

Intro to Git

Mark Turner - mark@amerine.net

Twitter: <http://twitter.com/amerine>

Code: <http://github.com/amerine>

Blog: <http://amerine.net>

About Me

- Dad
- Programmer
- Sys Admin
- Job: IT Director, Western Communications Inc.



SCM Concepts

What is SCM?

SCM Concepts

What is SCM?

S - Source

C - Code

M - Management

What do SCM's DO?

What do SCM's DO?

Track File Changes

What do SCM's DO?

Track File Changes

Hold File History

What do SCM's DO?

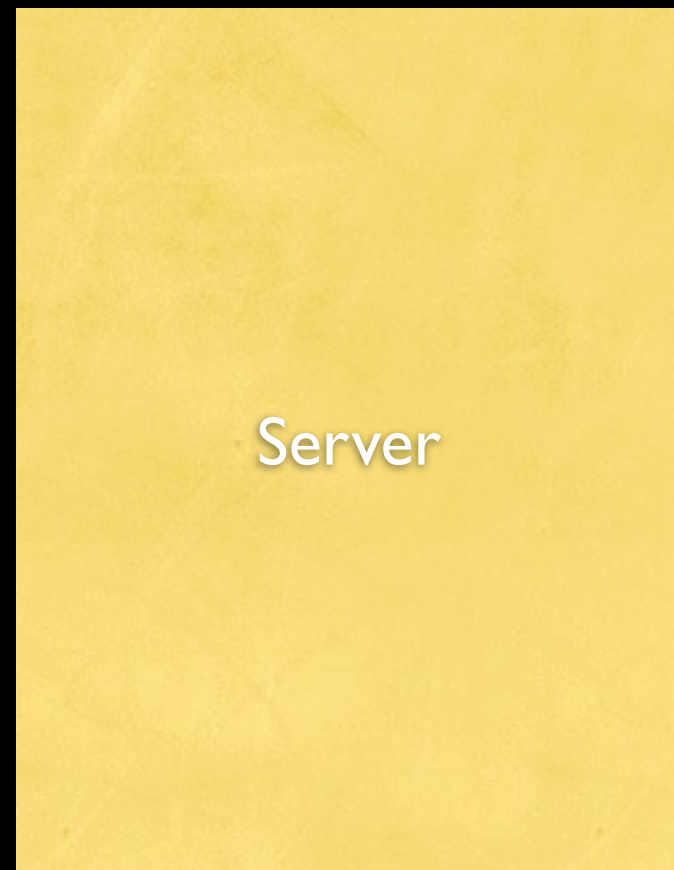
Track File Changes

Hold File History

Current State

There are two Kinds of SCM's

Centralized



Server

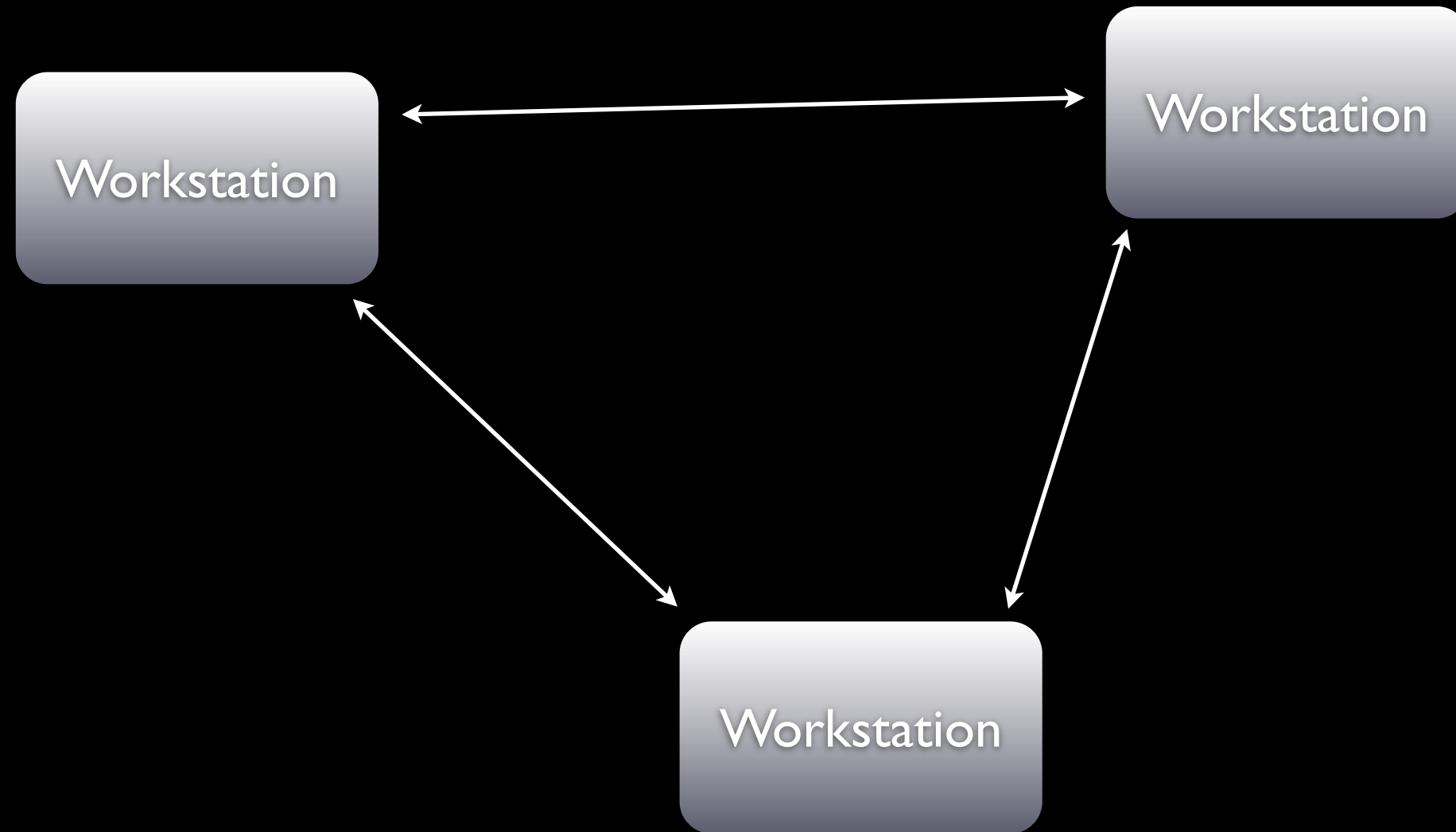
Repository



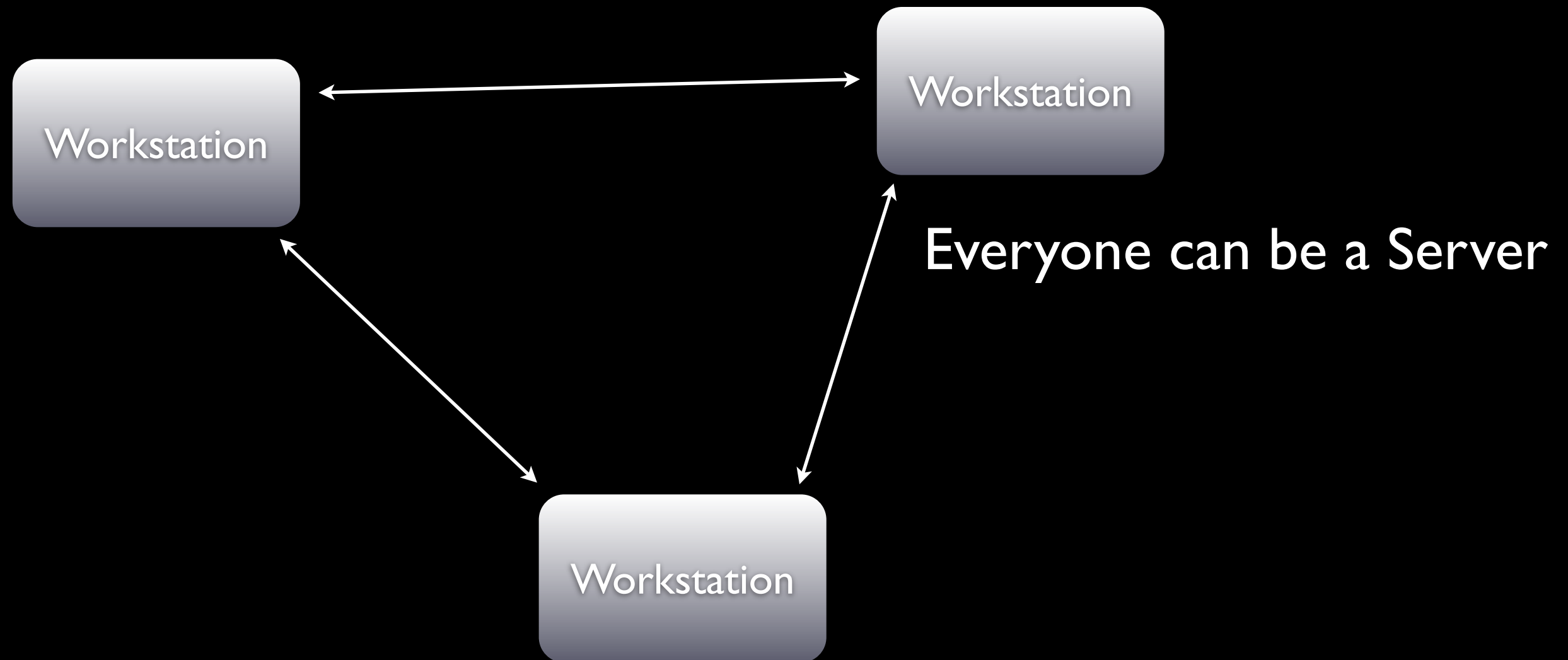
Workstation

Working Directory
State

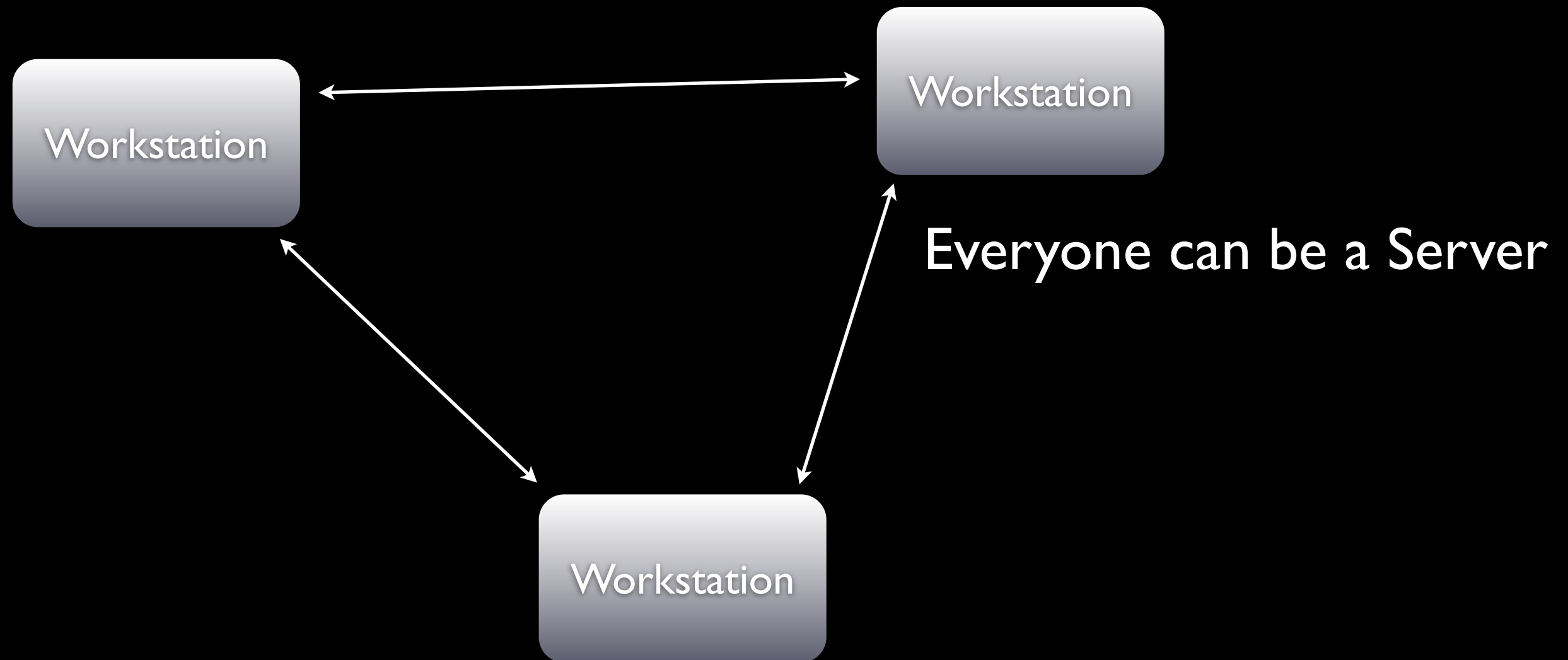
Distributed



Distributed

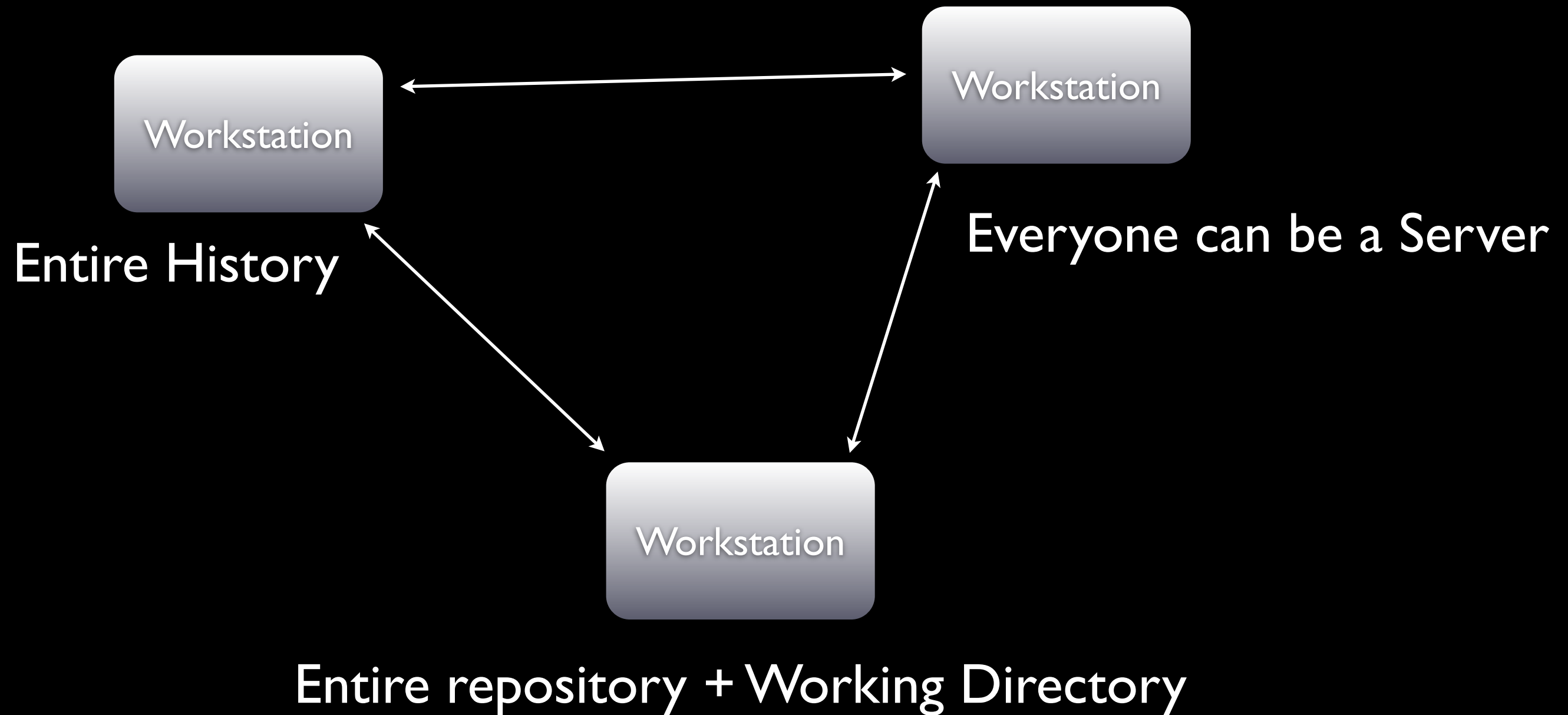


Distributed

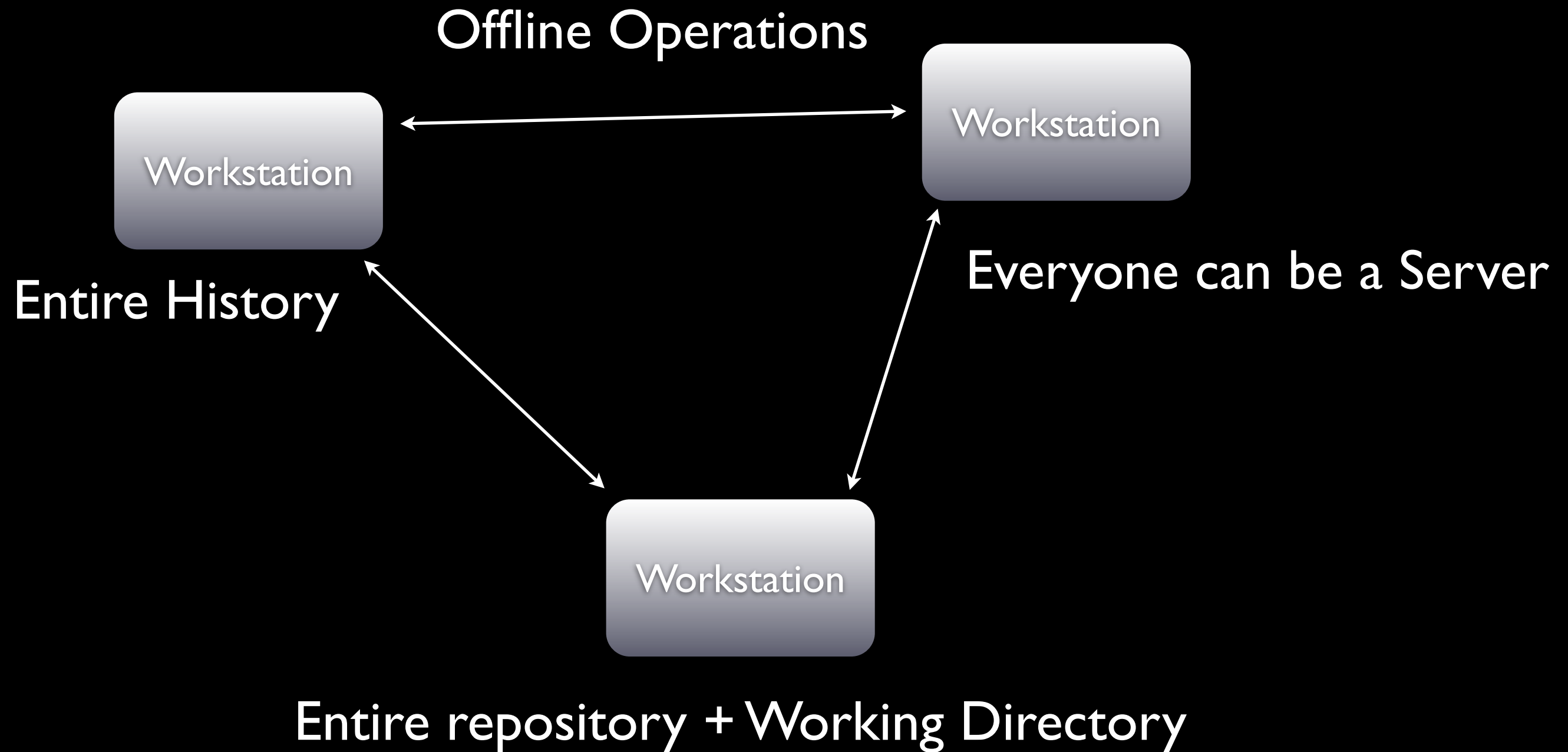


Entire repository + Working Directory

Distributed



Distributed



Repository Contents

- File
- Commits
- Ancestry

References

References

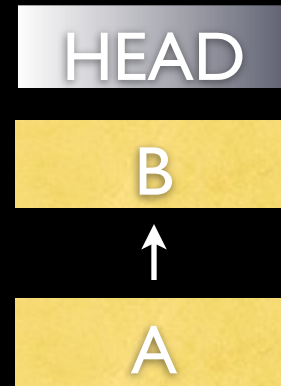
Tags

References

Tags

Branches

HEAD



Current Checkout
Points to Branch

Index

Staging Area

What's going to be committed.

SCM Operations

Bootstrap

init

checkout

change branches

Modify

delete
add
rename

commit

Info

status

log

diff

Reference

branch
tag

Centralized SCM's

Centralized SCM's

Most operations require a server

Centralized SCM's

Most operations require a server

This means you have a single point of failure

Centralized SCM's

Most operations require a server

This means you have a single point of failure

Potential cause of bottlenecks

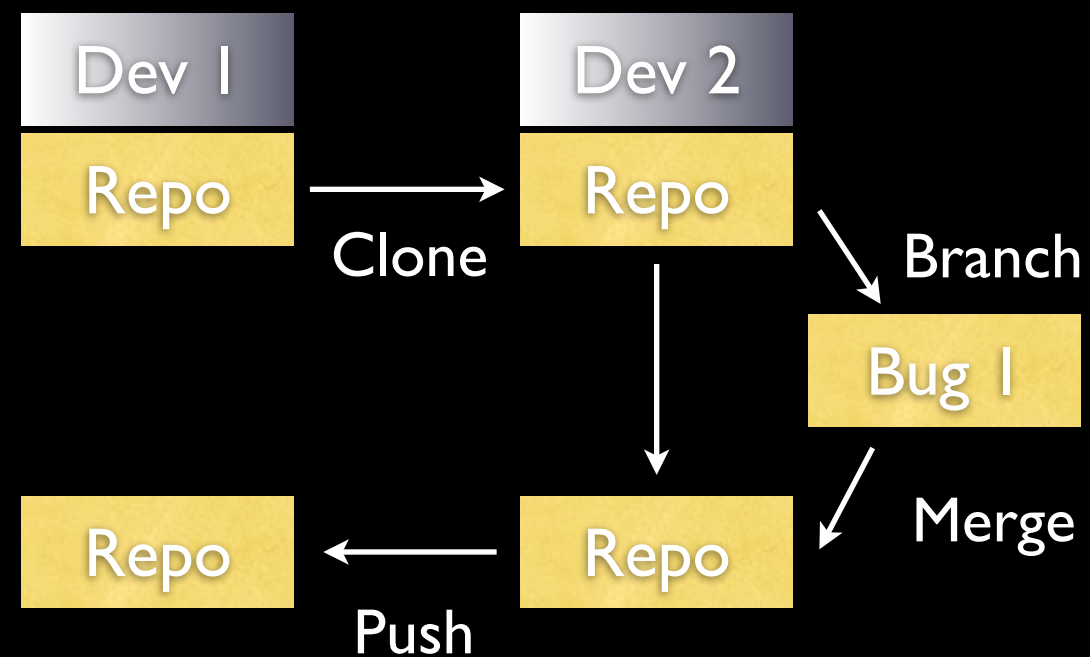
Distributed SCM's

Add the following

clone
pull
push

Anyone can be the server

Every system that has the repository can clone, push and pull between each other



Is Distributed worth it?

Is Distributed worth it?

- Non Intrusive Commits
- Detached Work
- No Single Points of Failure
- Easy, Cheap backups

Git

What is Git?

Git

What is Git?

Git is a distributed source code management system.

Git History

- 2002 - Linus is using BitKeeper (from BitMover) for tracking the kernel. The world was full of Rainbows and Unicorns.
- April 5th, 2005 - BitMover drops free license. Linus begins working on git.
- April 18th, 2005 - Git Can Merge
- June 16th, 2005 - Git officially tracks the kernel
- Feb 2007 - 1.5.0 Released.
- Today - 1.6.X Series in use

Highlights

- Distributed
- Fast
- Space Conscious
- Efficient protocol
- No special web server requirements.

Development Support

Git has great support for quick branching and merging, and it includes great tools for visualizing and navigating your history.

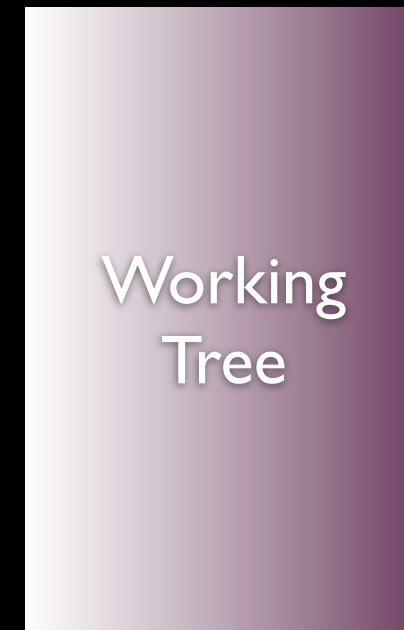
Efficient

Git is very fast, especially for large projects with long histories.

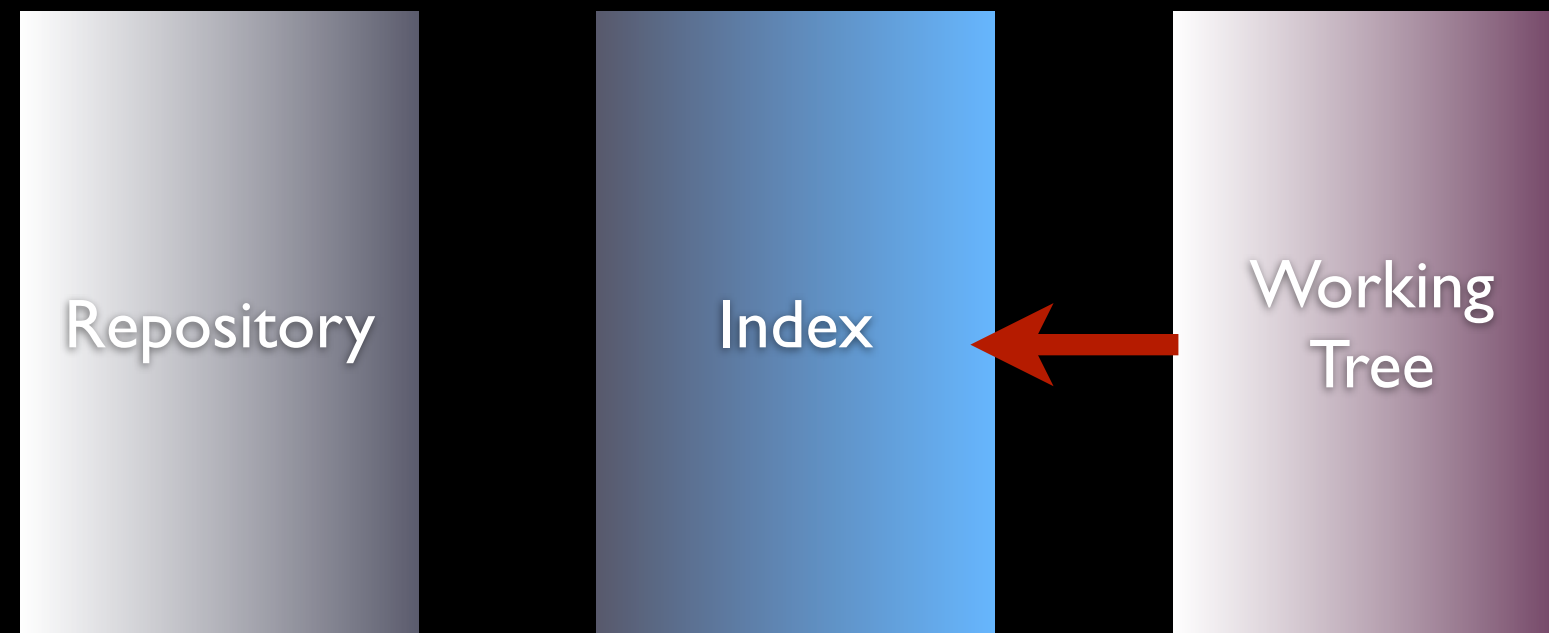
Design

Git follows the unix tradition of many small tools written in c, with wrapper scripts that make it easy to use.

Git Repository Structure



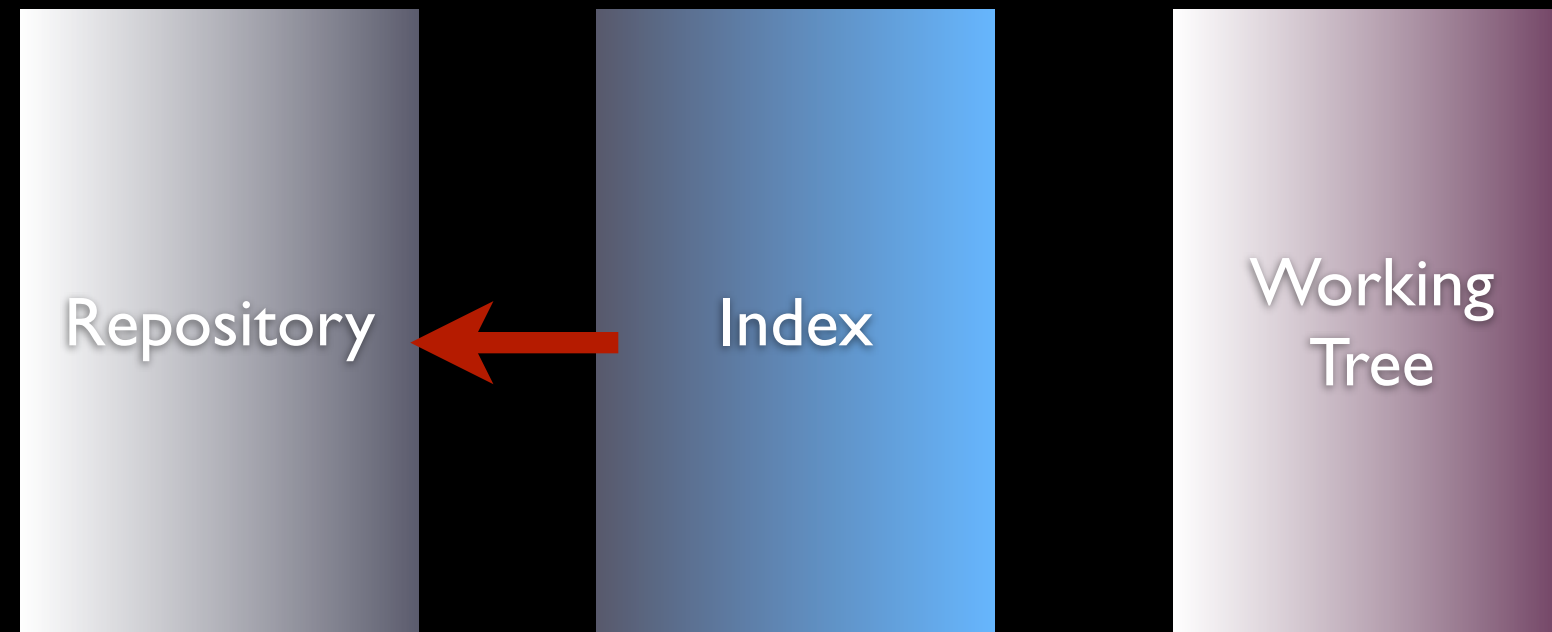
Git Repository Structure



The operations that allows you to move files from the working tree to the index "Staging" are things like:

add
remove
rename

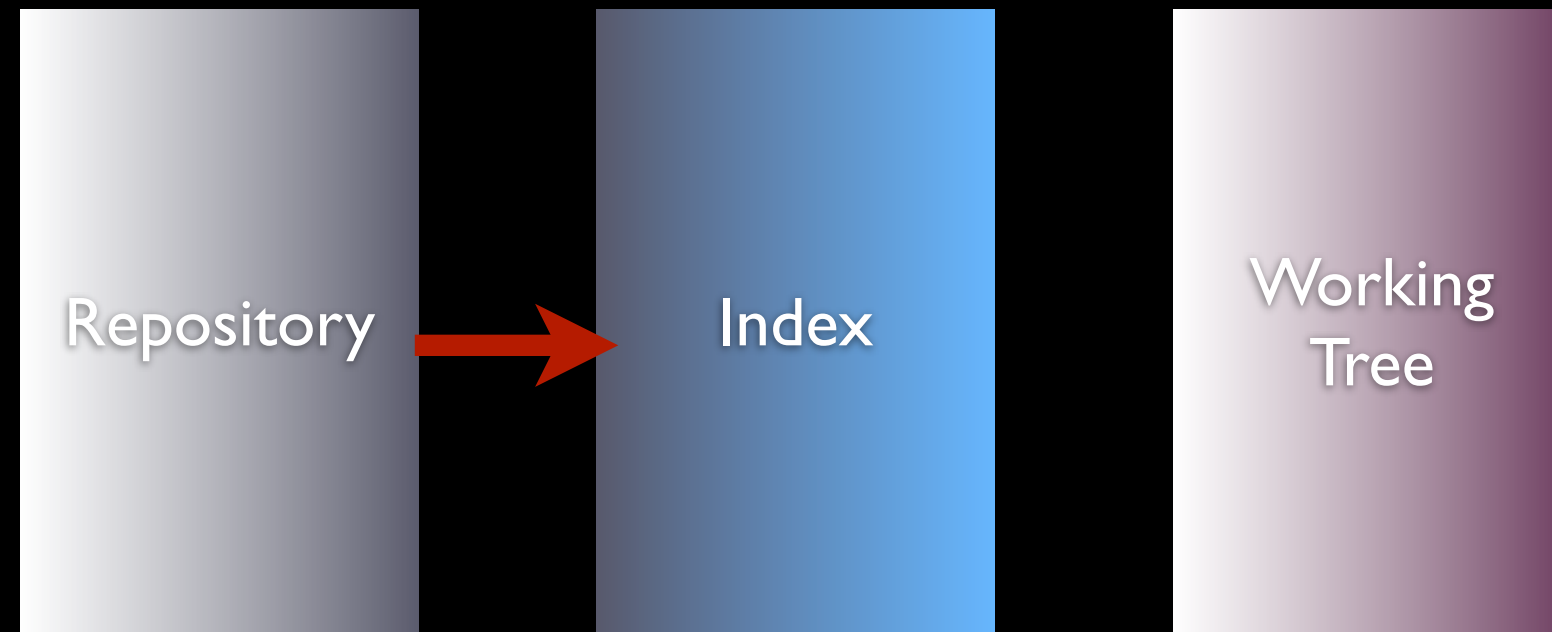
Git Repository Structure



You move the index
into the Repository
via commits

commit

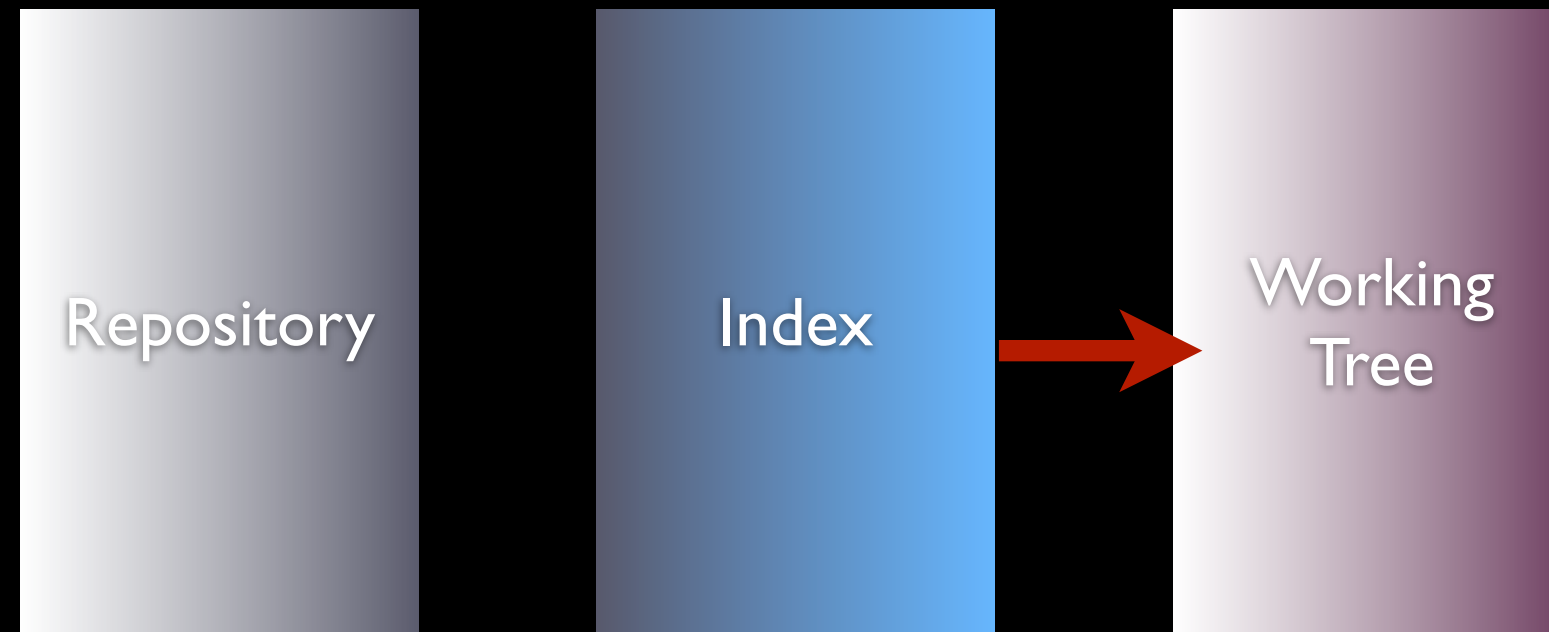
Git Repository Structure



To read a tree into an
index

checkout
read-tree
reset

Git Repository Structure



To read the index to
the working directory

checkout
reset

.git Directory Structure

Important Files

`.git/config` - Repository Local config

`.git/description` - Describes the Repository

`.git/info/exclude` - Items you want to ignore

Objects

Loose Objects

`.git/objects`

`|--23`

`| -- 4be6 | | 28eef7 | 3459ca4e32398d689fe80864e`

Objects

Loose Objects

`.git/objects`

`|--23`

`| -- 4be6 | | 28eef7 | 3459ca4e32398d689fe80864e`

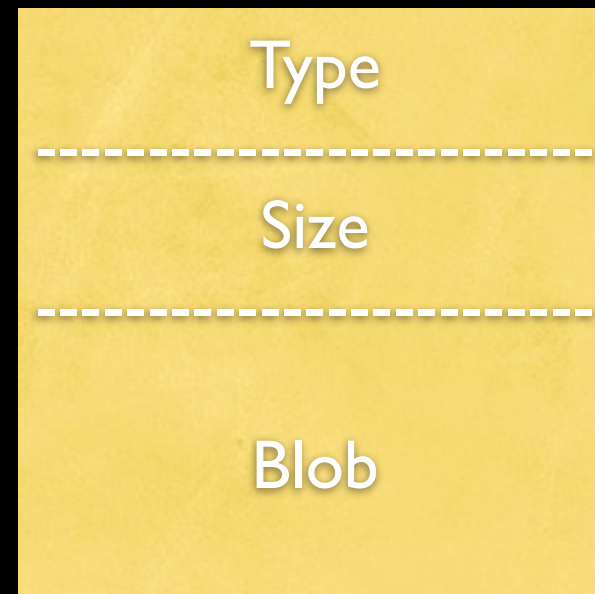
Pack Files

Pack files (gzipped)

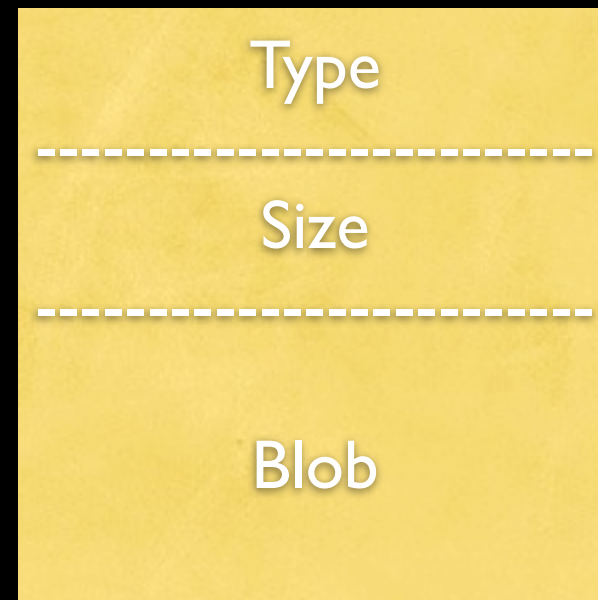
`|--pack`

`| |-- pack-b7ad43h3asd0dab702b | cc...idx`

What are Objects?



What are Objects?



bddf69f3f66a47705cac97bc0dd50acd91d8b9ba

4 Types of Objects

- File
- Trees
- Commits
- Tags

Objects are Immutable

Installing Git

Linux - install git-core via packaging system

OS X - Use the git installer (google it)

Windows - Use Linux

Just Kidding. Use Msysgit @ <http://code.google.com/p/msysgit/>

Using Git

Using Git

Everything starts with git

Using Git

Everything starts with git

There are over 100 commands

Using Git

Everything starts with git

There are over 100 commands

For everyday use there are maybe 20 you'll use

Using Git

Everything starts with git

There are over 100 commands

For everyday use there are maybe 20 you'll use

There are some GUI commands

Getting Help

```
$ git help
```

You can get a list of common commands

```
$ git help git - Basic git intro
```

Getting Help

```
$ git help
```

You can get a list of common commands

```
$ git <command> -h
```

Every command supports help that will include information on its usage

```
$ git help git - Basic git intro
```

Getting Help

```
$ git help
```

You can get a list of common commands

```
$ git help git - Basic git intro
```

Getting Help

```
$ git help git - Basic git intro
```

Getting Help

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

All man pages

```
$ git help git - Basic git intro
```

Configuration Files

Global: `$HOME/.gitconfig`

System: `/etc/gitconfig`

Per Repo: `/path/to/repo/.git/config`

Configuring Git

You can edit the config with an editor or the git config command.

```
$ git config --global user.name "Your Name"  
$ git config --global user.email your@email.tld  
  
$ git config --global color.pager true  
$ git config --global color.ui auto
```

What you should configure

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email your@email.tld
```

```
$ git config --global color.pager true
```

```
$ git config --global color.ui auto
```

Creating a new Repo

```
$ git init
```

Run this in your working directory

Staging

Additions:

```
$ git add <file>  
$ git add .
```

Staging

Removal:

```
$ git rm <file>
```

Staging

Rename:

```
$ git mv <file1> <file2>
```

Committing

```
$ git commit -m "Initial Commit"
```

Commits the staged files

```
$ git commit -a -m "Commit Message"
```

Will add and commit all the files in the working directory that git is tracking.

Status

```
$ git status
```

Diff

Diff

```
$ git diff
```

Changes between index and working files

Diff

```
$ git diff
```

Changes between index and working files

```
$ git diff --staged
```

Changes between HEAD and index

Diff

```
$ git diff
```

Changes between index and working files

```
$ git diff --staged
```

Changes between HEAD and index

```
$ git diff HEAD
```

Changes between HEAD and working files

Diff

```
$ git diff
```

Changes between index and working files

```
$ git diff --staged
```

Changes between HEAD and index

```
$ git diff HEAD
```

Changes between HEAD and working files

```
$ git diff commit commit
```

Changes between commits.

Object References

full hash	6bb1276bb1276bb1276bb1276bb1276bb1276bb1276bb127
short hash	6bb127
tag	v1.3.2
local branch	master
remote branch	origin/master
message	"/text"
checkout	HEAD
last_fetch	FETCH_HEAD
previous head	ORIG_HEAD

Object References

HEAD^ == HEAD~1

HEAD^^ == HEAD~2

HEAD^^^ == HEAD~3

Object References

What was the repo like yesterday?

`{yesterday}` or `HEAD@{yesterday}`

Object References

What was the repo like June 1st?

`{June.1}` or `HEAD@{June.1}`

Object References

What did another branch look like on July 20th 2008?

`another_branch@{July.20.2008}`

Object References

What did master look like 4 commits ago?

```
master@{4}
```

Looking at the Repo

```
$ git show
```

Shows last commit

Looking at the Repo

```
$ git show --stat
```

Short summary of the commits.

Looking at the Repo

```
$ git show --name-status
```

SVN like status. Shows A, M, D's

Looking at the Repo

```
$ git show HEAD
```

```
$ git show HEAD^^^
```

```
$ git show master~5
```

```
$ git show master@{May.16.2008}
```

Looking at the Repo

```
$ git show HEAD:file
```

Shows a file at a particular commit

Git Log

```
$ git log
```

Git log is fantastic

Git Log

```
$ git log tag..branch
```

Specify a range, from a tag to branch

```
$ git log HEAD~10..
```

Opposite of this commit all the way to head. Head is implicit.

```
$ git log branch1 branch2 ^common
```

Show everything between two branches and stops at the common point.

Git Log

```
$ git log -10
```

Shows the commits from 10 commits ago to HEAD

```
$ git log -10 master{@yesterday}
```

Shows the commits from 10 commits ago to master yesterday

```
$ git log --since="--May 1" --until="--June 1"
```

Shows the commits between two dates.

Git Log

```
$ git log --author=mark
```

Filter all commits by author

```
$ git log --committer=joe
```

Filter all commits by joe as the committer

Git Log

```
$ git log --grep="commit.*message.*text"
```

Grep the commit message

```
$ git log -S"Some Code"
```

We can also search through the commit code. We can look for method foo and find

Git Log

```
$ git log -- some/file
```

Only show commits that modify a particular file

References

Things that point to commits (or trees)

Lightweight
mutable
disposable
3 basic types

Branches

A very important part of git workflows.
branch all the time, for any changes.

List Branches

```
$ git branch -l  
branch1  
branch2  
*master
```

Tags

List Tags

```
$ git tag -l  
tag1  
tag2  
tag3
```

Remote Branches

remote branches are remote servers that we are tracking.

List Remote Branches

```
$ git branch -r  
mark/master  
paul/master  
chris/master
```

Creating Branches

```
$ git branch name
```

Creates a branch named name

```
$ git branch name commit
```

Creates a branch named name from the specified commit

Switching Branches

```
$ git checkout name
```

Checkouts the branch named name

```
$ git checkout -b name
```

Creates a branch named name and checks it out in one command

Deleting Branches

```
$ git branch -d name
```

Delete branch named name

Merge Branches Easy Sample

```
$ git checkout -b bug1
$ vim file
$ git add file
$ git commit -m"Fixed"
$ git checkout master
$ git merge bug1
$ git commit
```

Remote Branches

```
git remote add timmy mturner@timmy.local:/code/projectname.git
```

Adds a remote branch.

```
$ git pull timmy master
```

Pull and merge the changes from “timmy” into my current branch

```
$ git push timmy master
```

Push my master to the “timmy” server

Remote Branch Demo

Resources

- GIT's website: <http://git-scm.org>
- GitHub Guides: <http://github.com/guides/home>
- Git Ready: <http://www.gitready.com/>
- Great Git Intro: <http://www.eecs.harvard.edu/~cduan/technical/git/>