
git

10,000 feet

Git is an open source, distributed version control system designed for speed and efficiency.

Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.

Internals

SHA-1 hash

`af5626b4a114abcb82d63db7c8082c3c4756e51b`

a checksum of the content you're storing plus a header

blob

immutable piece of content

```
$ echo 'Hello, world!' > greeting

$ git hash-object greeting
af5626b4a114abcb82d63db7c8082c3c4756e51b
```

tree

like a directory

contains the metadata (e.g. filename)

has blobs and other trees as leaf nodes

commit

snapshot of your repository

holds a tree and metadata

has one or more parents

only deltas are stored

understand commits, you understand git

branch

a named commit

name moves with the latest commit

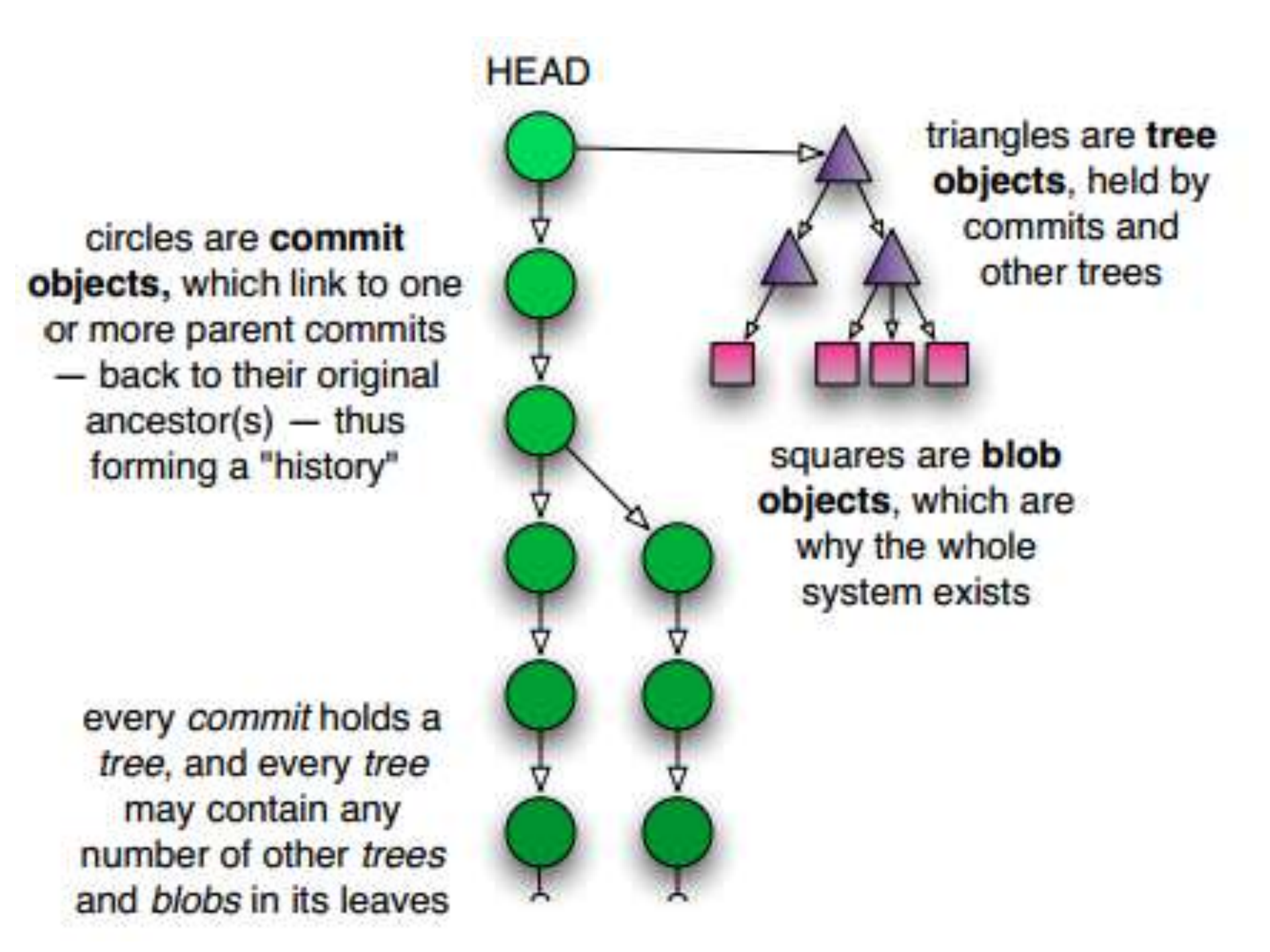
tag

a named commit

will always point at the same commit

HEAD

the latest commit in current branch


HEAD
triangles are **tree objects**, held by commits and other trees
circles are **commit objects,** which link to one or more parent commits — back to their original ancestor(s) — thus forming a "history"
squares are **blob objects**, which are why the whole system exists
every *commit* holds a *tree*, and every *tree* may contain any number of other *trees* and *blobs* in its leaves

commit names

branchname

tagname

HEAD^

HEAD^^

tagname~5

af562..3d29a

af562..HEAD

af562..tagname^^^

basic commands

status / log

diff

remote / branch -av

.gitconfig / aliases

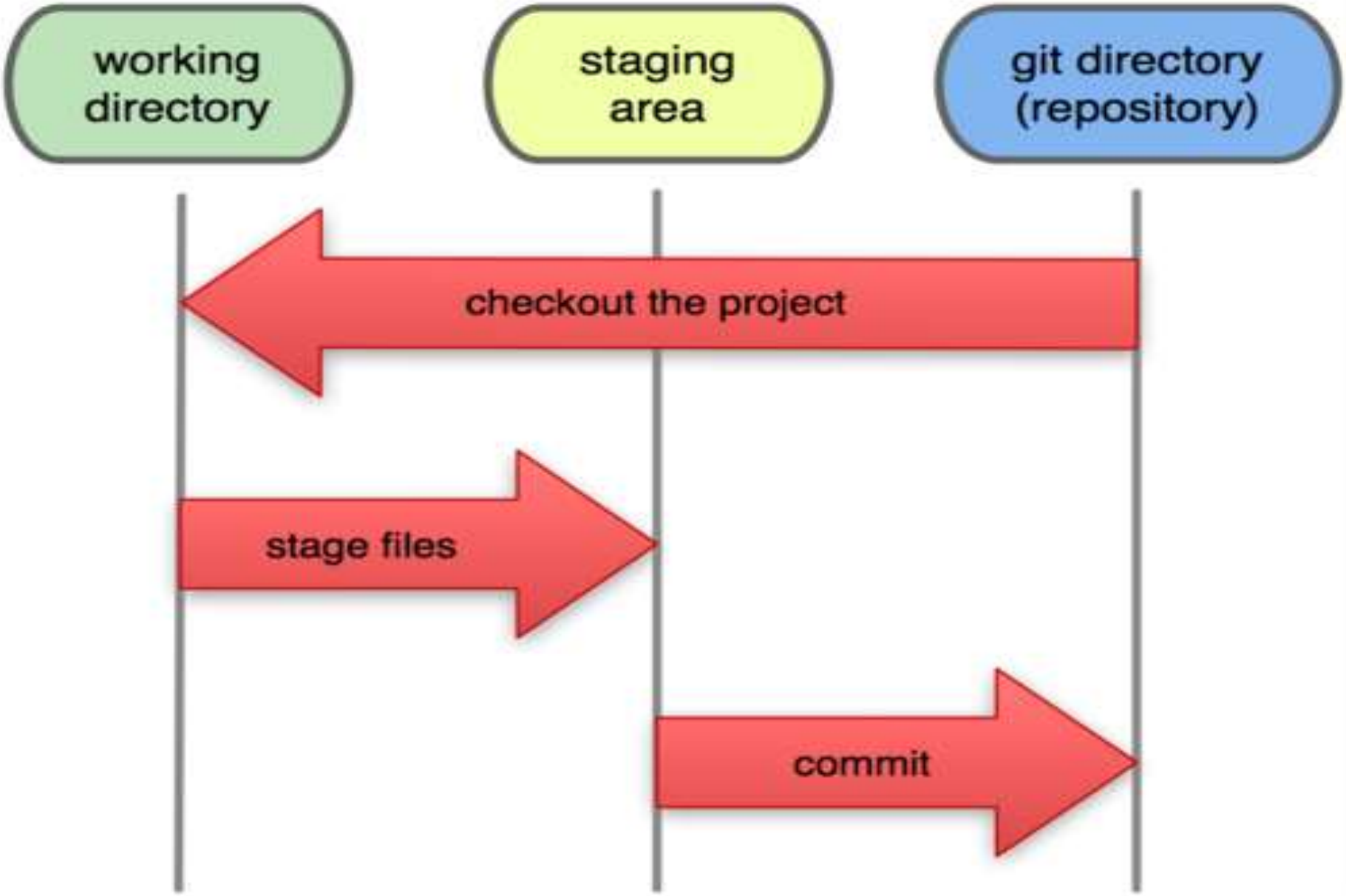
pull --rebase

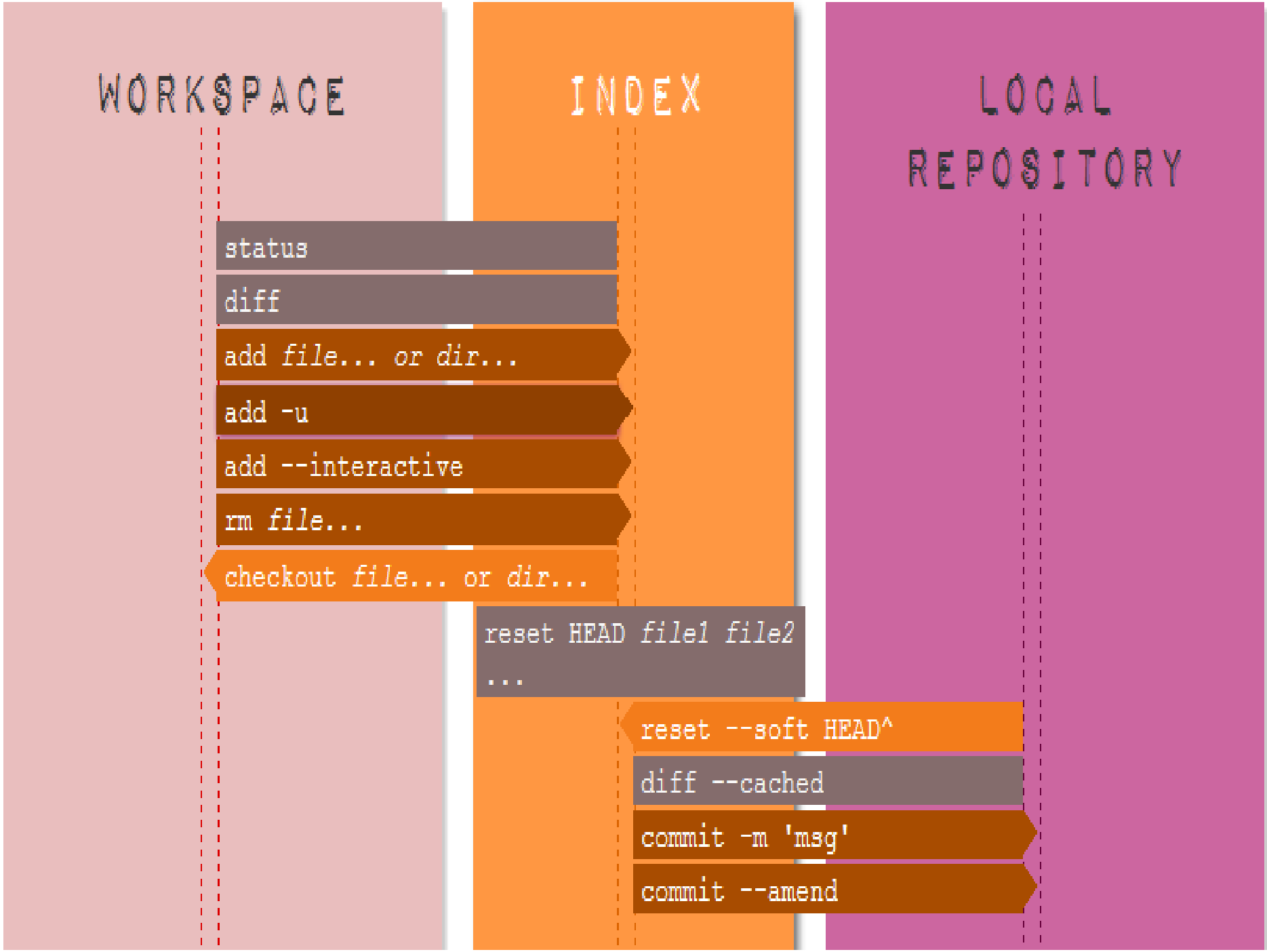
Workflow

Changes go through several stages before ending up in their final destination

working dir	current checkout - editing happens here
index	aka "cache" aka "staging area" - changes that are selected to be commited
commit	now packaged up with a commit message and part of the history
master branch	move the commits over when the feature is finished
origin	get the changes upstream

Local Operations




WORKSPACE
INDEX
LOCAL REPOSITORY
status
diff
add file... or dir...
add -u
add --interactive
rm file...
checkout file... or dir...
reset HEAD file1 file2 ...
reset --soft HEAD^
diff --cached
commit -m 'msg'
commit --amend

Merge branch

Before

```
                A---B---C topic
               /
          D---E---F---G master
```

Command

```
$ git merge topic
```

After

```
                A---B---C topic
               /         \
          D---E---F---G---H master
```

Rebase branch

Before

```
                A---B---C topic
               /
          D---E---F---G master
```

Command

```
$ git rebase master topic
```

After

```
                          A---B---C topic
                         /
          D---E---F---G master
```

Alternative

```
$ git rebase -i topic
```

pro commands

squash / rebase -i / cherry-pick

fork / pull requests (github)

merge

bisect

Q & A