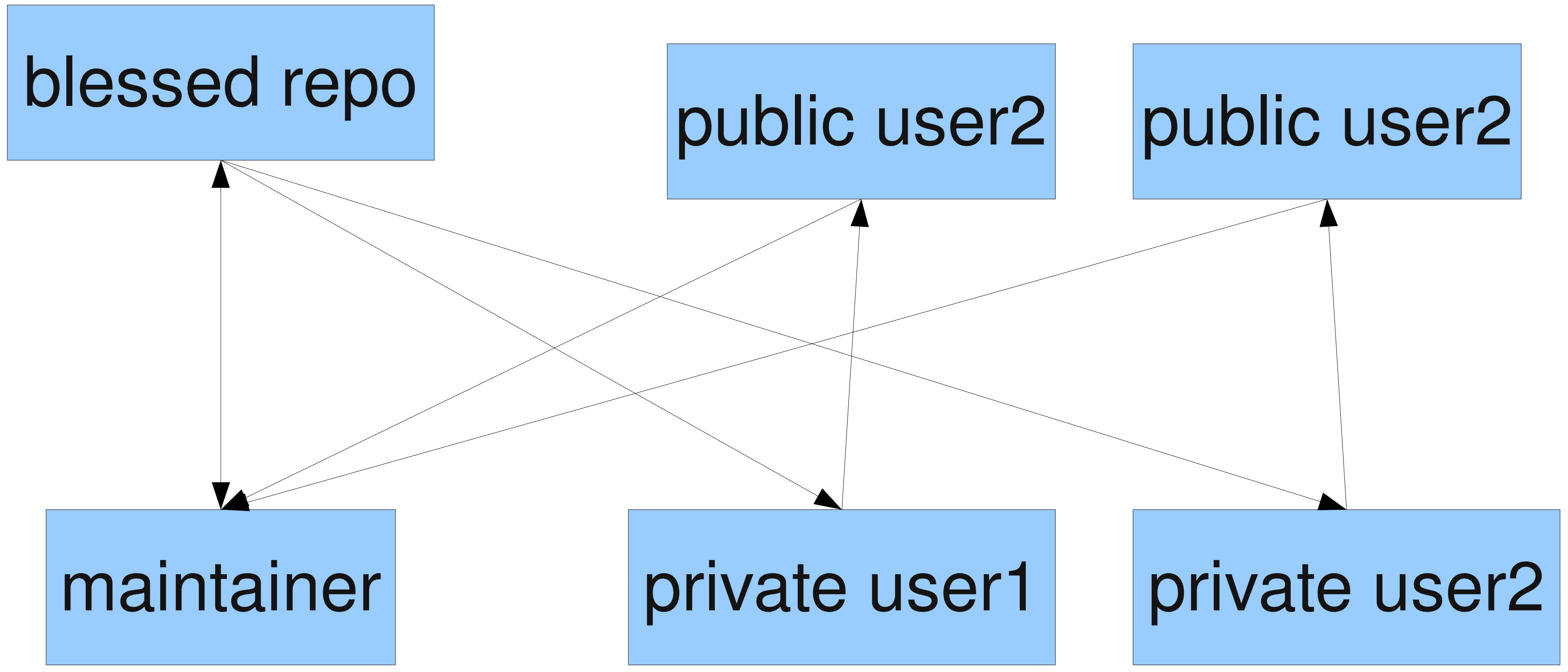
Distributed VCS

- No centralized repository
- Each developer has its own repository
- Developer share changes
  - Push or pull changes between repositories
- Most operations are local (no network conn.)
- Local branches and commits
- Different possible workflows
- Other DCVS: mercurial, bazaar

Shared main repository everyone can push or pull

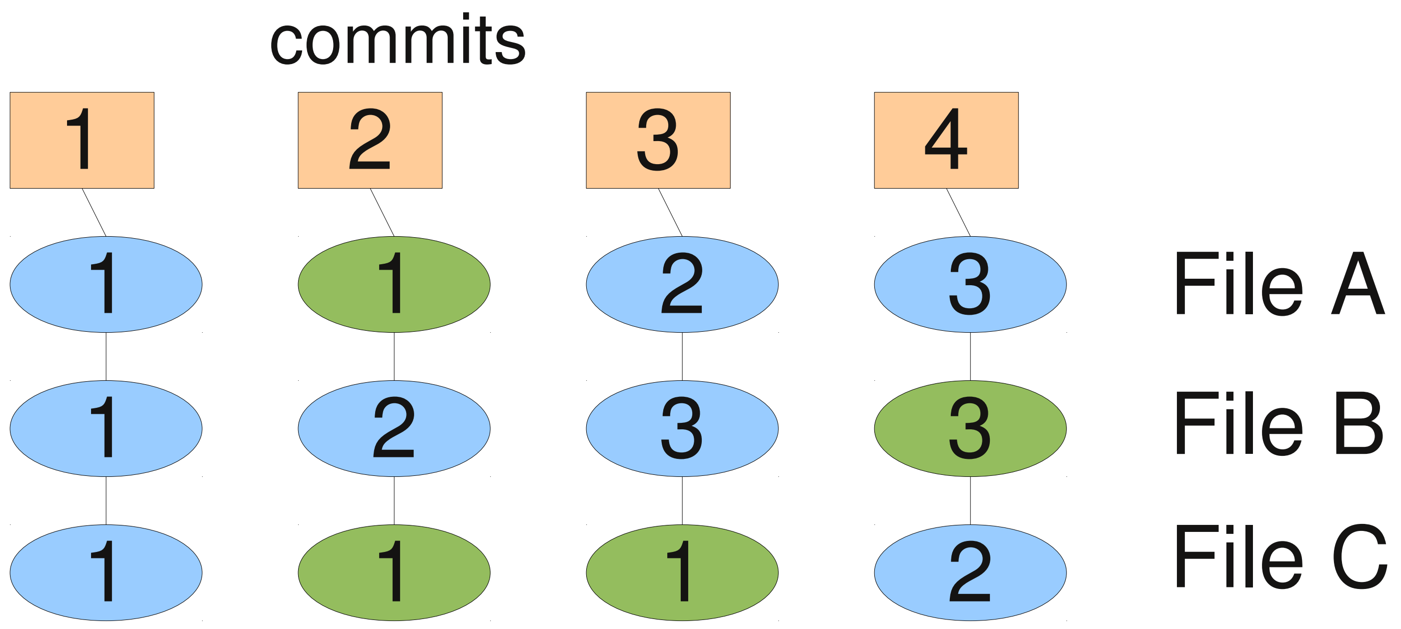


A single maintainer manages a blessed repository



Basics

- Git tracks the status of directories not single files.
- For every commit a snapshot is stored (no diffs).

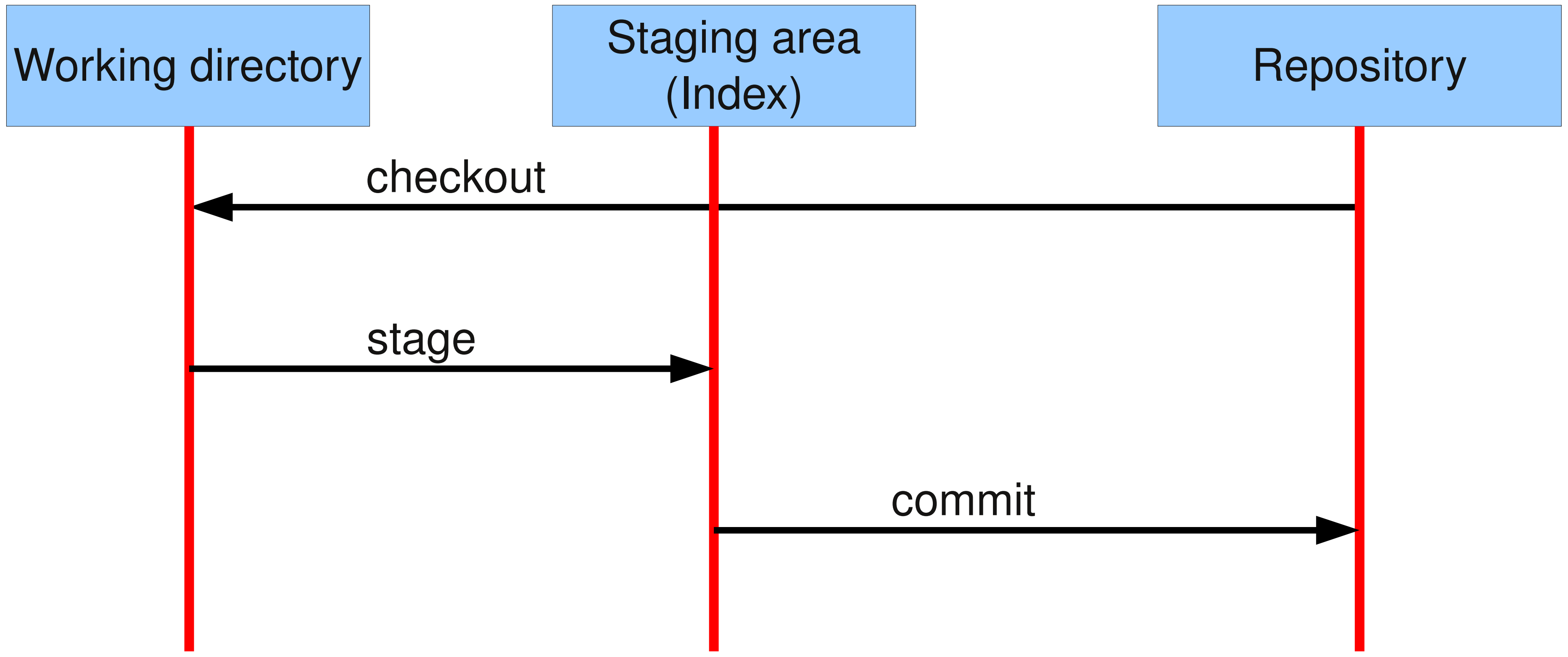


Basics

- Everything is checksummed before stored
- SHA-1 is used
- Checksum is used to reference objects

  (files, commit)
- Allows to detect corruption

Interaction between working dir and repository



Using Git

Create/clone repository

```
> cd yourProject
> git init
> git add .          # add all files in current dir
> git commit
> ls .git
    branches/  config  description  HEAD  hooks/
    index  info/  logs/  objects/  packed-refs  refs/

> git clone http://xrootd.org/repo/xrootd.git
> git clone ssh://iris/path/to/xrootd.git
> git clone /afs/slac/..../xrootd.git
```

Git log

```
> git log
commit 26dc3df9f3beb68a7ebbf3df23c282e0f5e56716
Author: Andrew Hanushevsky <abh@stanford.edu>
Date:   Thu Oct 14 13:55:42 2010 -0700

    Fix to actually use /dev/urandom after discovering the machine has it.

commit 4c3c728e8df07b7265f7e028500f9493a8203166
Author: Andrew Hanushevsky <abh@stanford.edu>
Date:   Mon Oct 11 19:11:57 2010 -0700

    Implement the handling of the kXR_replica open() option. This option
allows the creation of replicas by giving you a server that doesn't have the file.
```

There are many options to git log

Git status

```
> git status
```

```
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   src/XrdClient/XrdCommandLine.cc
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Git diff

> git diff

```
diff --git a/src/XrdClient/XrdCommandLine.cc b/src/XrdClient/XrdCommandLine.cc
index 9f80a32..3d8566e 100644
--- a/src/XrdClient/XrdCommandLine.cc
+++ b/src/XrdClient/XrdCommandLine.cc
@@ -1530,7 +1530,7 @@ int main(int argc, char**argv) {

           char *reqcode = tkzer.GetToken(0, 0);
           const kXR_char *args = (const kXR_char *)tkzer.GetToken(0, 0);
-          kXR_char Resp[1024];
+           kXR_char Resp[2048];

           genadmin->Query(atoi(reqcode), args, Resp, 1024);
```

Edit → add → commit

```
> edit file        (git mv ;  git rm)

> git status
  # Changed but not updated:
  #   (use "git checkout -- <file>..." to discard changes in working directory)
  #       modified:   src/XrdClient/XrdCommandLine.cc

> git diff  [file]

> git add             (> git commit -a)

> git status
  # Changes to be committed:
  #   (use "git reset HEAD <file>..." to unstage)
  #       modified:   src/XrdClient/XrdCommandLine.cc

> git commit
```

create a tag

Each git commit is unique and could be used as a tag but hard to read:

```
55db755c81adf99d87b4f42320c0bd421c265e25
```

```
> git tag                # list all tags
```

```
> git tag 3.0.0  -m “greatest ever xrootd version”
```

Annotated tag

```
> git tag rc1-3.0.0
```

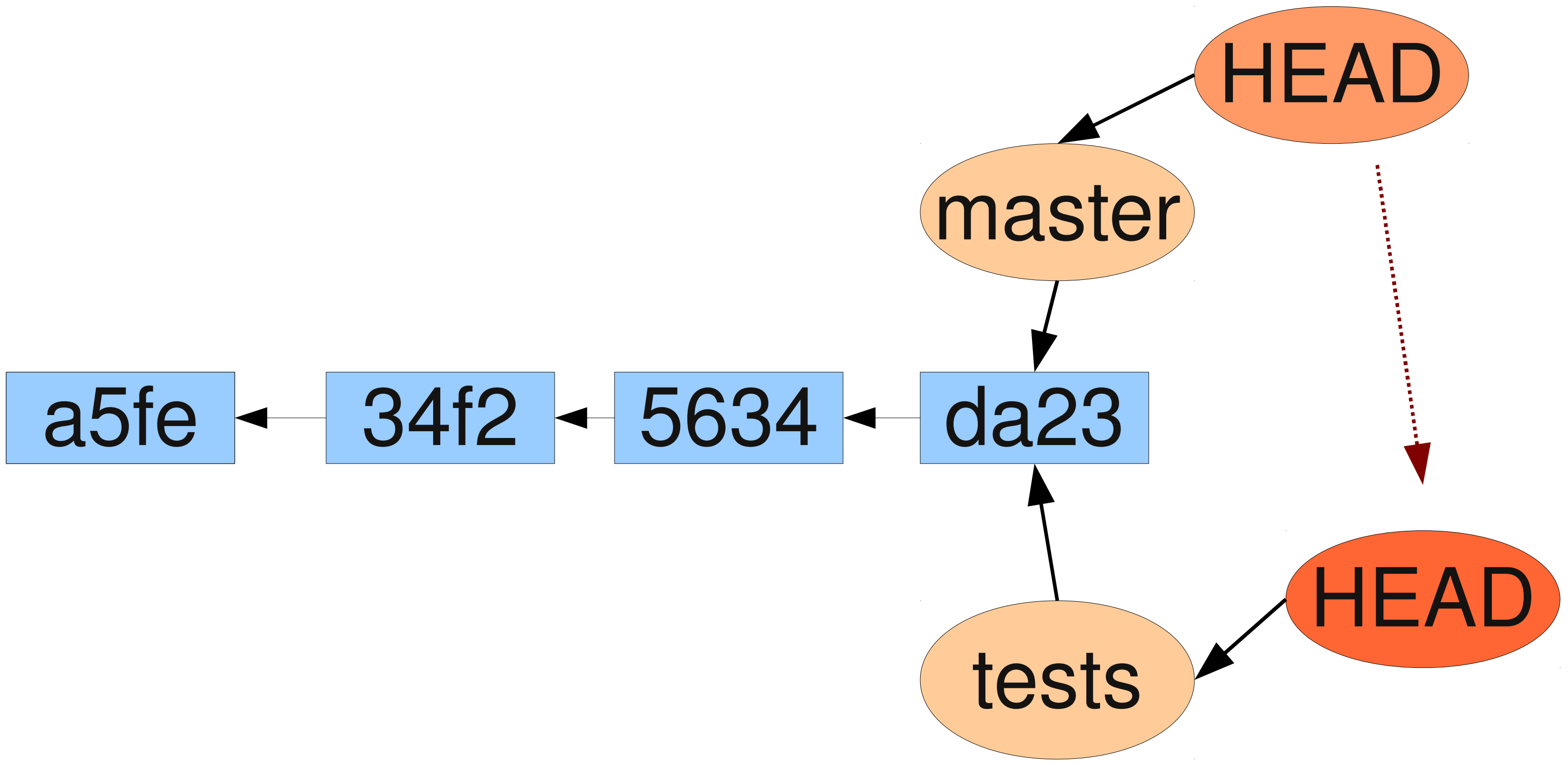
Light weight tag

```
> git show rc1-3.0.0
```

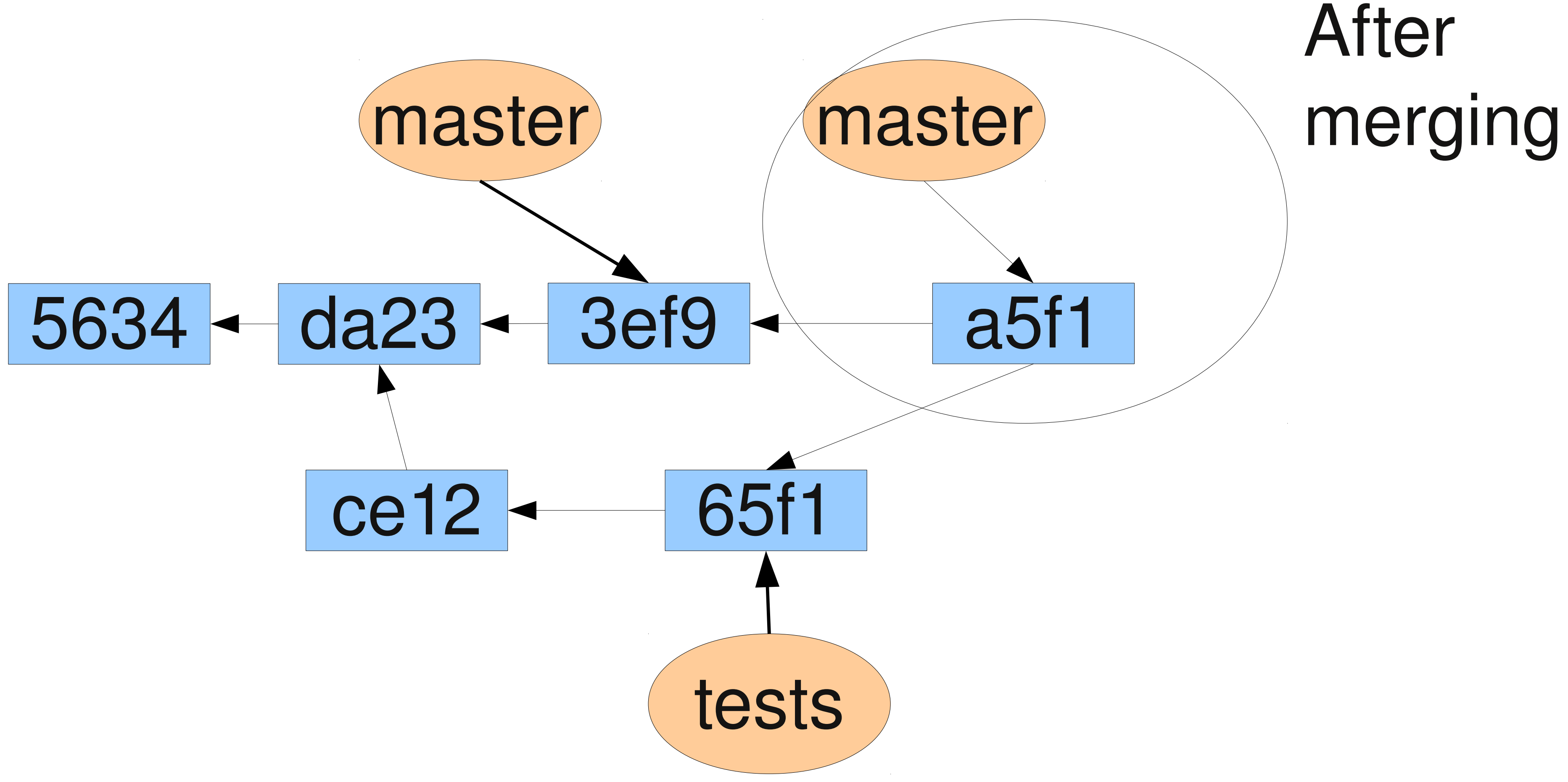
```
tag rc1-3.0.0        Tagger: Wilko Kroeger <wilko@slac.stanford.edu>
Date:   Wed Oct 13 15:46:41 2010 -0700        Release candidate for 3.0.0
```

Branching

```
> git branch tests
> git checkout tests
```



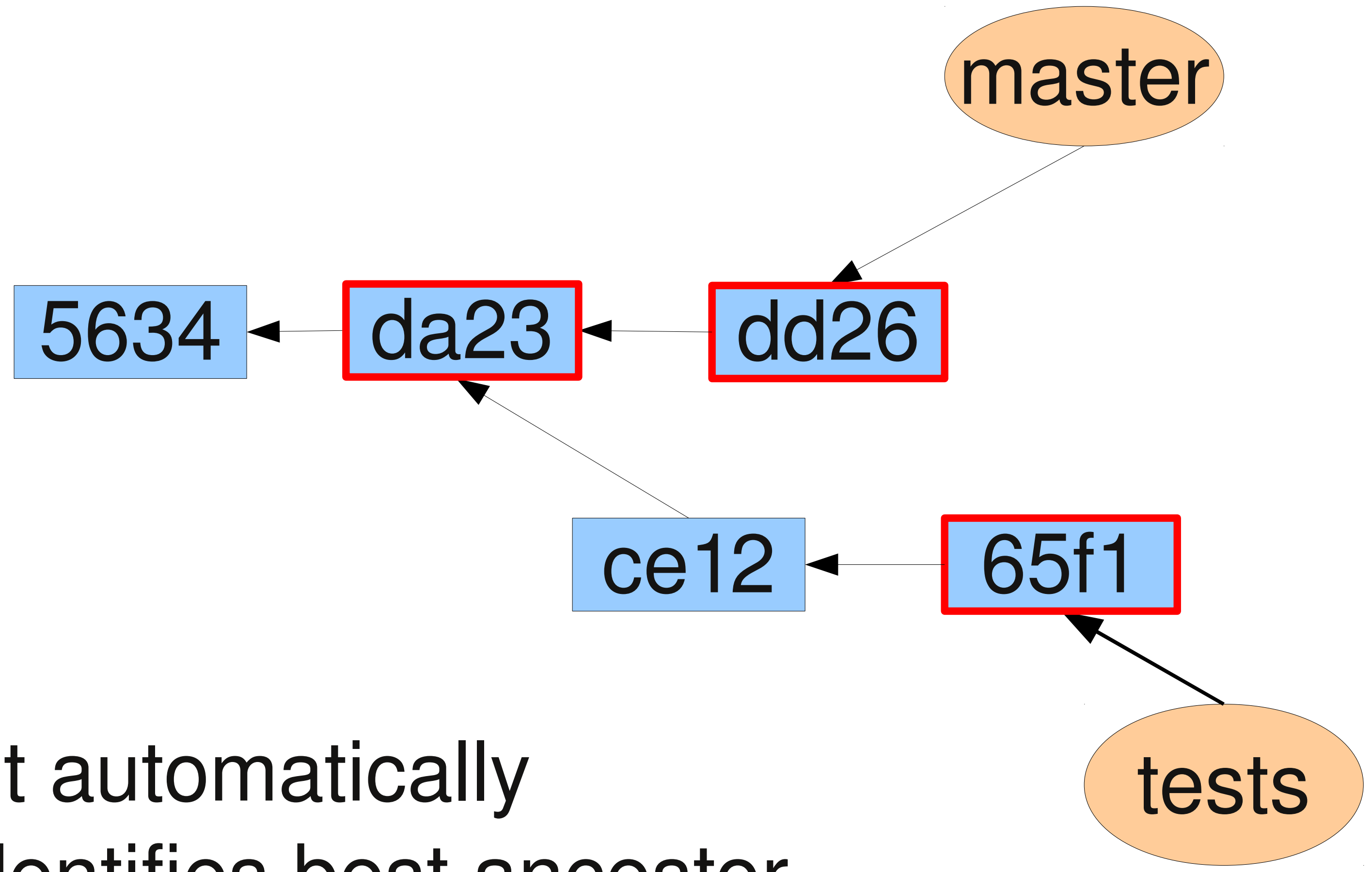
updates to branches and merging



```
> git checkout master
> git merge tests
```

Three way merge

Git use the two branches and the common anchestor for merging



git automatically
Identifies best ancestor

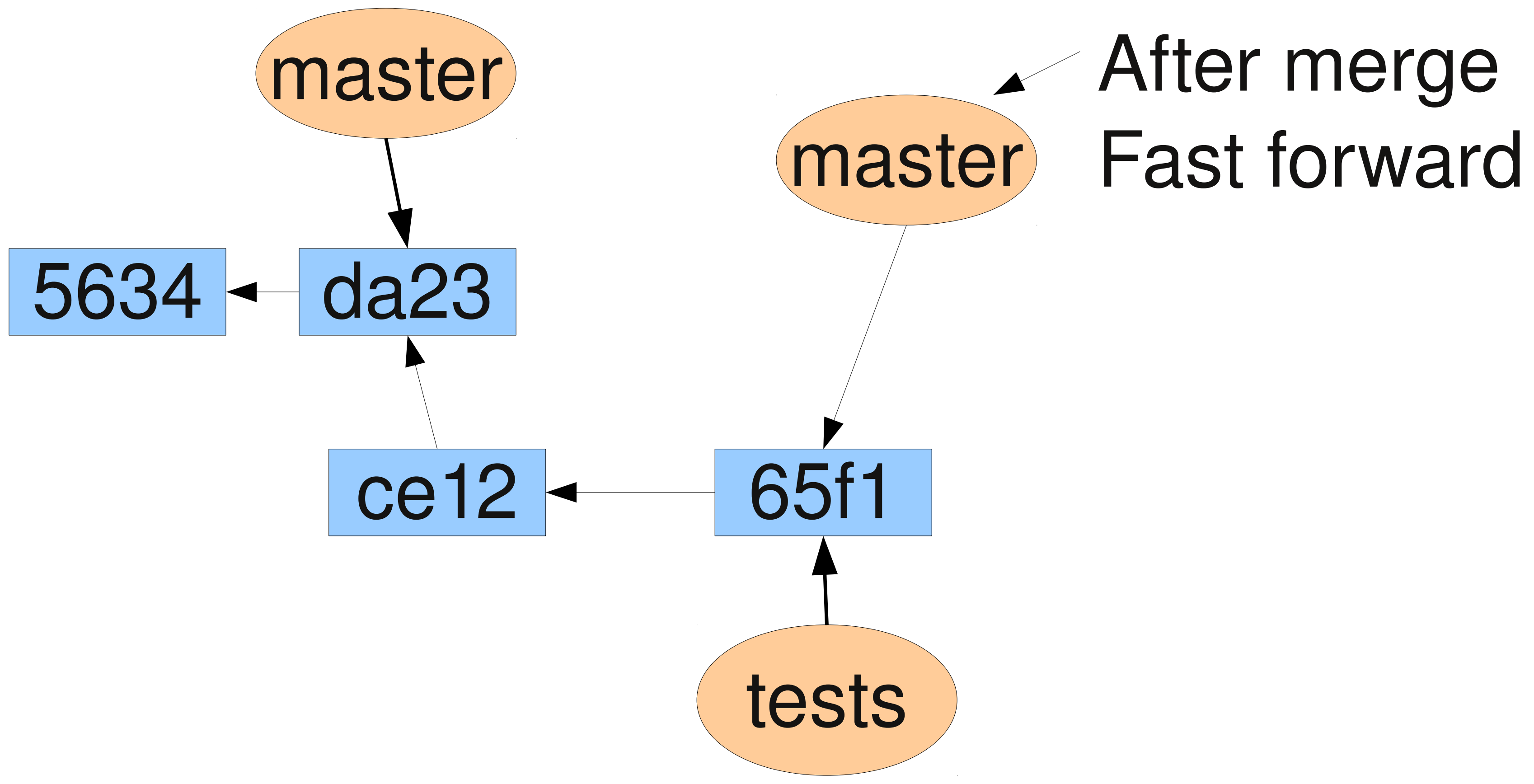
merge conflicts

```
> git commit        # will halt operation if conflicts
> git status        # shows files with conflicts
```

Resolve conflicts (marker in files)
e.g.: git mergetool

```
> git add
> git commit
```

Fast forward merge



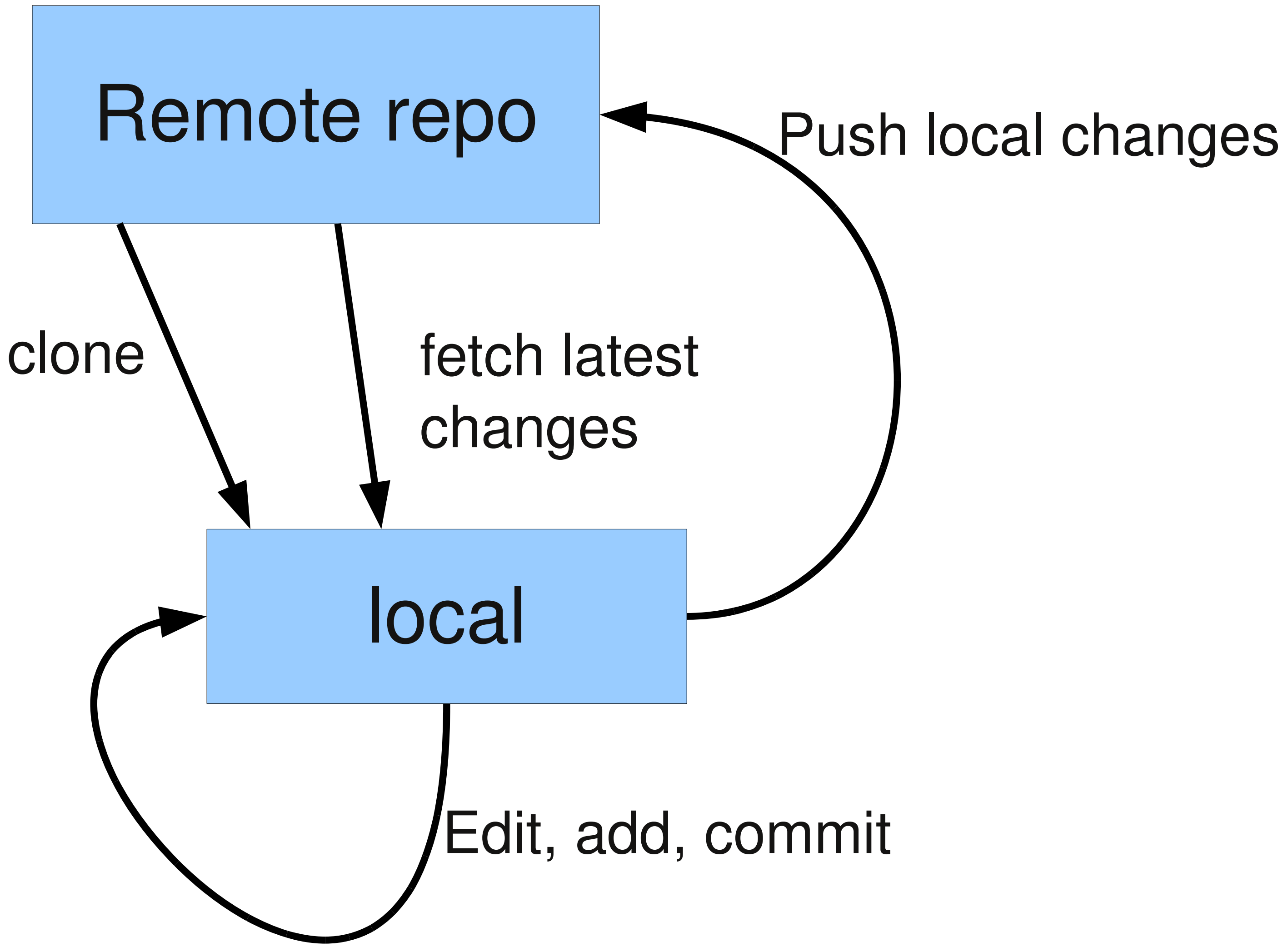
```
> git checkout master
> git merge tests
```

More Branching

```
> git branch           # list branches
    * master
      tests

> git branch -d test       # remove a branch
```

Remote repository



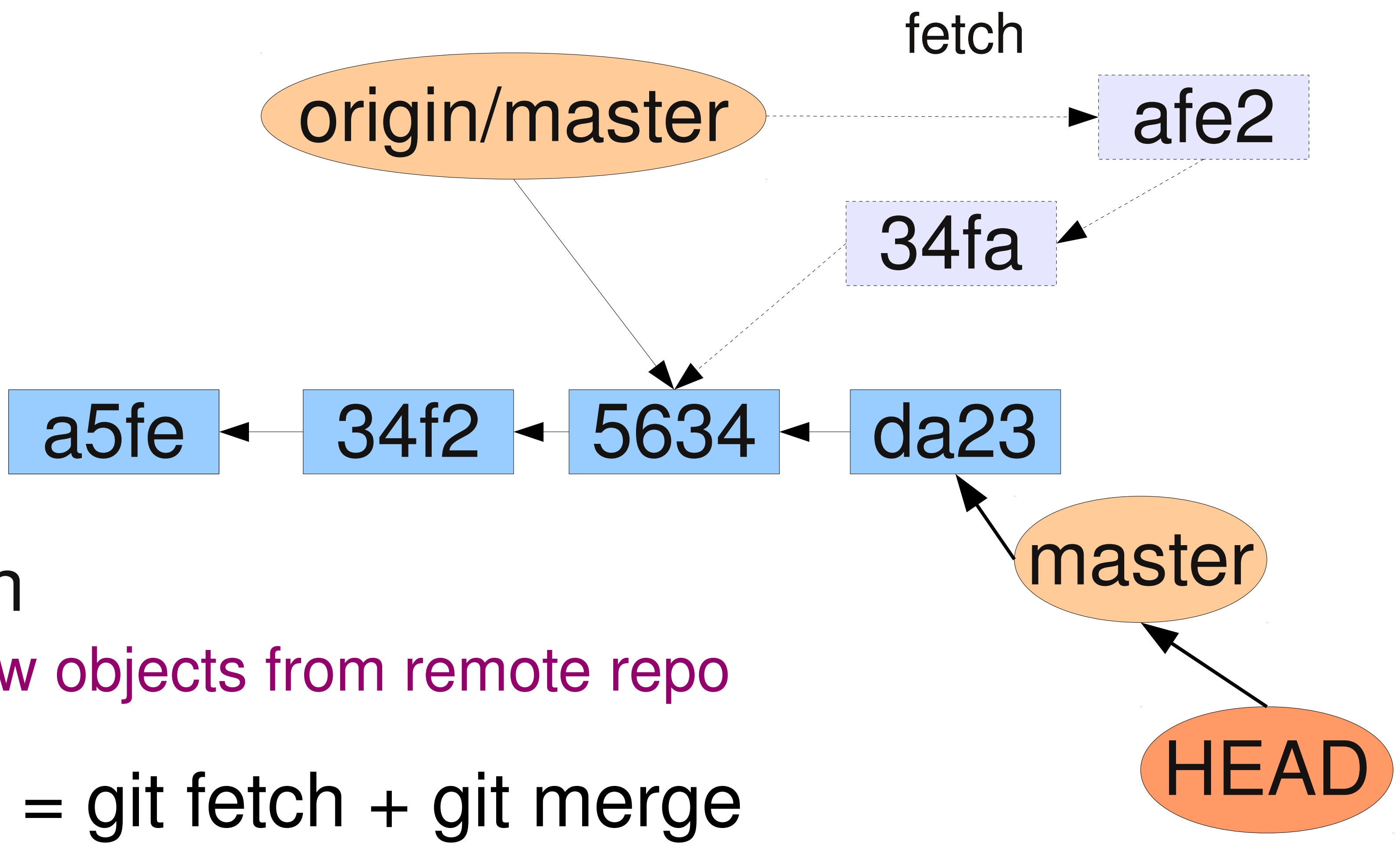
Remote repository

```
> git remote -v
    origin  http://xrootd.org/repo/xrootd.git
```

```
> git remote show origin
  * remote origin
    URL: http://xrootd.org/repo/xrootd.git
    HEAD branch: master
    Remote branches:
      master                  tracked
      v20071001-0000-patches tracked
      v20081007-0500-patches tracked
     Local branch configured for 'git pull':
        master merges with remote master
     Local ref configured for 'git push':
        master pushes to master (up to date)
```

```
> git remote add wk  /path/to/remore/repo.git
```

Remote repository



> git fetch

Get all new objects from remote repo

> git pull = git fetch + git merge

> git push

Push local master to the remote repo. Local repo has to be up-to-date.

Not talked about

- Configure git (name, email, editor, color,...)
- Ignoring files
- Hooks (e.g.: email notification)
- Attributes
  - Keyword Expansion (but not like CVS)
  - Merge strategy
  - Diffs for binary files
- git bisect, git rebase
- I am sure much more

Access to git and tools

- local   file:// or /path/../..
- ssh   ssh://   or user@host:/path/
- http   http://   (only read)
- Git server   git://   (only read)
- Public and private hosting: GitHub, gitorious
- Browsing history: gitk , qgit , gitX, TortoiseGit
- Plugins for: Eclipse, Netbeans, emacs, ...

Git users

- Git
- Linux Kernel
- Perl
- Gnome
- Qt
- Ruby on Rails
- Android
- PostgresSQL
- Wine
- Fedora
- Debian
- X.org
- Xrootd

Summary

- Easy to use to use and fast
- Allows frequent commits without inflicting others
- Strong support for branching
- Can be used offline
- Tools to convert CVS/SVN to git
- Lots of documentation
- Fun to use

More Info

GIT home page:

http://git-scm.com

Documentation:

http://git-scm.com/documentation

Man pages:

http://www.kernel.org/pub/software/scm/git/docs