# Intro to Git and GitHub

**Boston Code Camp 2018**

Andrew Babiec

2018-04-07

# Boston Code Camp 29 - Thanks to our Sponsors!

**Platinum**

Microsoft

**Gold**

HealthcareSource®
Quality Talent Suite™

**Silver**

slalom

BLUEMETAL

rh Robert Half®
Technology

**Bronze**

RGood
SOFTWARE

In-Kind Donations

GrapeCity.

MANNING PUBLICATIONS

JET BRAINS

steema
software

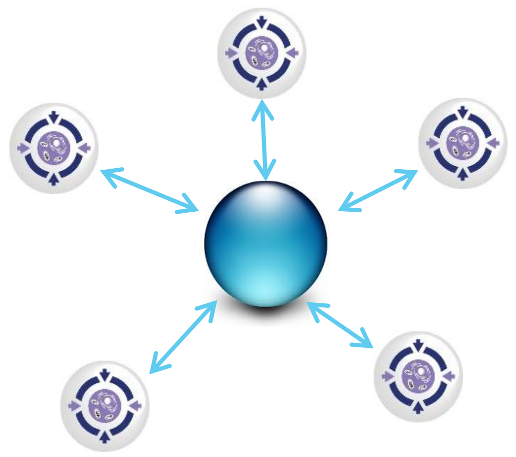
## What is Git?

Distributed Version Control System

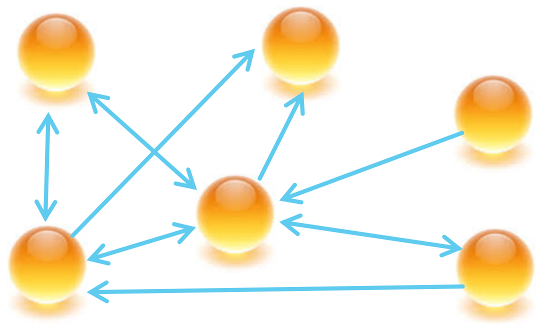
- Version Control System: track changes to source code: who, what, why, when

# Centralized vs. Distributed

Centralized repository



Distributed repository



# Git - Distributed Version Control System

Git **Repository** (repo/project)

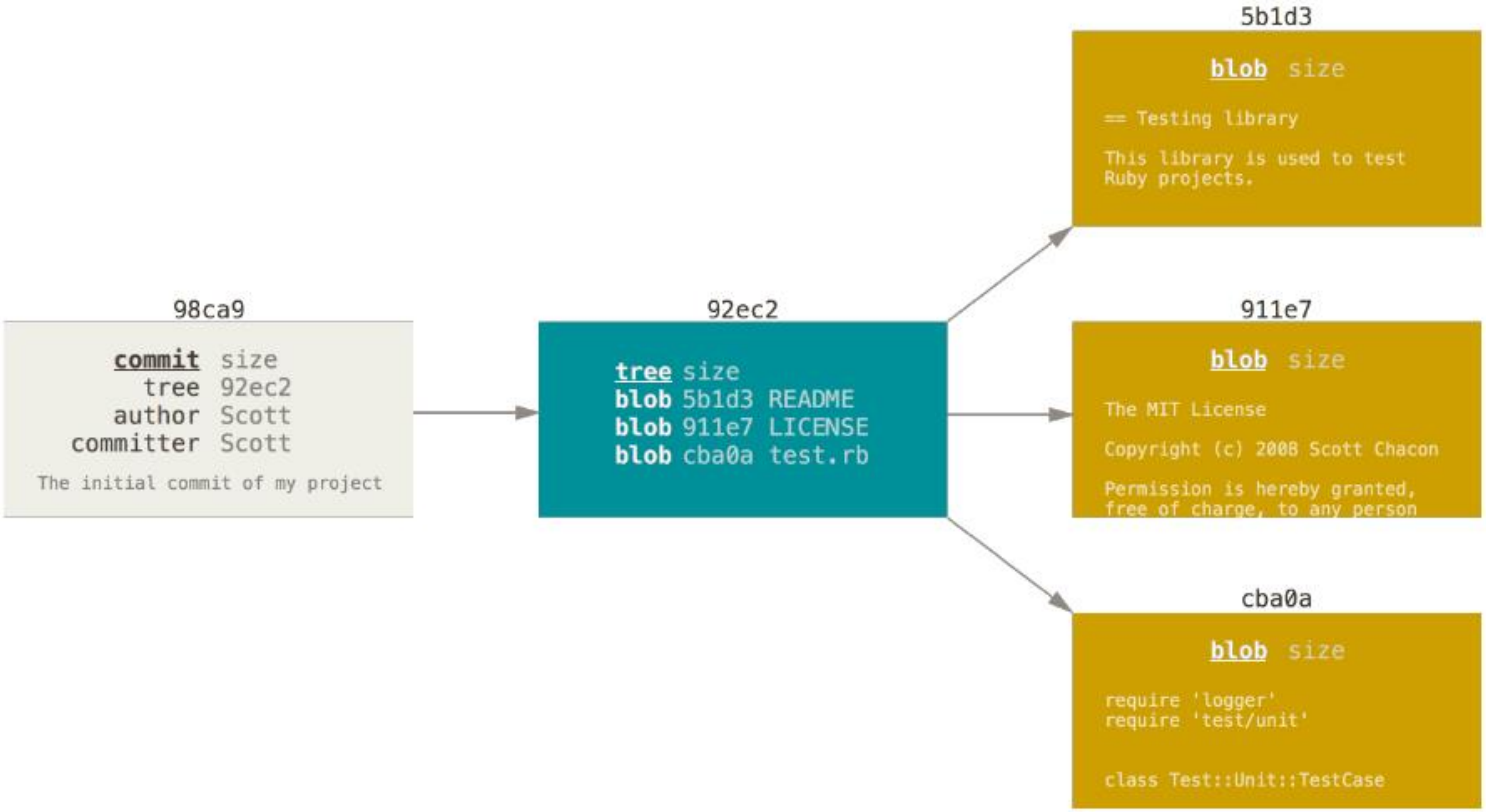
- Entire collection of files and folders, along with full history

Characteristics

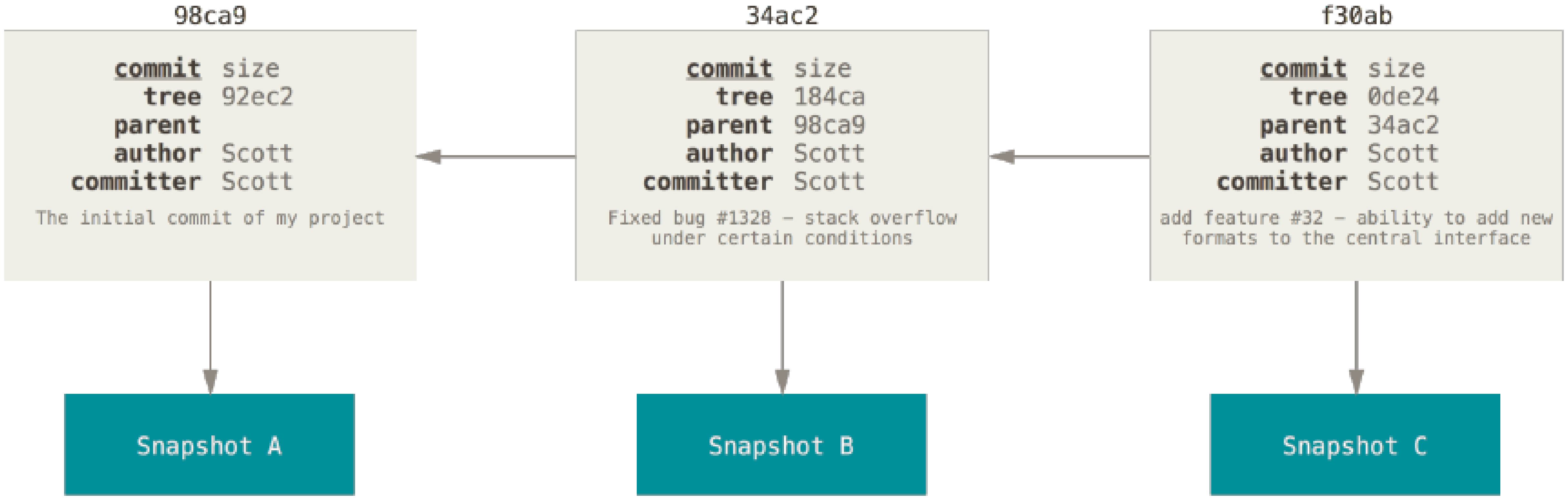
- Local & offline - Entire timeline/history of changes
- Snapshots, not deltas
- Changes are always additions

File history - snapshots in time called **commits**

# Git data structure – Commit, Tree, Blob



# Commit linked-list relationship



## Git Branches

### Exist at the repository level

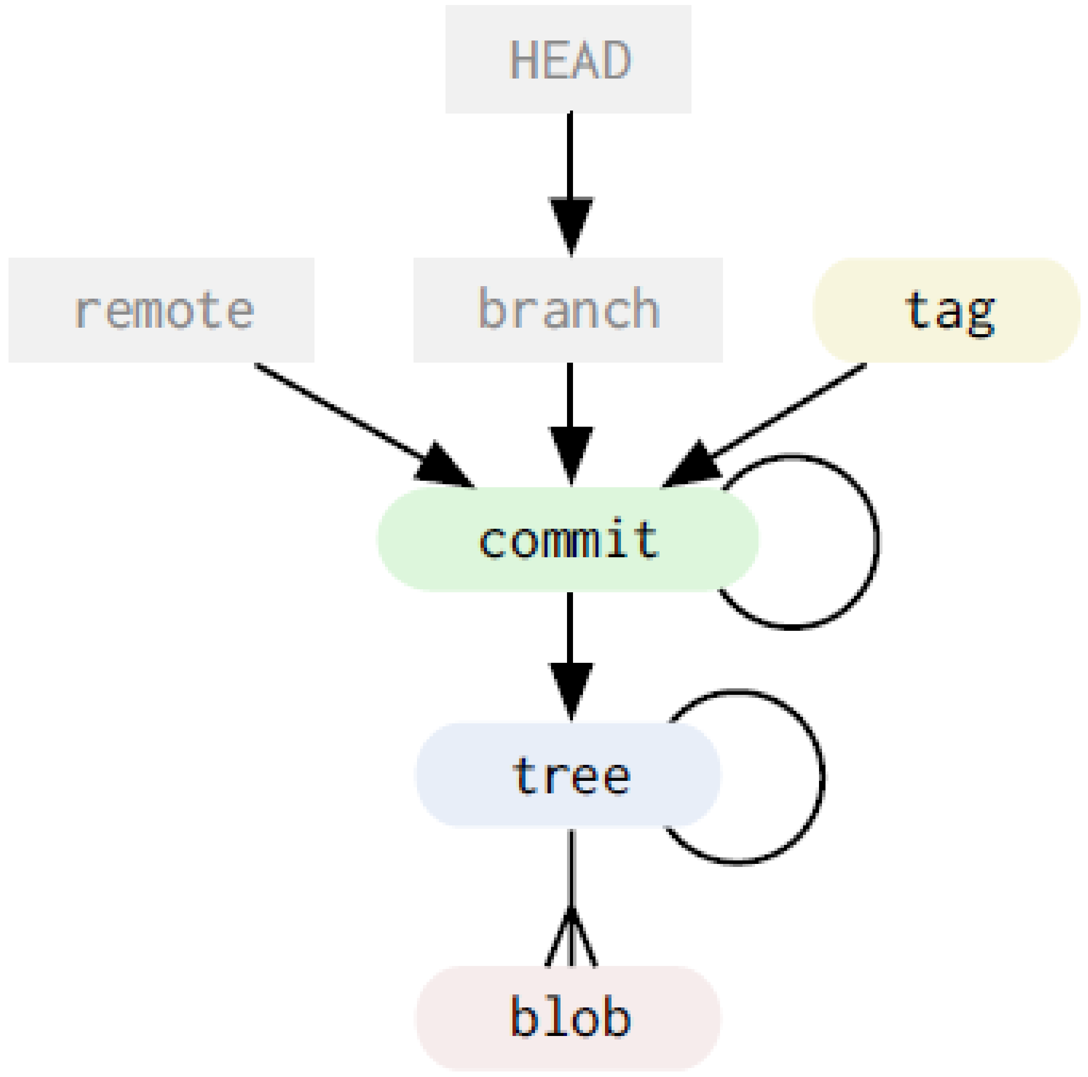
- A branch applies to the entire repository
- Unlike most centralized version control tools where branches exist inside the repository
- Default branch is ‘**Master**’
- Can be local or remote

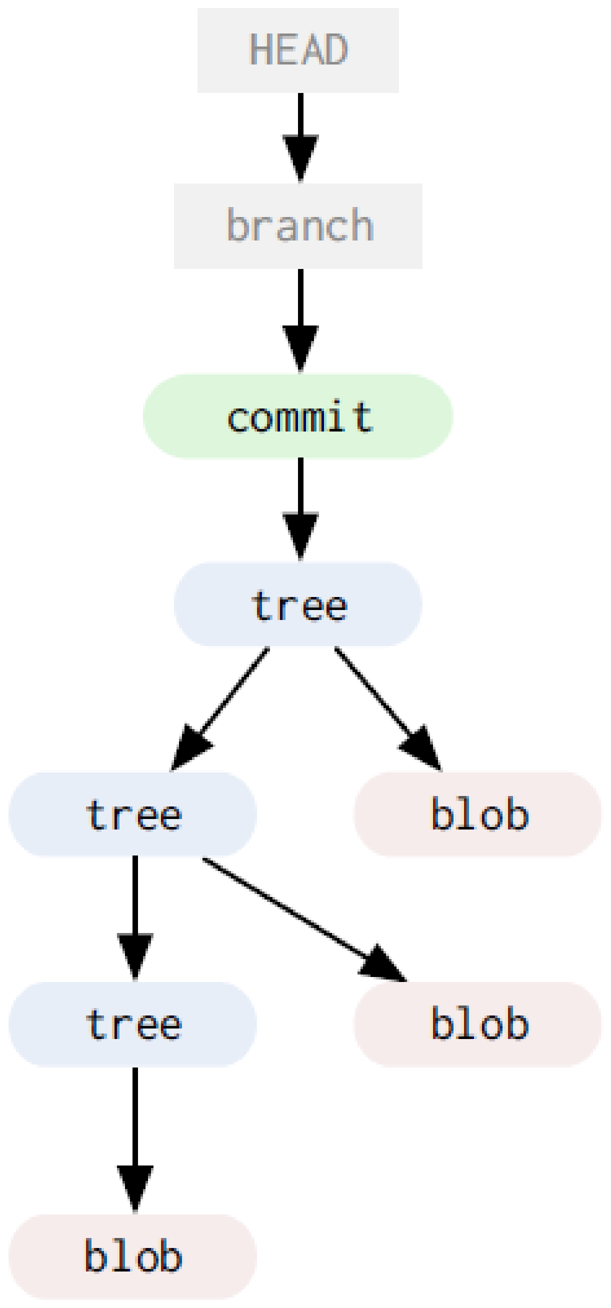
### Exceptionally lightweight

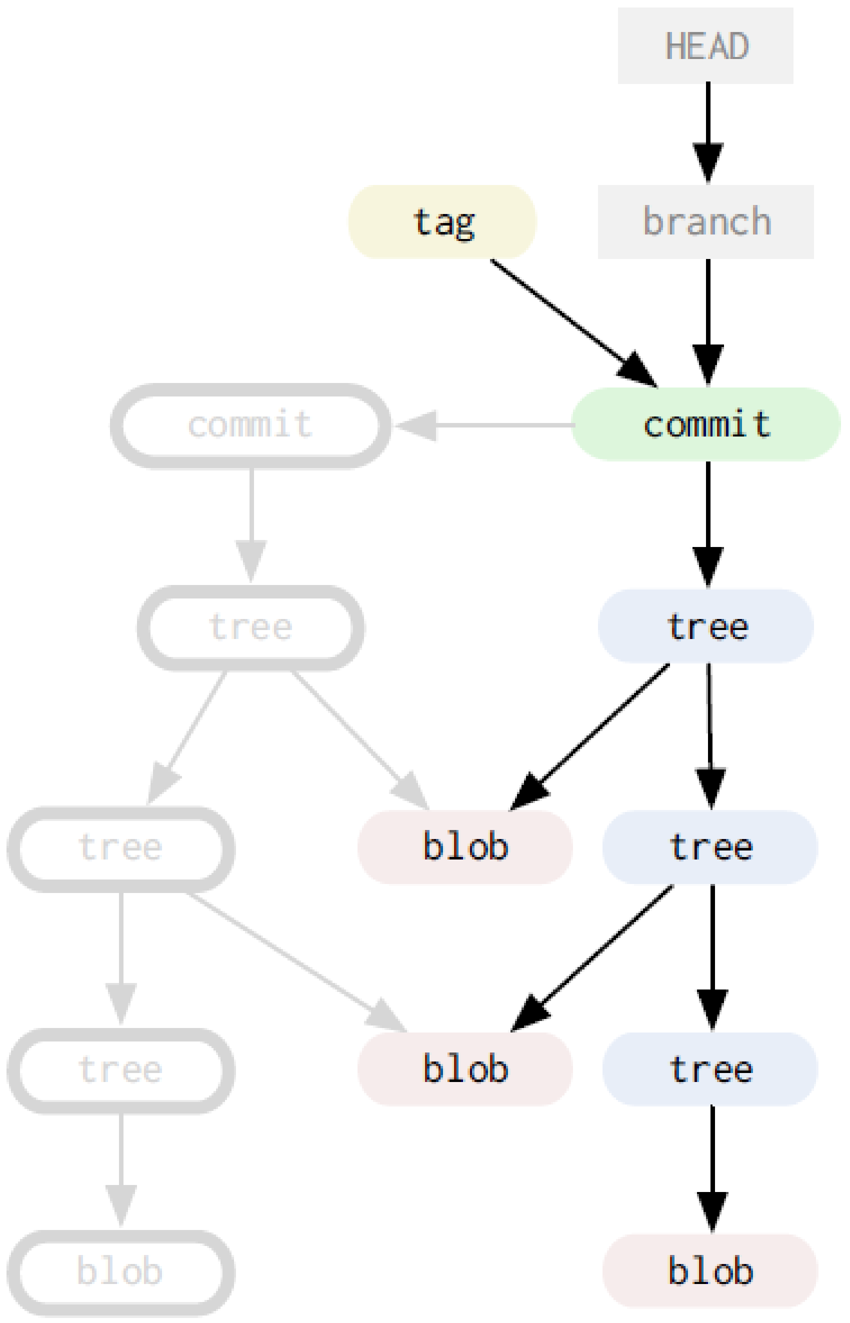
- Implemented as a pointer to a commit in the graph
- Exist only in the local repository until they’re explicitly shared
- Encourages feature branching

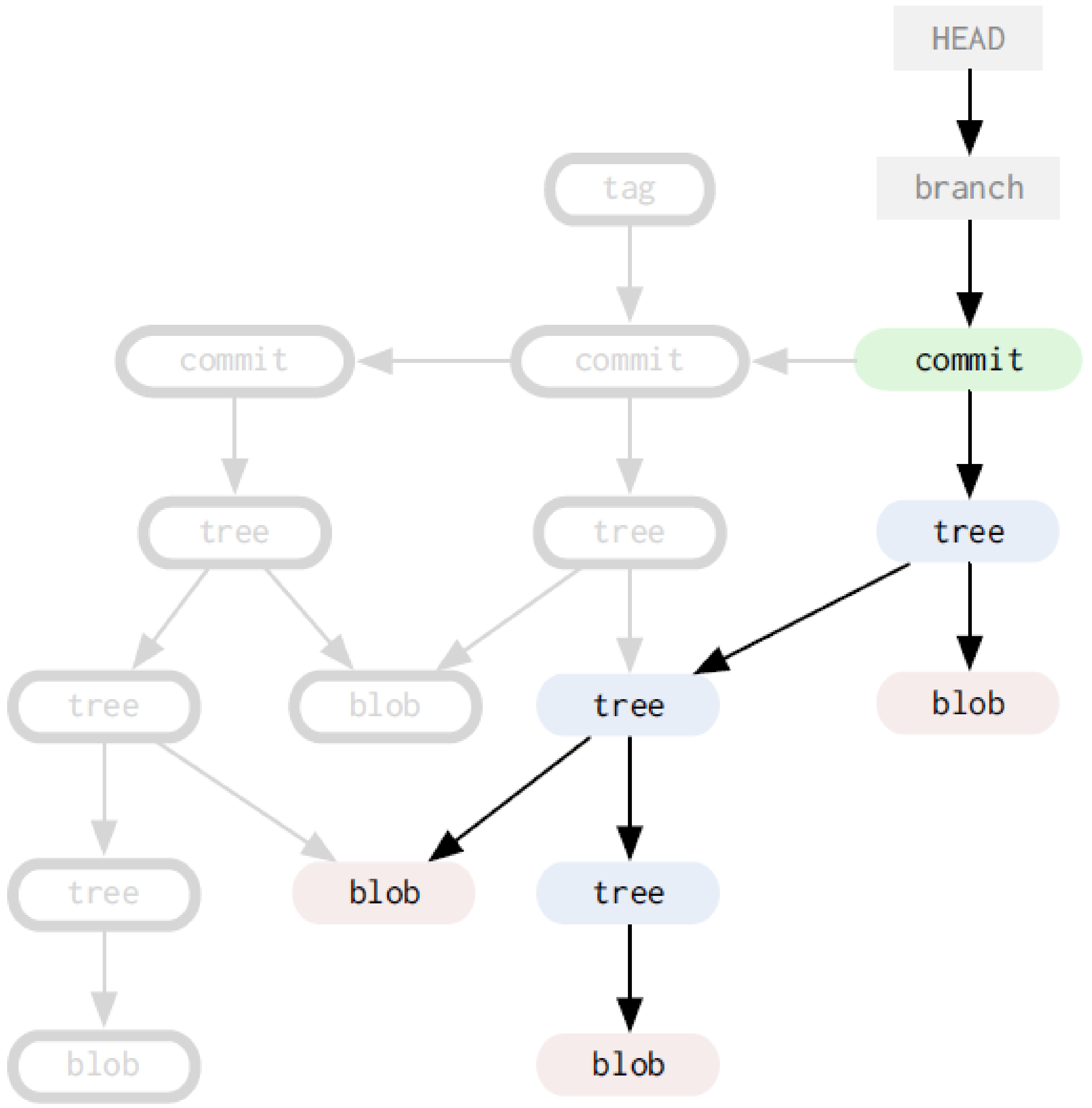
### **Checkout** – switch branch

## Git Data Model



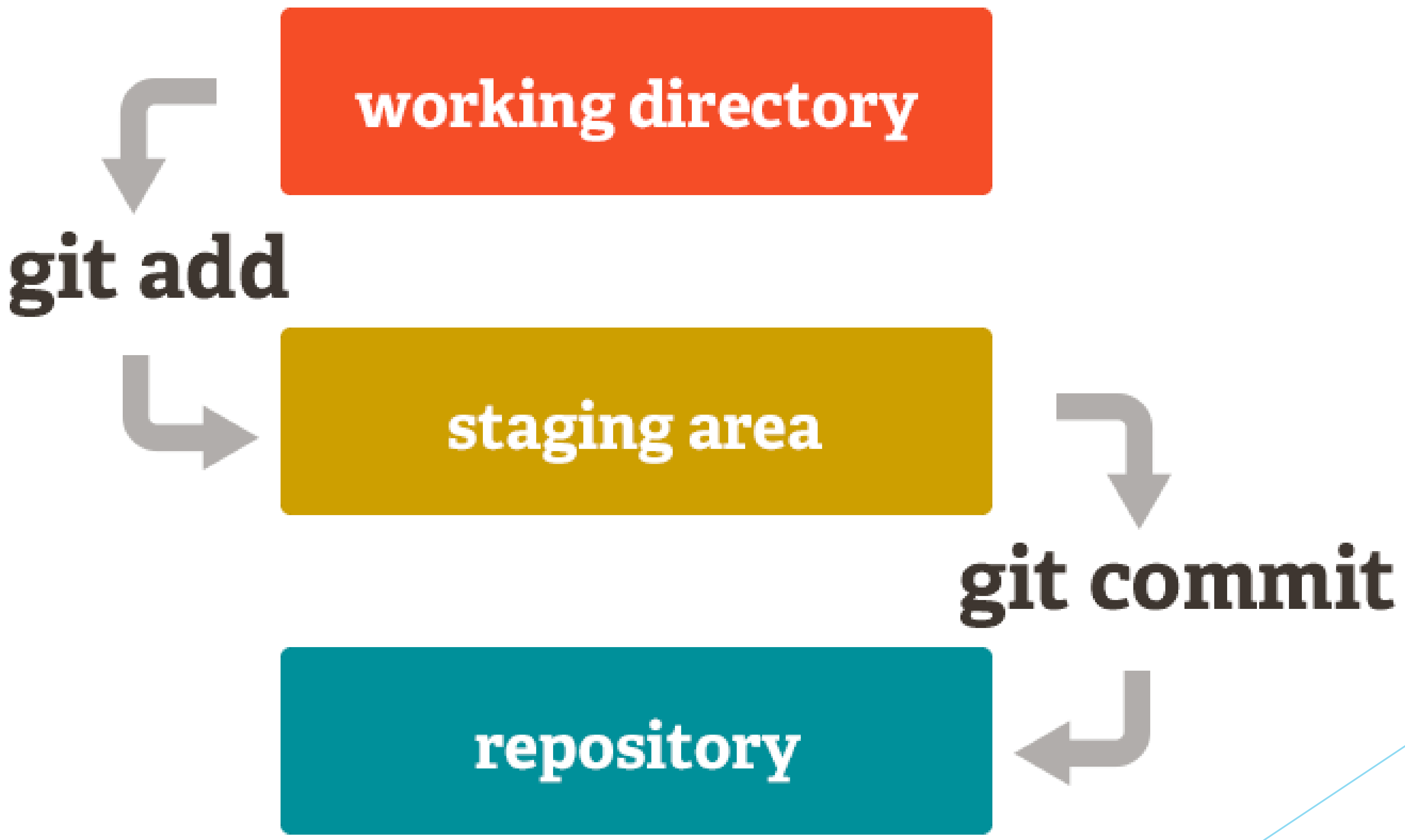

HEAD
branch
commit
tree
tree
blob
tree
blob
blob


HEAD
tag
branch
commit
commit
tree
tree
tree
blob
tree
tree
blob
blob
tree
blob
blob


HEAD
branch
tag
commit
commit
commit
tree
tree
tree
tree
blob
tree
blob
tree
blob
tree
blob
blob

# Git Two Stage Commit

**Working**/modified/workspace > **Staging** (index) > **Committed**/history/repo



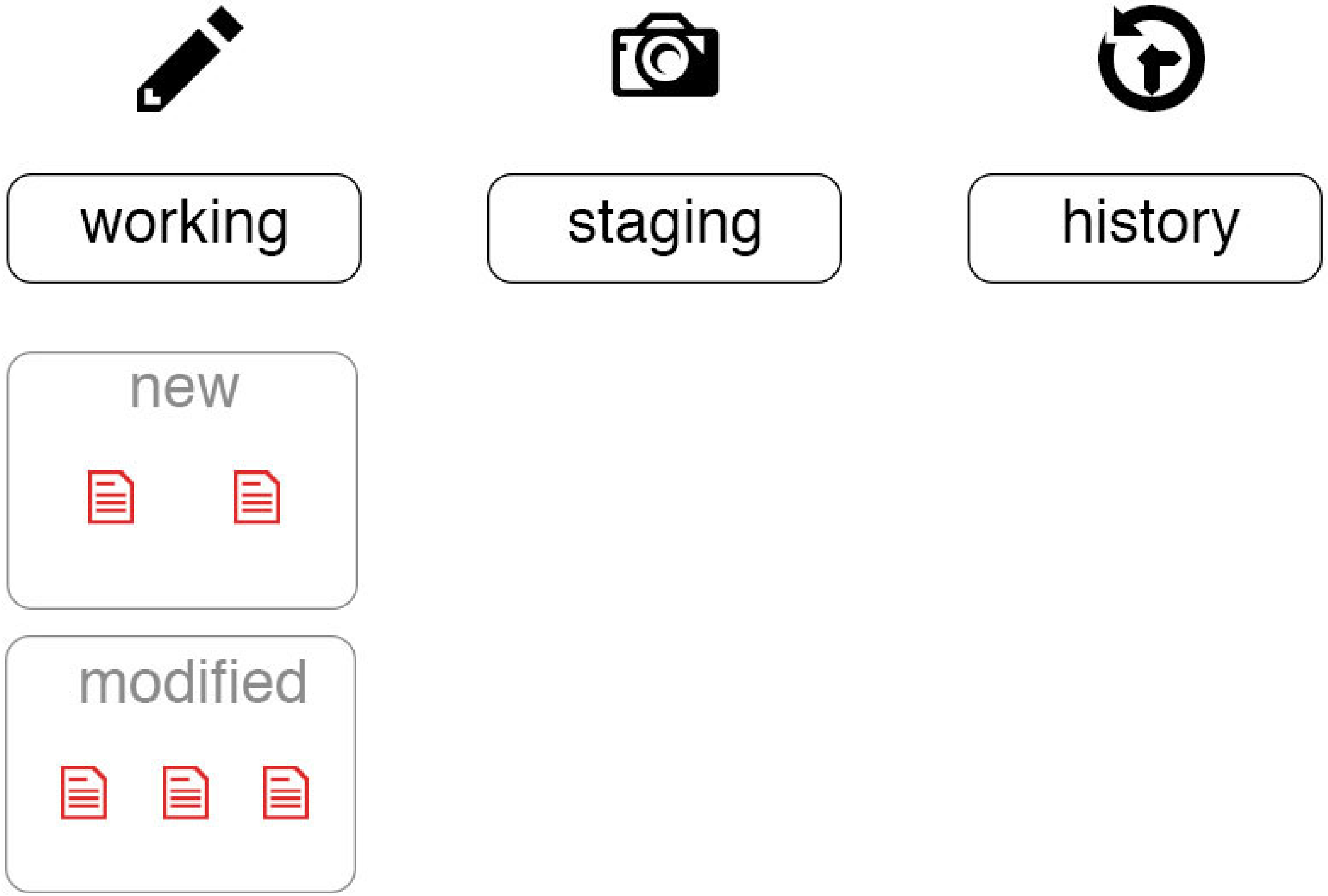
# Git File States

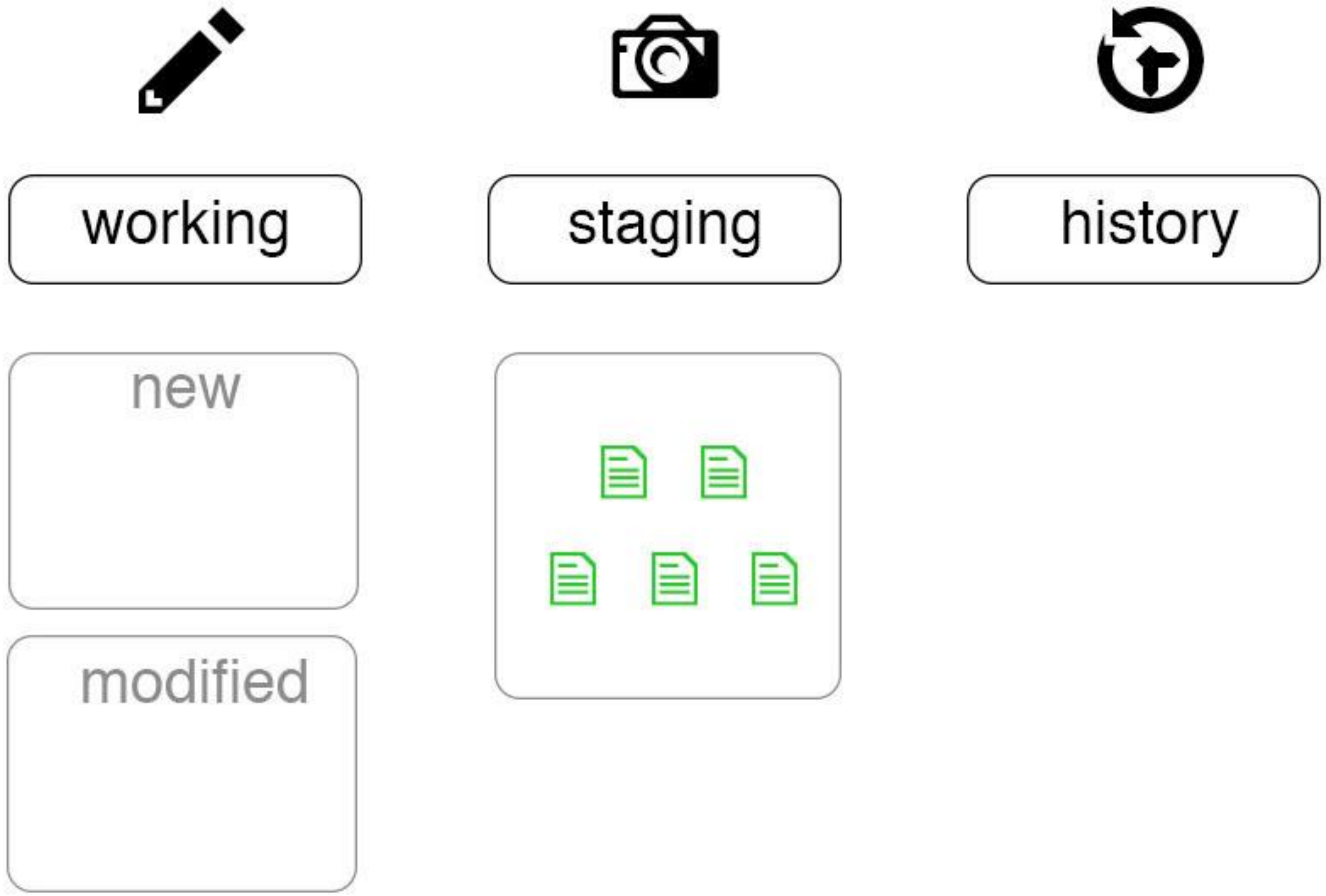
## Files exist in 1 of 4 states

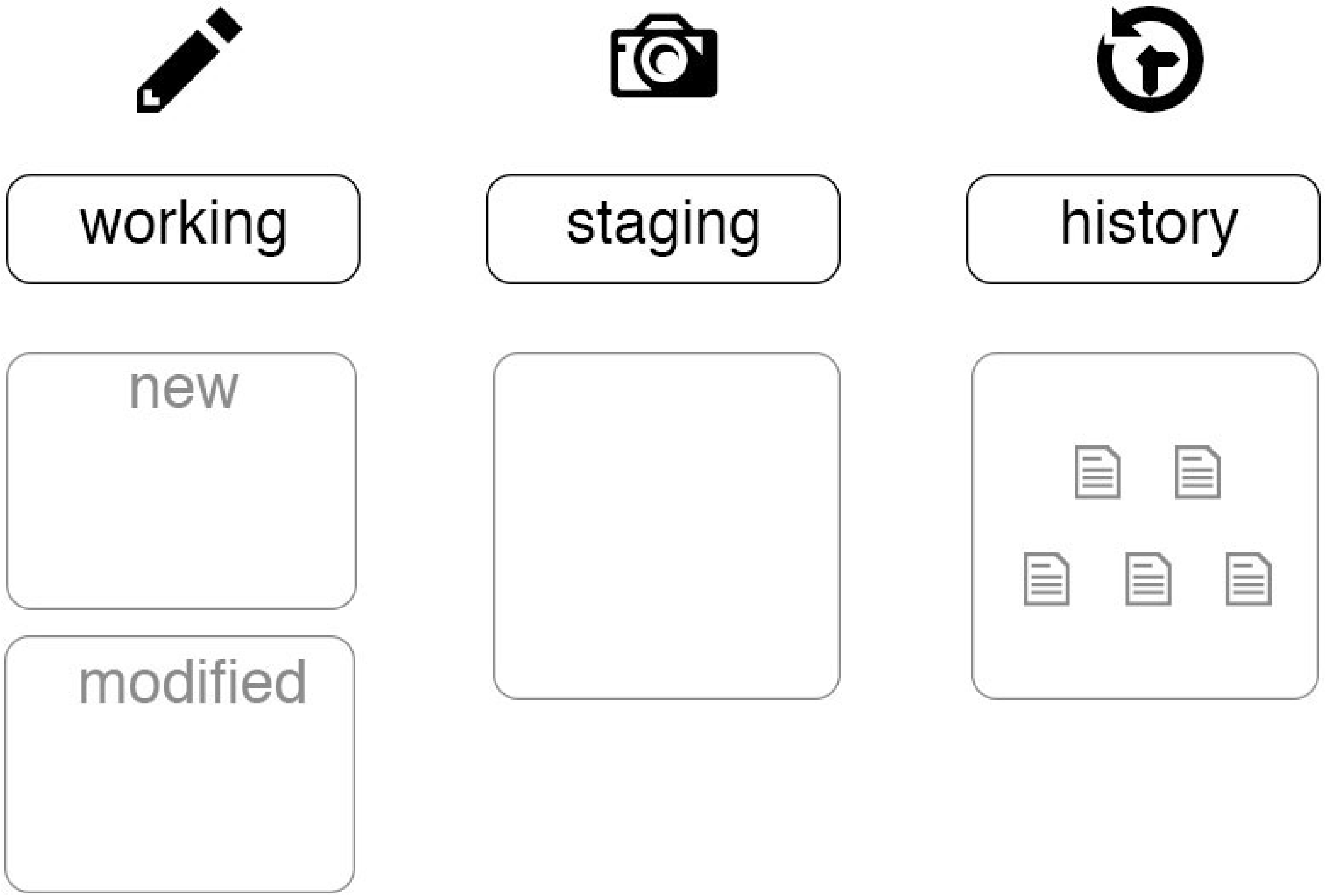
- Untracked, Modified, Staged or Committed
  - An untracked file is one that is not currently part of the version controlled directory

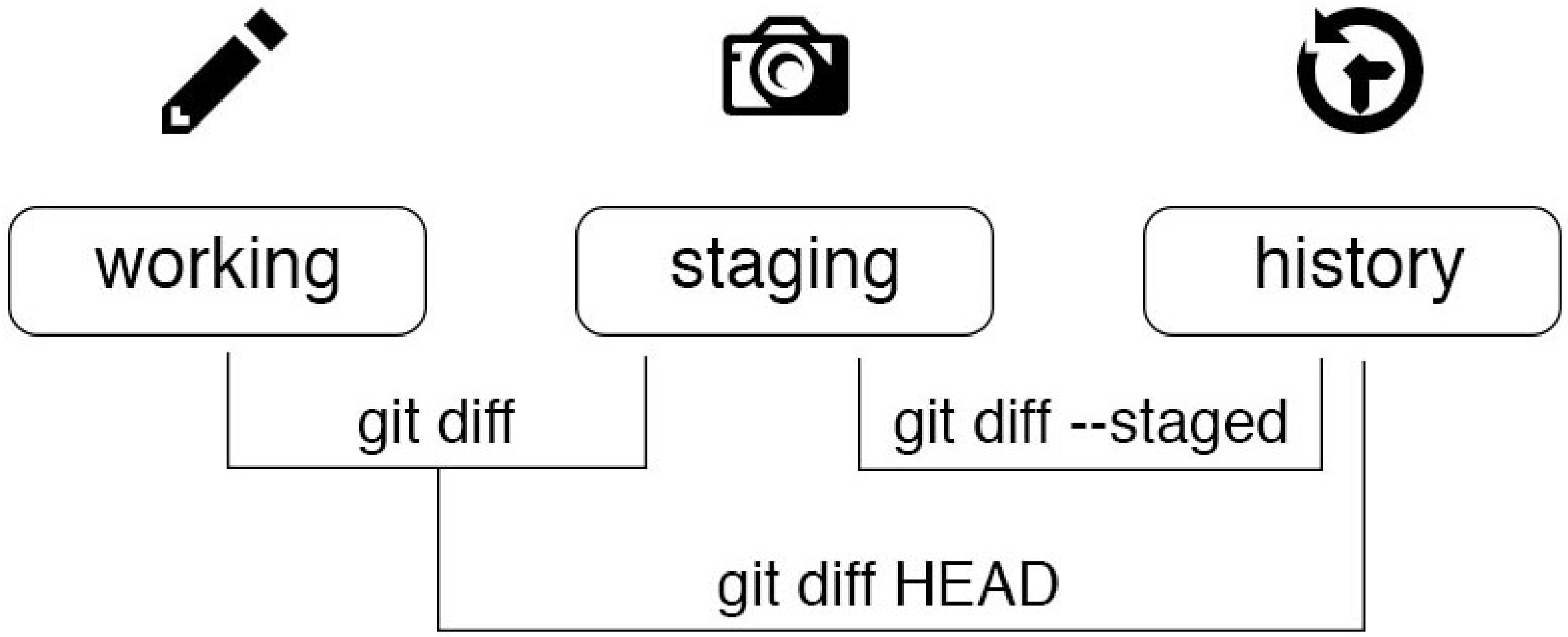
## Use git commands for adding, moving, renaming and removing/deleting files

- git add
- git mv (move and rename)
- git rm (remove)


working
staging
history
new
modified


working
staging
history
new
modified


working
staging
history
new
modified


working
staging
history
git diff
git diff --staged
git diff HEAD

## Git repo/project data

Git stores 3 copies of a project on your workstation.

- One copy is your own repository with your own **commit history**.
- The second copy is your **working copy** where you are editing and building.
    - State can be Working or Staged
- The third copy is your local "cached" copy of a **remote repository**.

Git saves space by storing file contents as unique, compressed blobs identified by a hash

# Git Remotes

Remotes - remote repository (internet or local network or local)

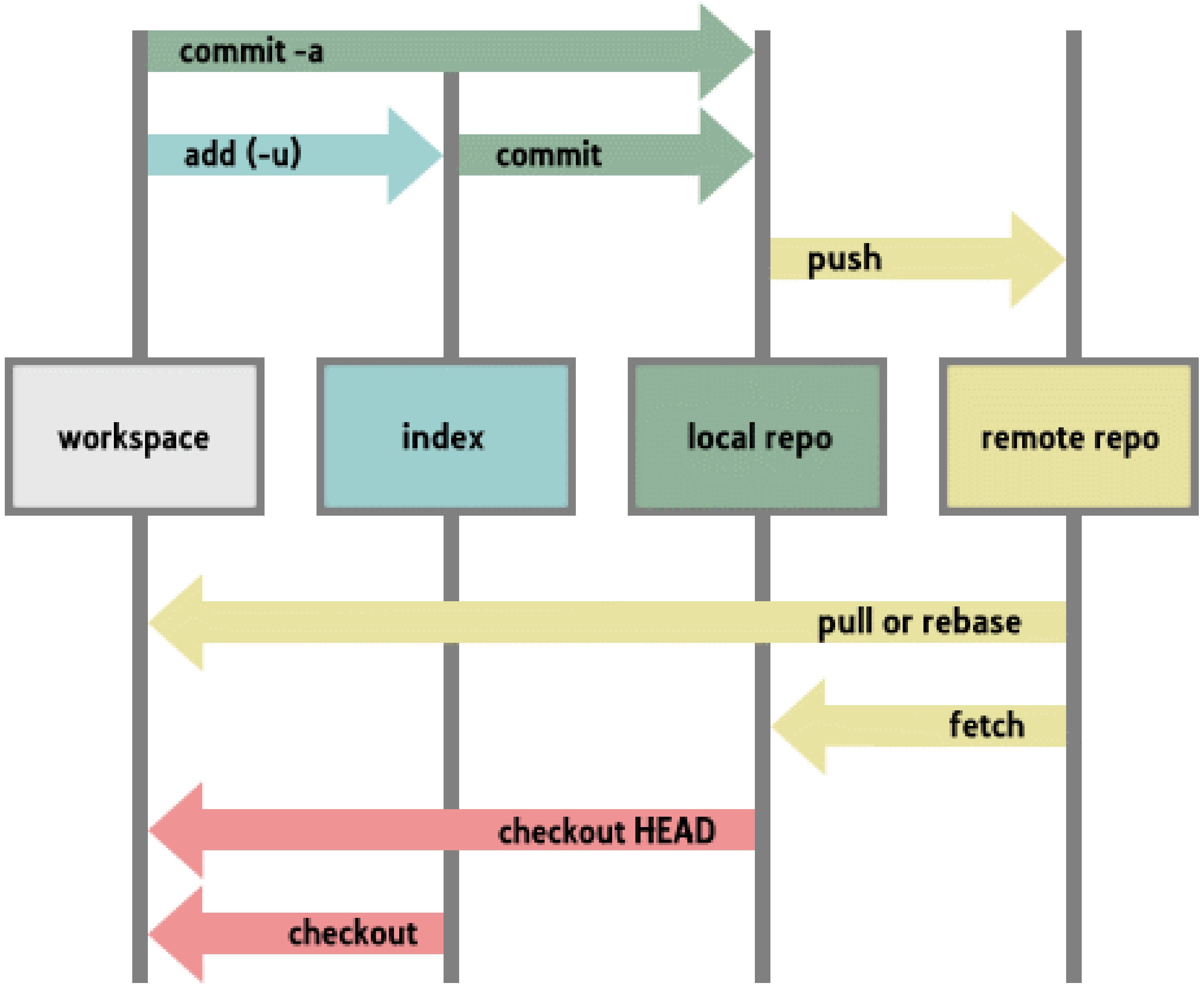
- **Origin** - default name for remote cloned from
- Merge Conflicts: local & remote

Commands

- **Fetch** - get changes from remote, updates remote-tracking branch
- **Pull** - combination of git fetch & git merge, updates current local branch
- **Push** - push changes to remote

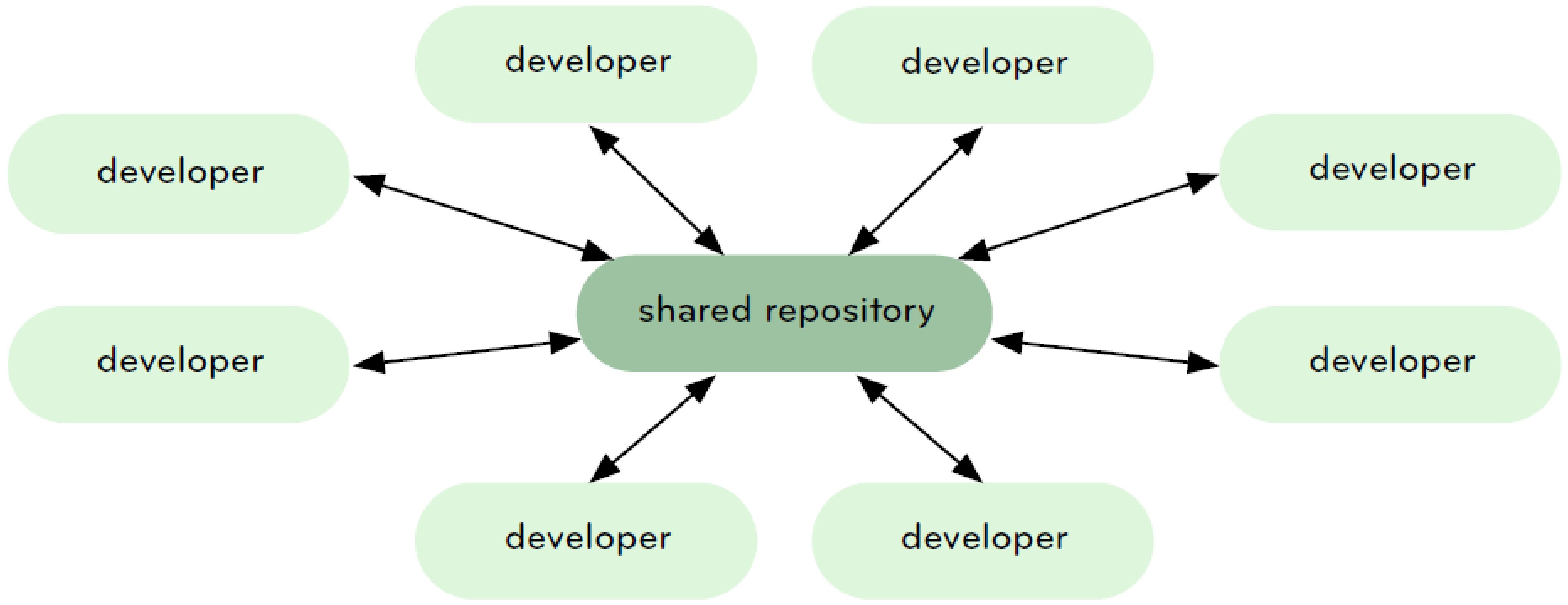
## Git Commands

- init, clone
- add, mv, rm, reset, clean
- checkout, commit
- status, log, show, grep, reflog
- branch, merge, diff, rebase, tag
- fetch, pull, push
- config, remote

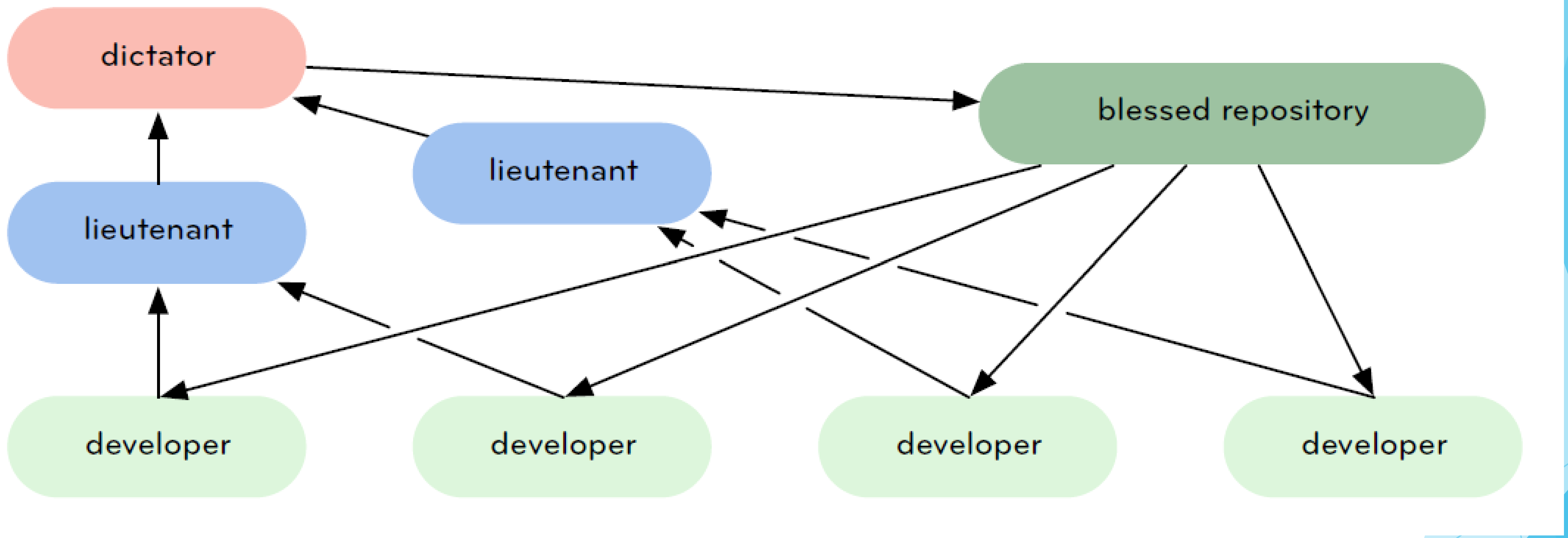

commit -a
add (-u)
commit
push
workspace
index
local repo
remote repo
pull or rebase
fetch
checkout HEAD
checkout

## Git Workflows

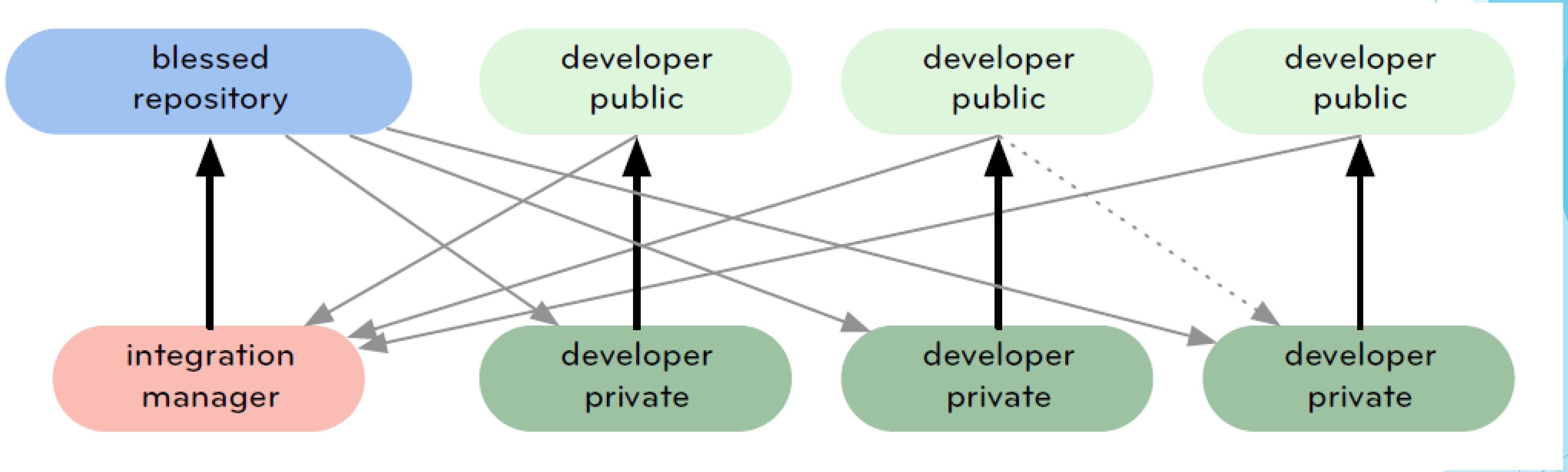
## Central Repository



## Dictator and Lieutenant Model



## Integration Manager



## Git terminology to know

- **head** - symbolic reference to the branch you're on. By default, there is a "head" in every repository called master.
  - HEAD - currently active head/branch
- **Tags** - pointer to a commit (like a label)
- **Reset** - rewrite history
- **Cherry-pick** - pick up a specific commit
- **Rebase** - reorder commits, edit them, squash multiple commits into one, etc.
  - Rebase vs. Merge (historical audit record vs. cleaned up record)

## Git Rebase warning:

Do not rebase commits that exist outside your repository.

## GitHub

## Github

### Online Service/Collaboration Platform

- Core function is an online git service
- Adds: issues, pull requests, code review, orgs & teams, pages, etc.

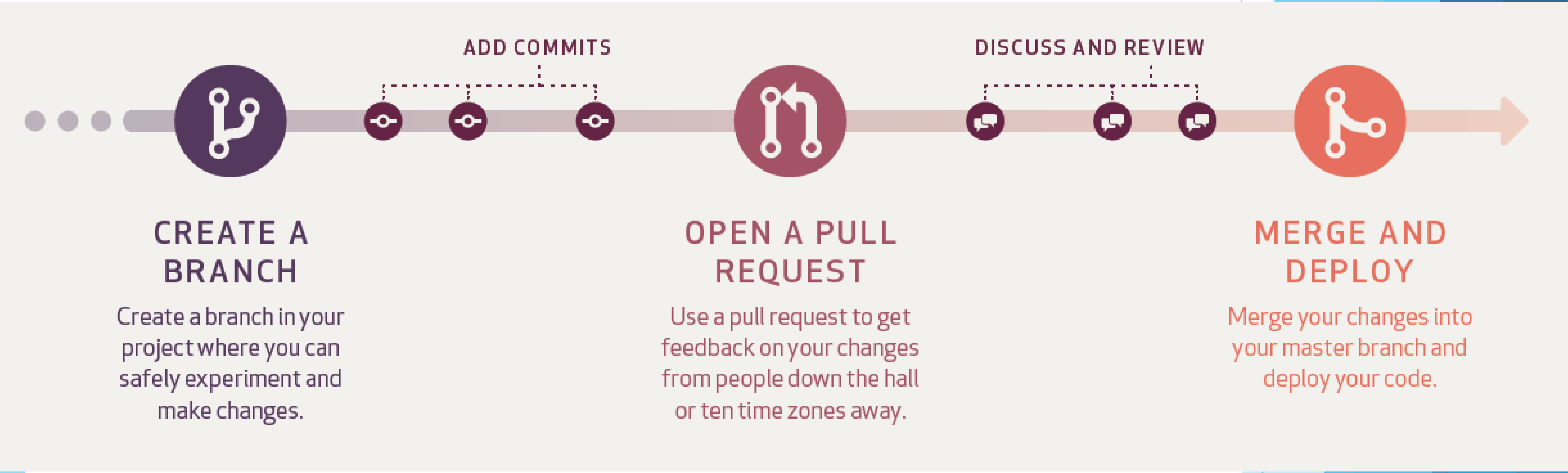
### Enhancements on top of Git

- **Fork -** copy of another repository (clone)
- **Pull Request (PR)** – code review & approval step

### Github Flow

- Create Branch, Add Commits, Open PR, Discuss & Review, Merge & Deploy

## GitHub Flow



CREATE A BRANCH

Create a branch in your project where you can safely experiment and make changes.

OPEN A PULL REQUEST

Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

MERGE AND DEPLOY

Merge your changes into your master branch and deploy your code.