1920x1080

112 quotes & text

72 URLs & citations

72 `code{:;}`

36 credits

Growing Pains

Software Repositories at **SCALE**

Do you put all of your bits in a single gigantic repository or many smaller ones?

Why are we even asking?

- Ten years ago most people were using centralized SCMs.
- Nature of Software Development has changed.
- Software projects have become more complicated.
- More outsourcing and partnering.

Outline

- Some historical context.
- Kinds of SCMs.
- Advantages and disadvantages of Monorepo & Multirepo.
- What serves you?

BitKeeper
1999

Arch
2002

monotone
2003

Darcs
2002

ArX
2003

SVK
2003

mercurial
2005

Bazaar
2005

git
2005

fossil
2007

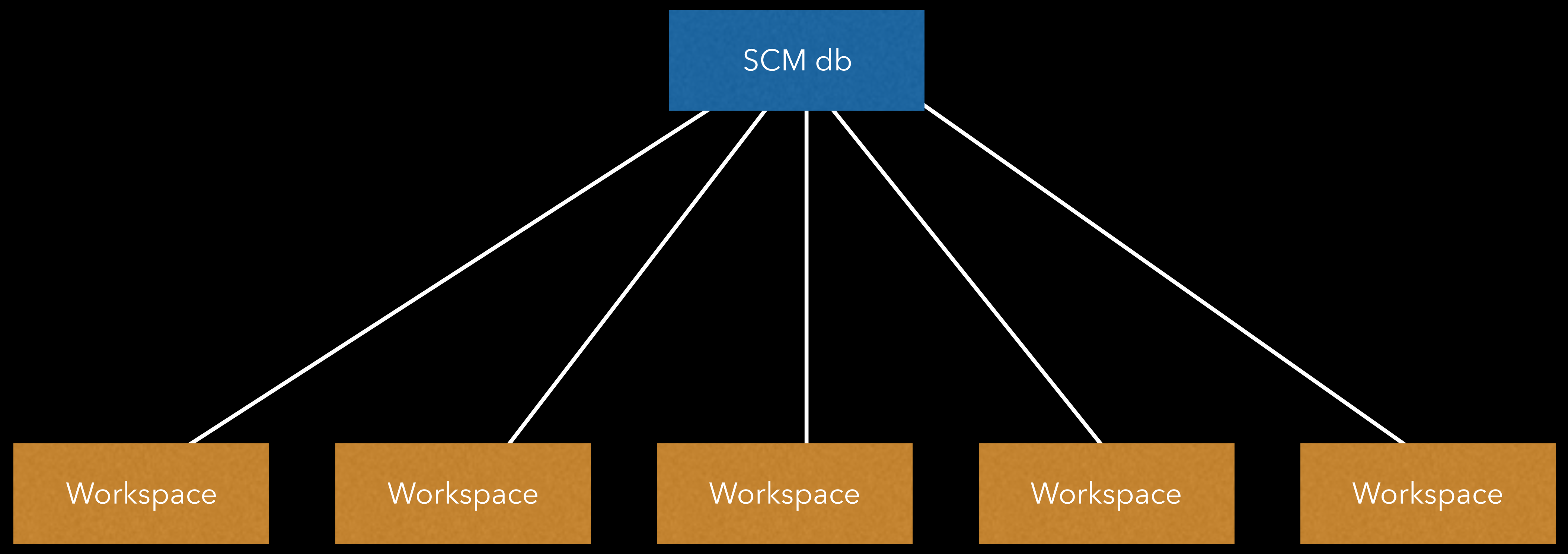
Subversion
2000

AccuRev
2002

TFS
2005

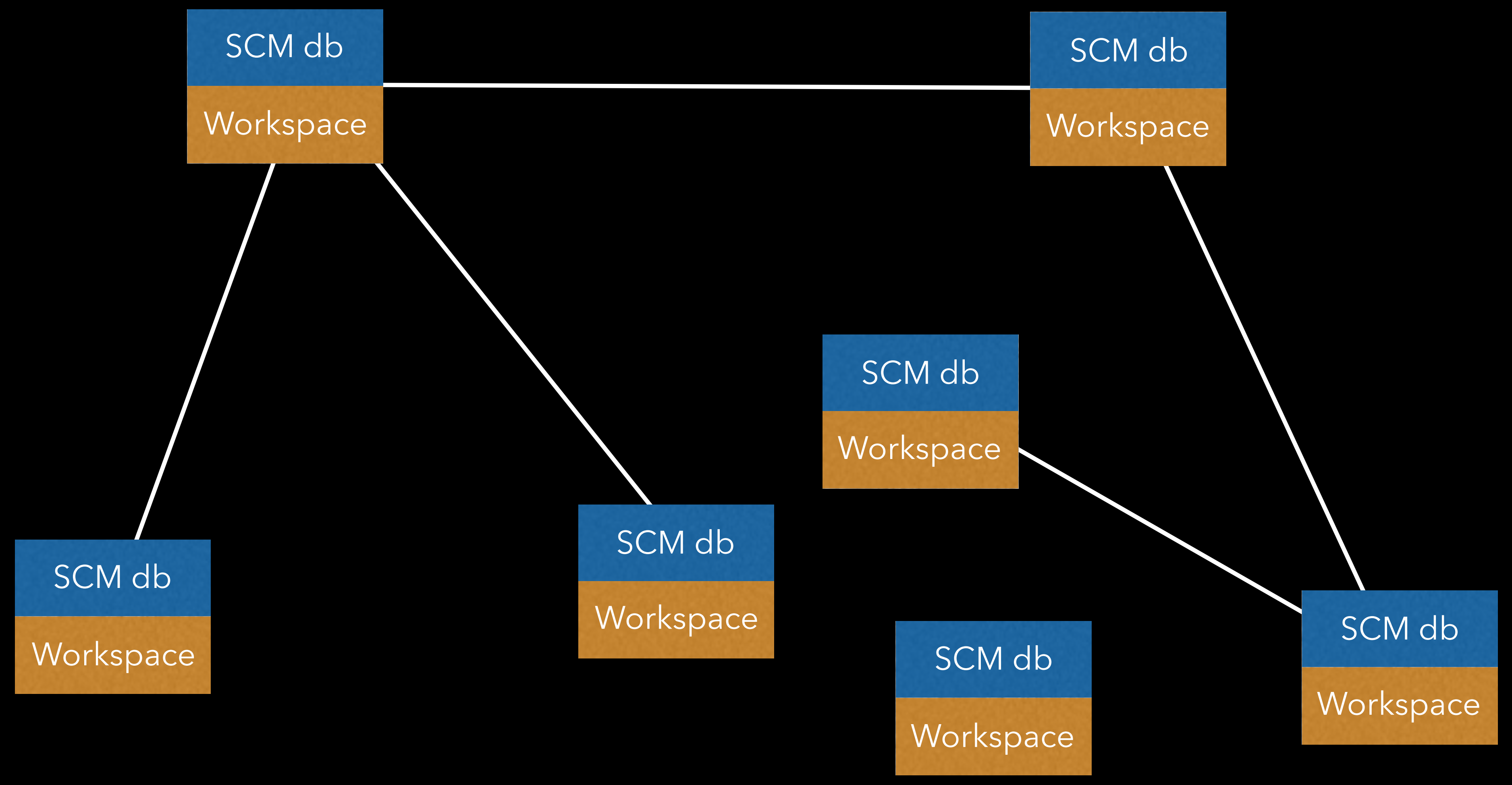
git
2005
BitKeeper
1999
mercurial
2005
2015
and beyond
Subversion
2000
Perforce
2000

Centralized



VS

Distributed



Centralized SCM

Advar

- Partial checkout
- Binary handling
- Single place to k where your sour
- Security: you ca permissions on the server.
- File Locking.

ntages

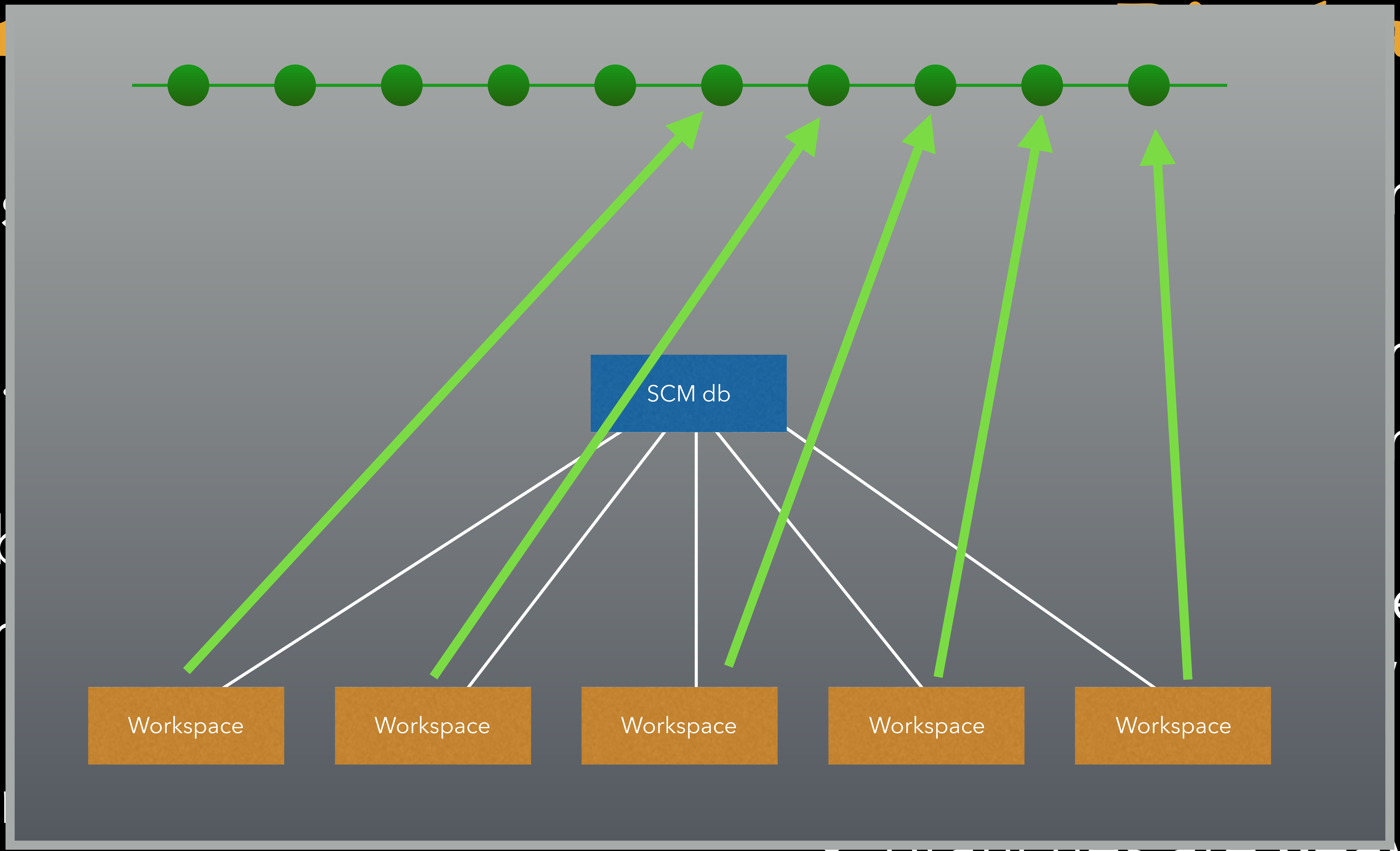
really parallel work.

nit model. Means nges in isolation.

es. Mixes 'publishing' code.

heavyweight.

- Limited workflow.



Distributed SCM

es

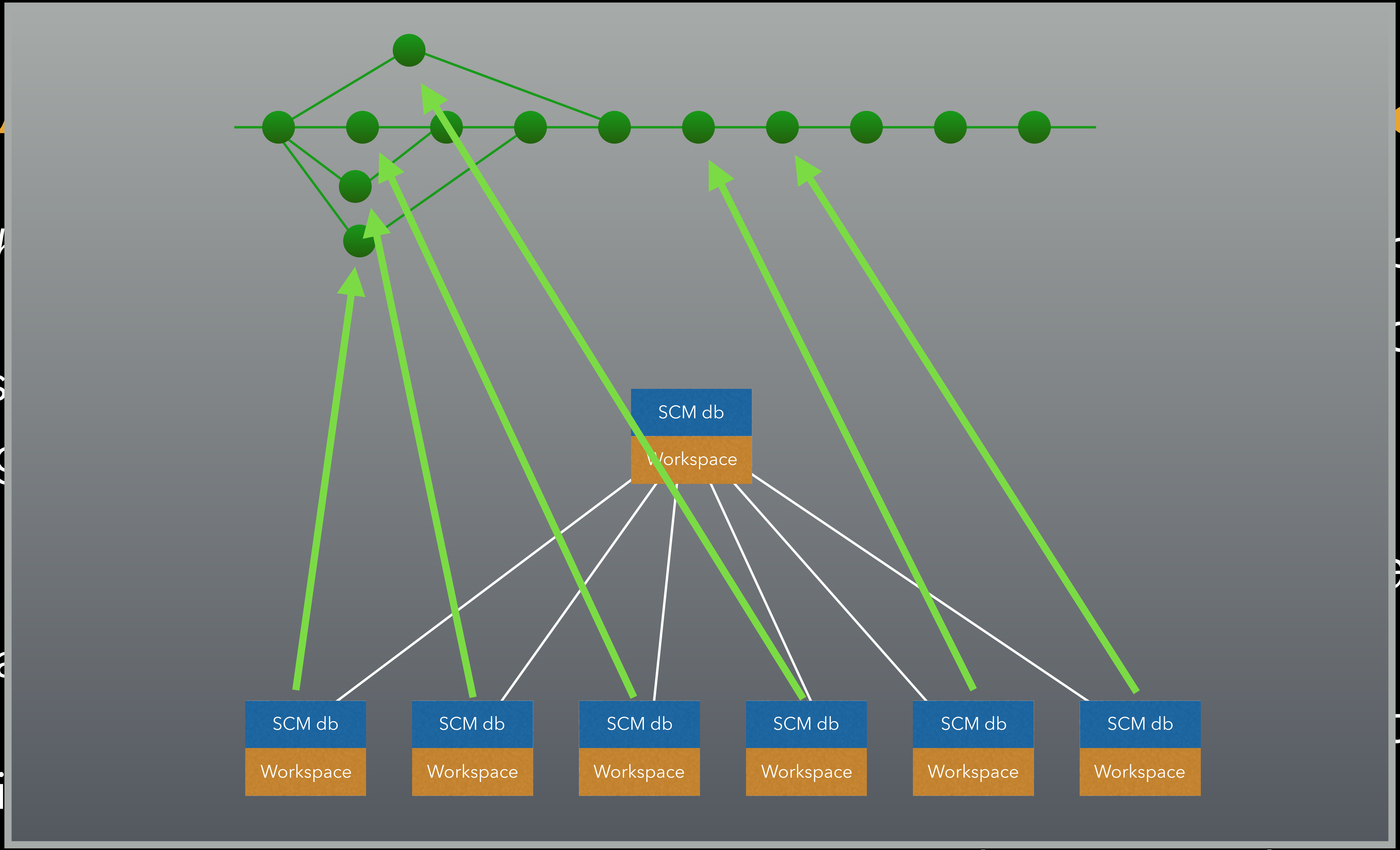
- Commit *th*
- Separates publishing sandbox.
- Implicit ba
- More flexi
- Branches are *lightweight*.

o more clude the

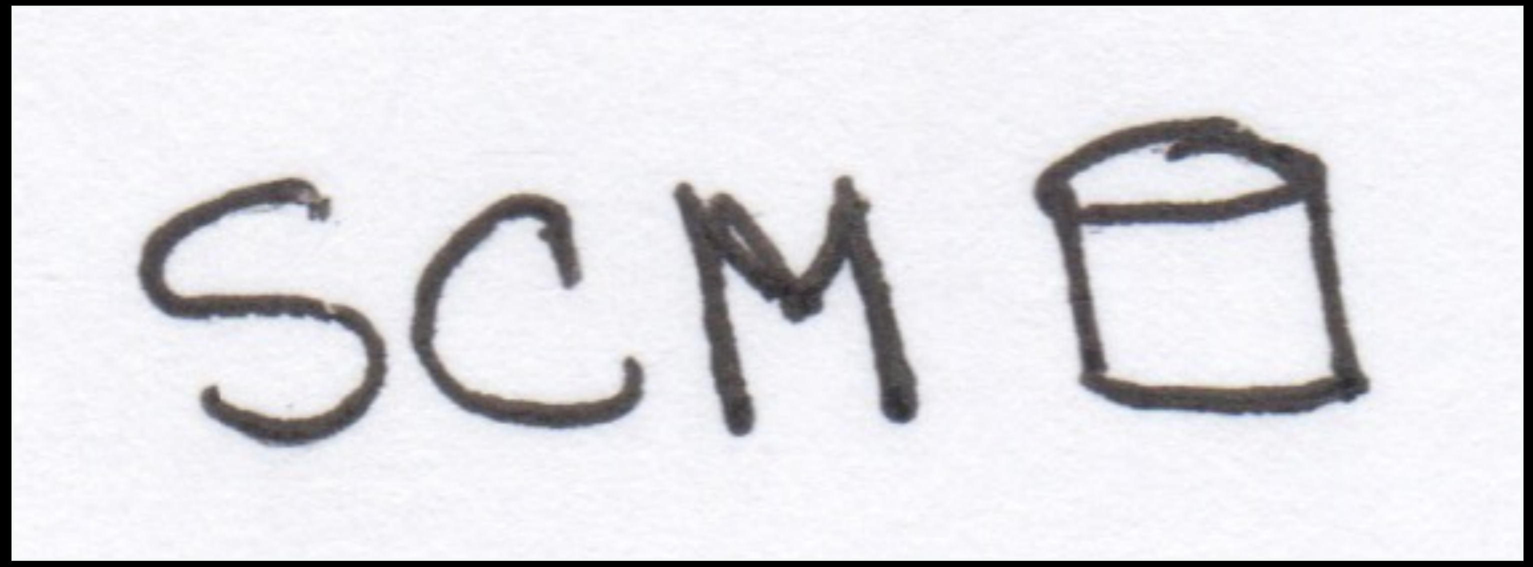
a problem.

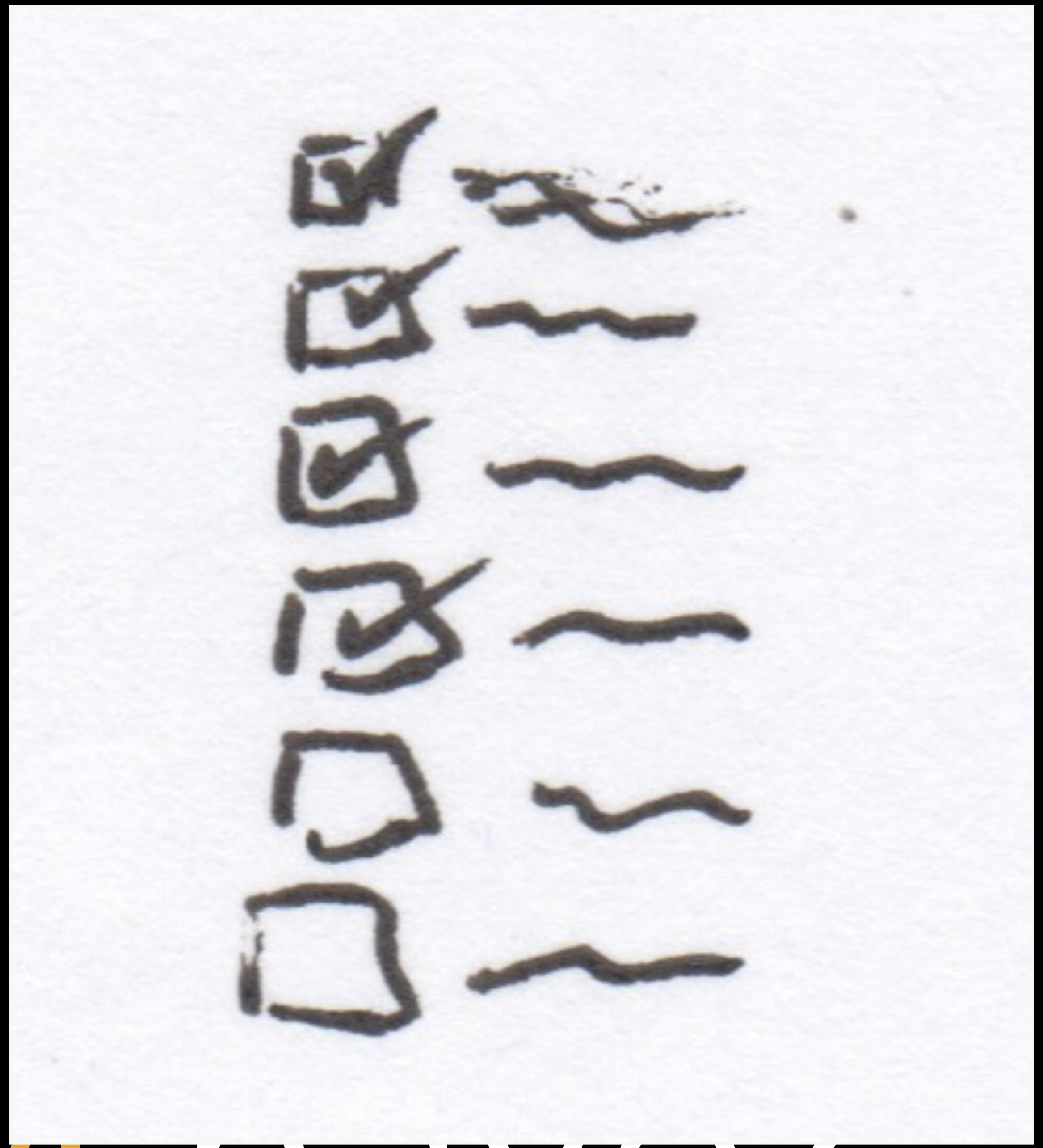
ts.

- Hard to control access.

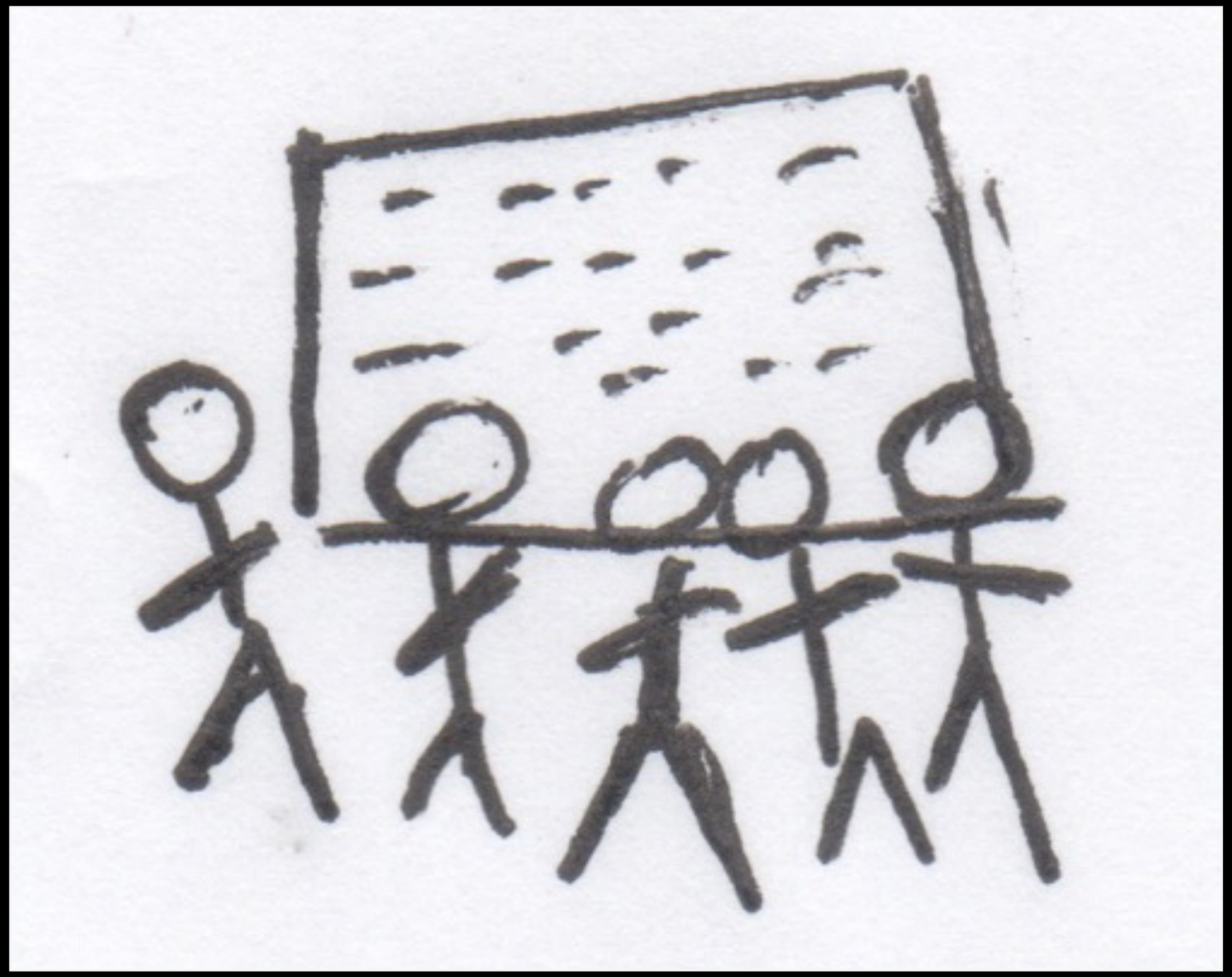
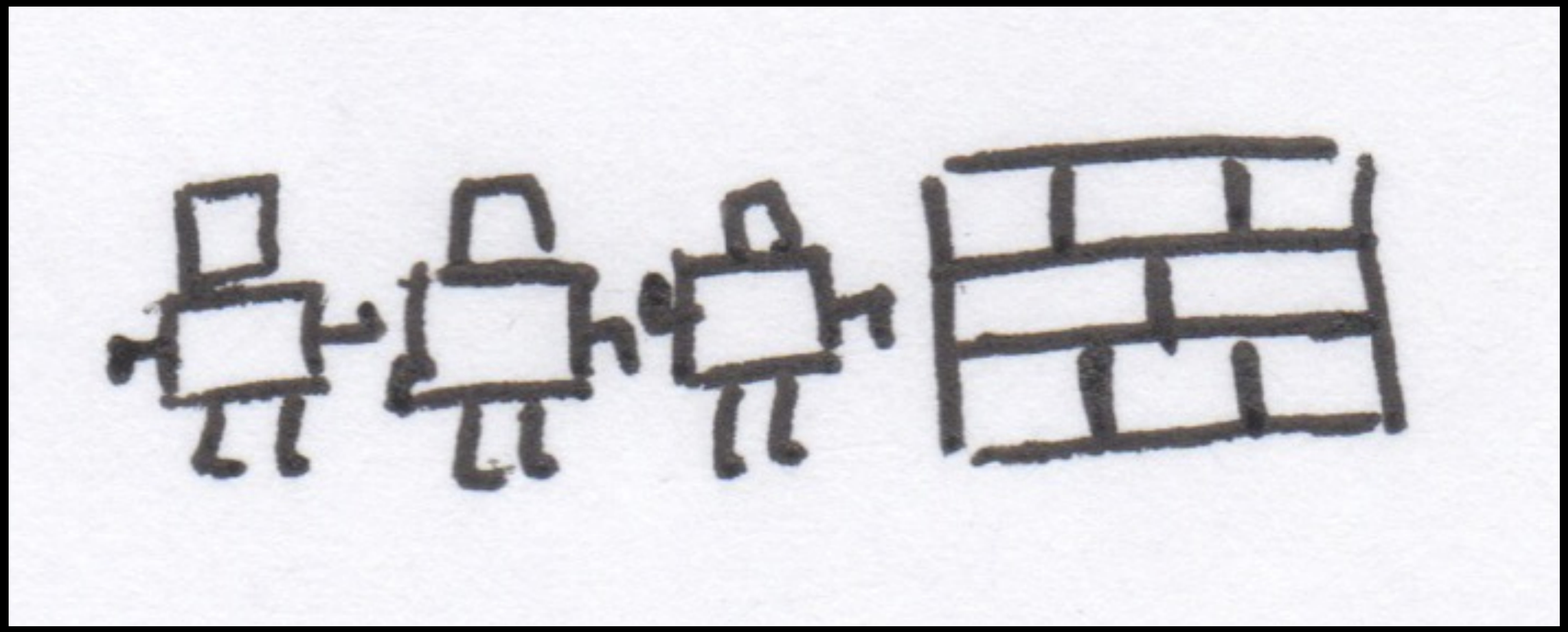
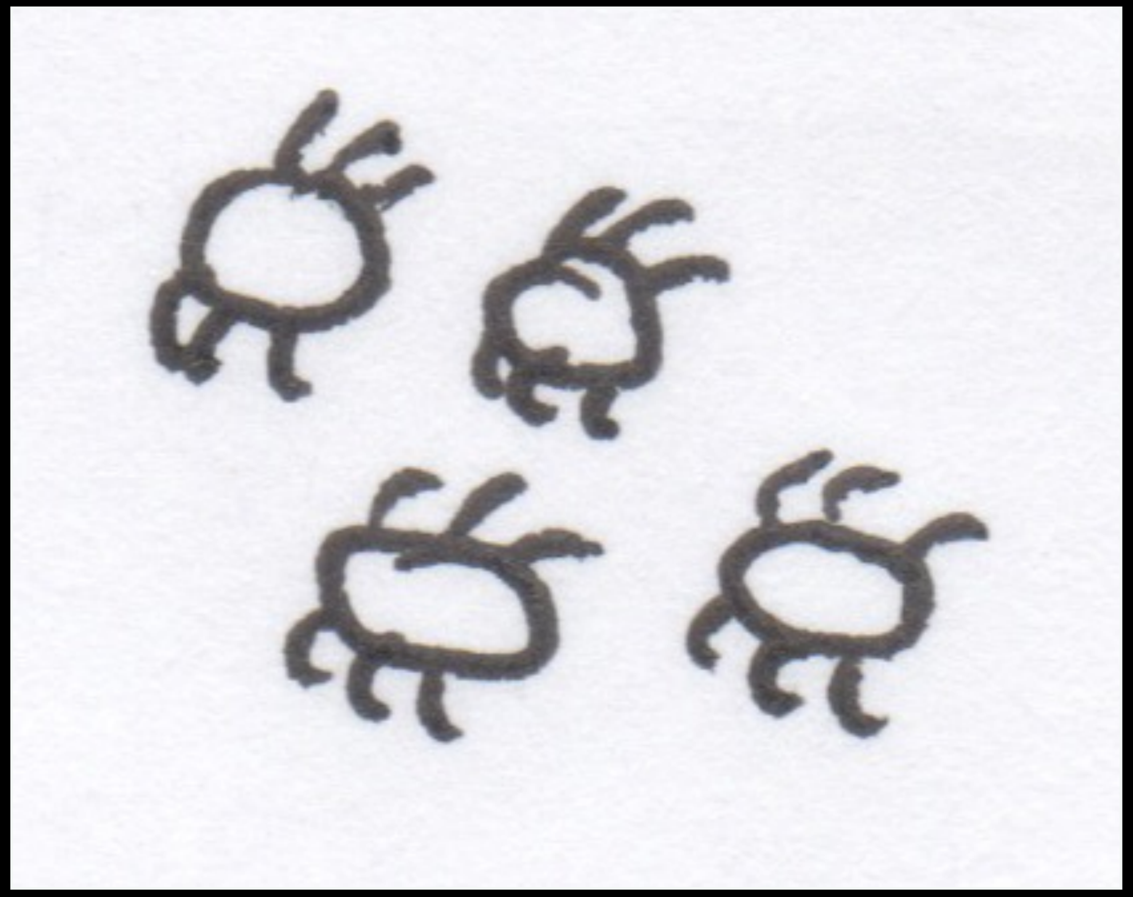


Why did **DVCS** overtake centralized systems?





What role does the SCM have?

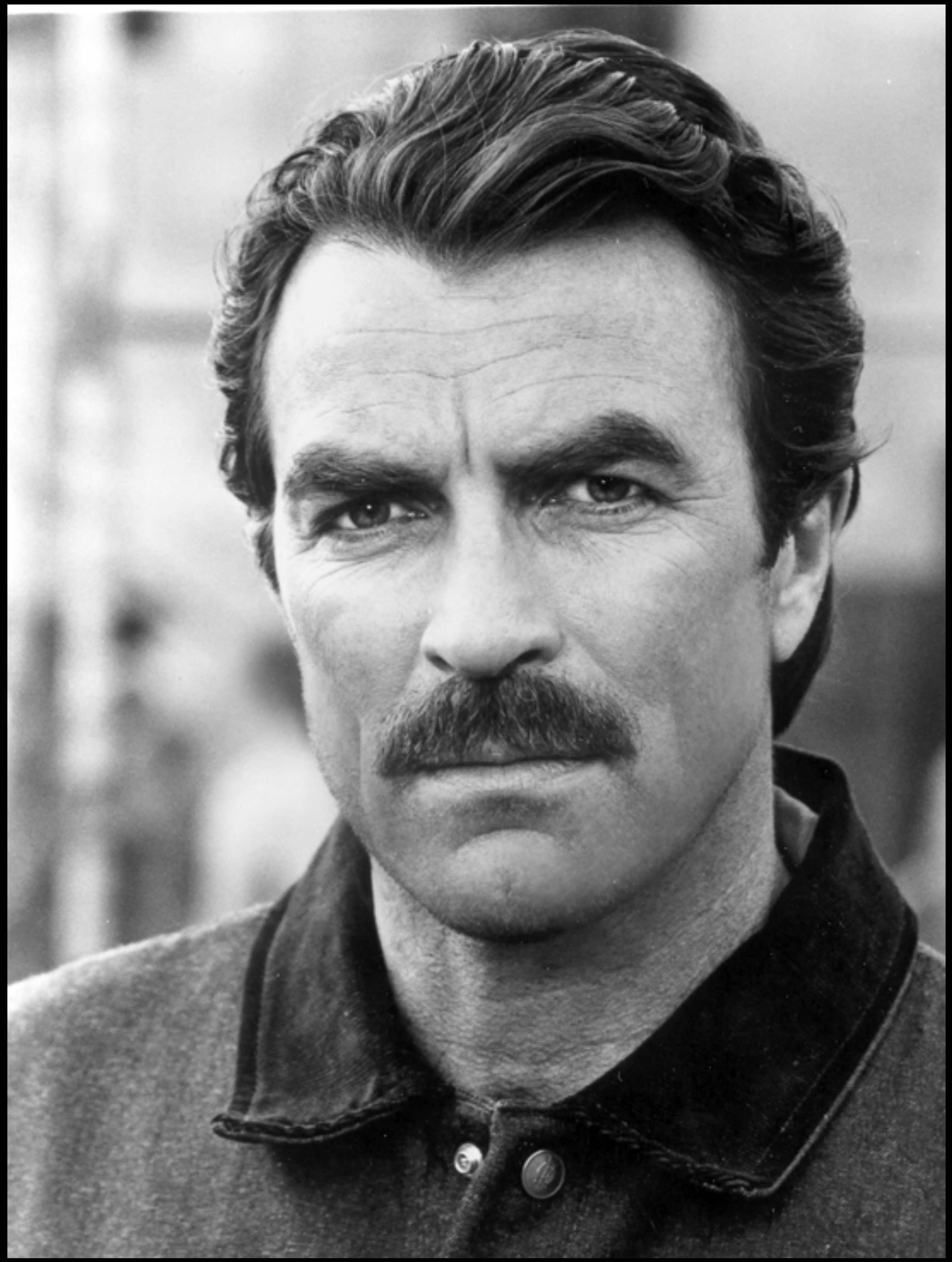




SCM as Backup

- Check files in.
- Check files out.
- Occassionally revert to a previous version.

SCM as Detective

- When was this bug introduced?
  - Bisect
  - History exploration tools.
- Who deleted this?
- Why is this code this way?



SCM as Data

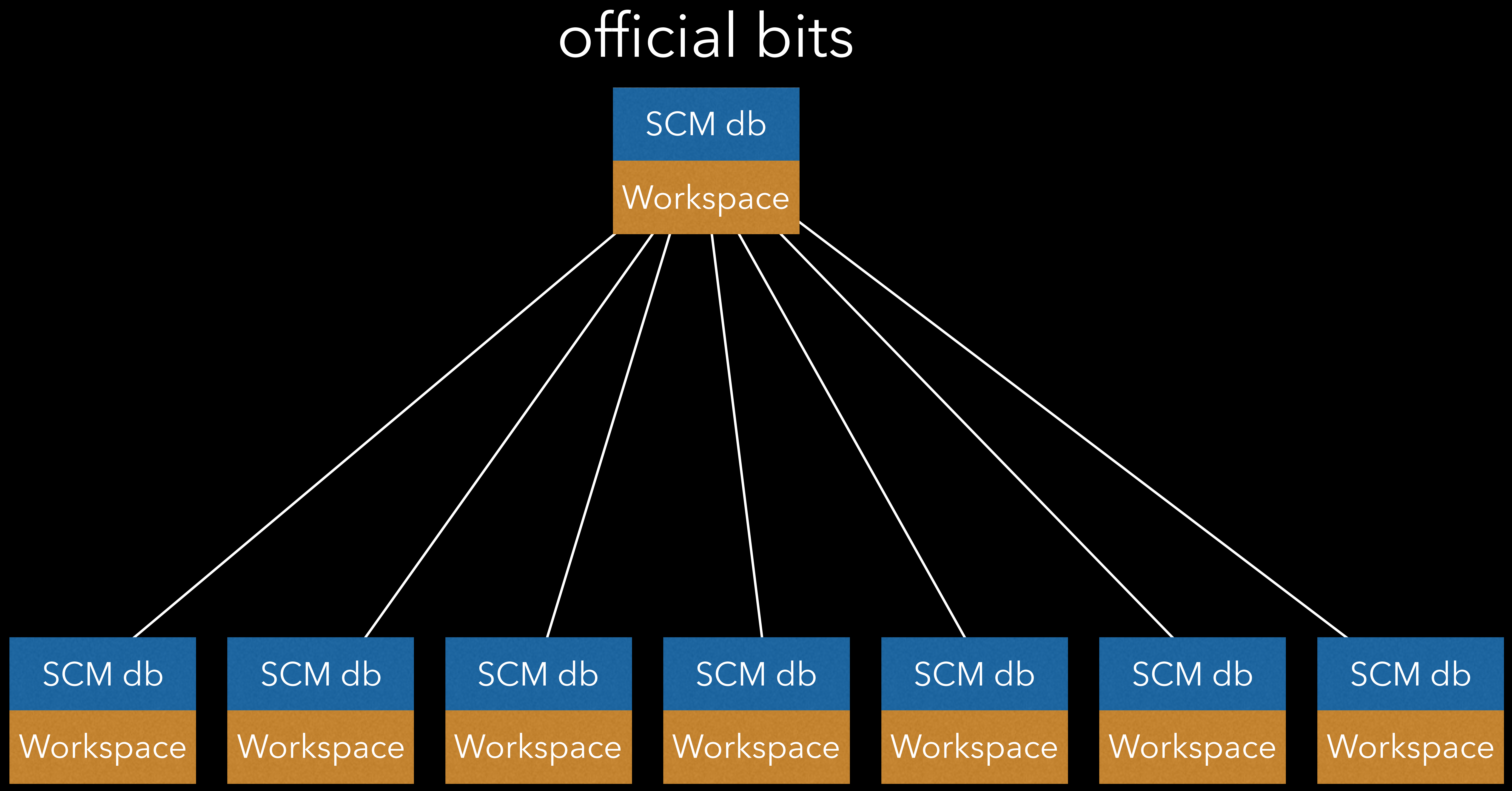
- Historically, how long does it take us to develop a feature?
- How long to fix a bug?
- Which areas of the code are unmaintained? Obsolete? Can be removed?

SCM as Post Mortem

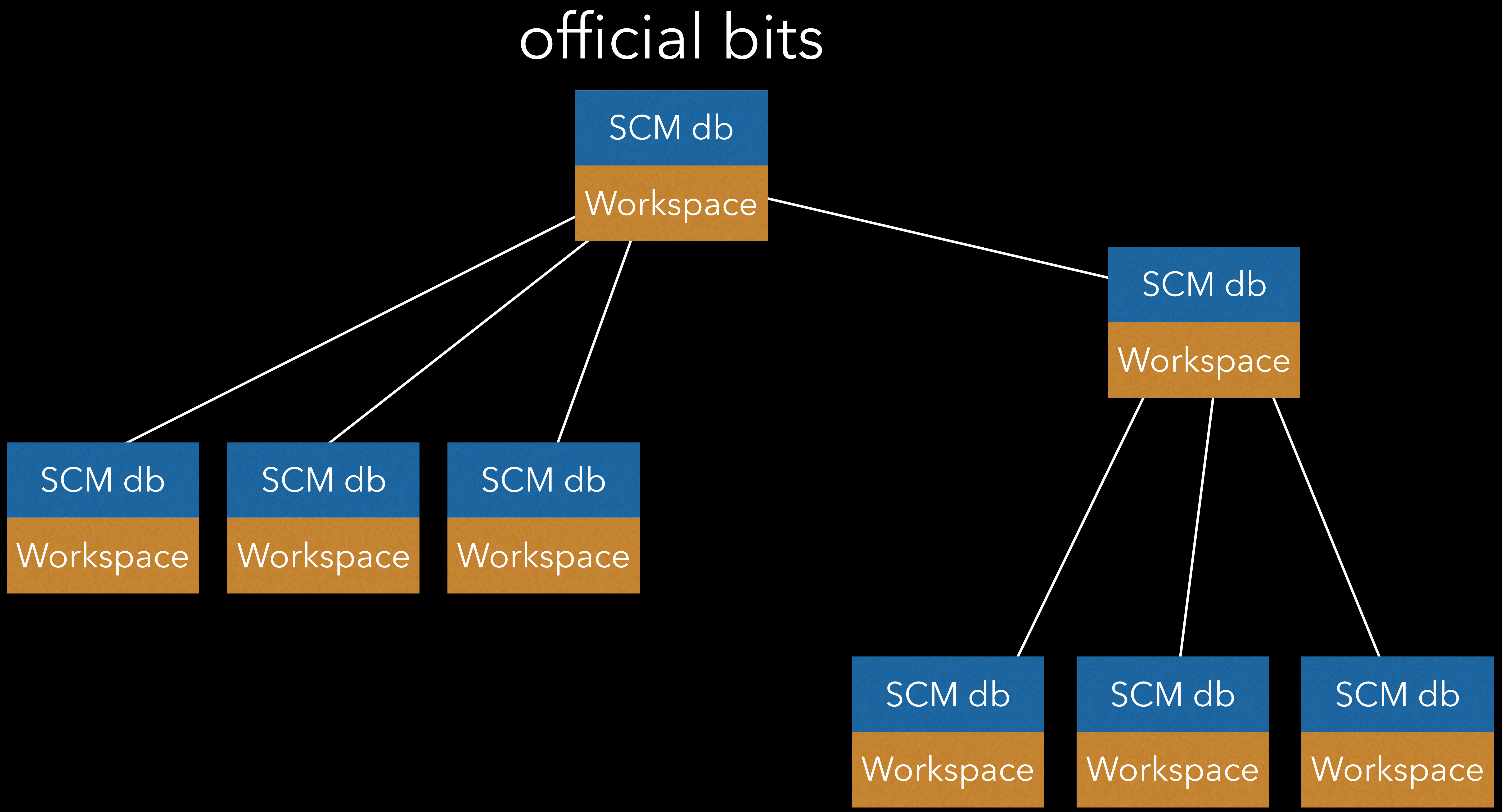
- What caused us to ship this bug?
- What could we have done to prevent it?

It’s about **Workflow**

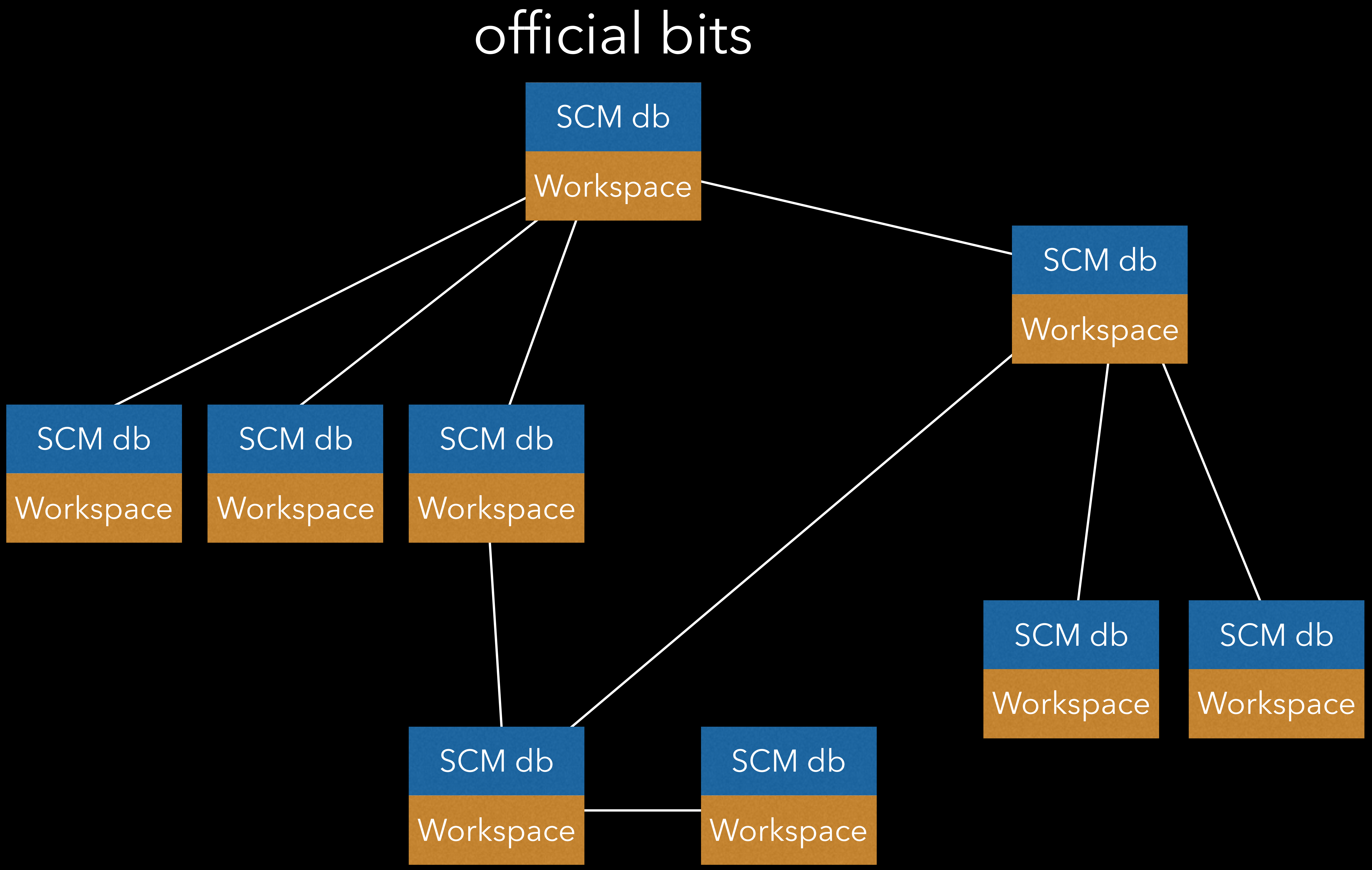
Centralized Workflow with DVCS



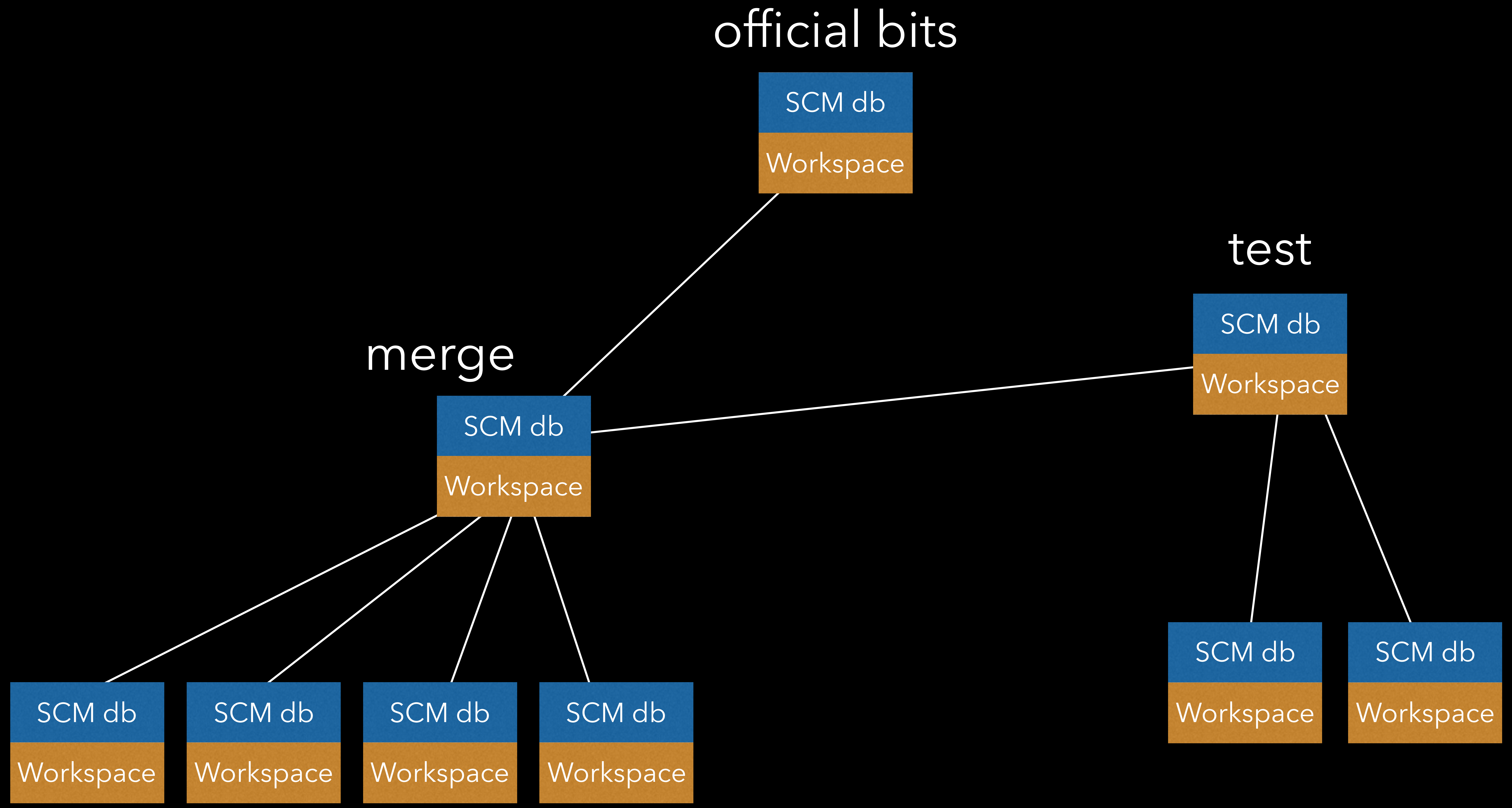
Workflow with DVCS



Workflow with DVCS



Workflow with DVCS



Every workspace is a branch

Three Problems with DVCS

Binary Files

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Security

Large Source Bases

Three Problems with DVCS

Binary Files

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Security

Large Source Bases

Binaries Don’t Diff Well

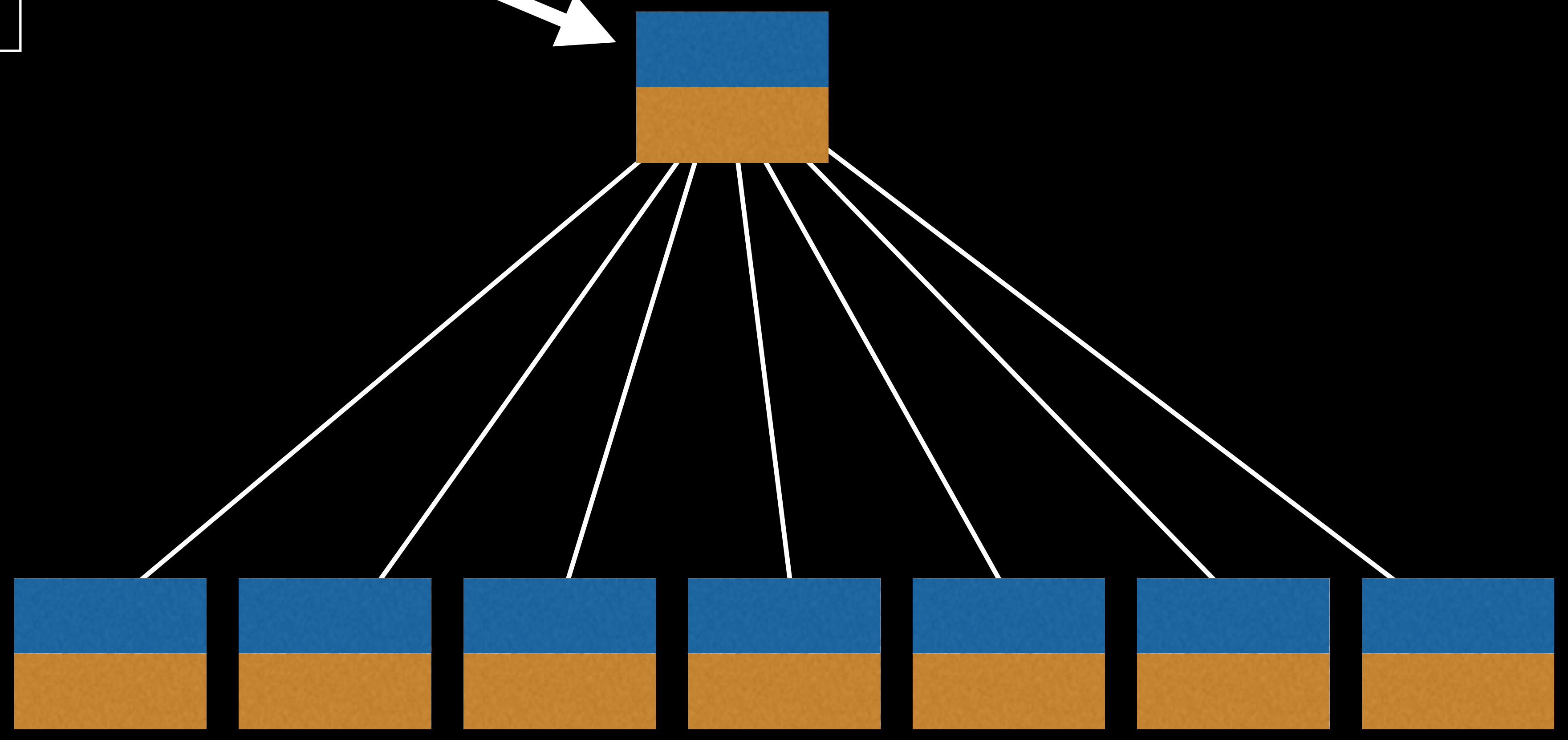
- Rolling checksums help “chunk”.
- However, some file formats *trickle* changes.
    - Video formats.
    - Image formats.
- Storing every copy bloats the history.

Binary Files

Solution: Make them act more like centralized systems!

And store the contents in a server (or many).

BitKeeper BAM
Git LFS
Mercurial LFE



Replace binary files in history with pointers.

If someone wants an old copy, it's fetched on demand.

Three Problems with DVCS

Binary Files

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Security

Large
Source Bases

Security in DVCS

- With a monorepo
  → All or nothing.

- With multirepo (including nested)
  → Access at a repository level.

- Read vs Write Access
  → Anyone can commit, don't let them push!

Three Problems with DVCS

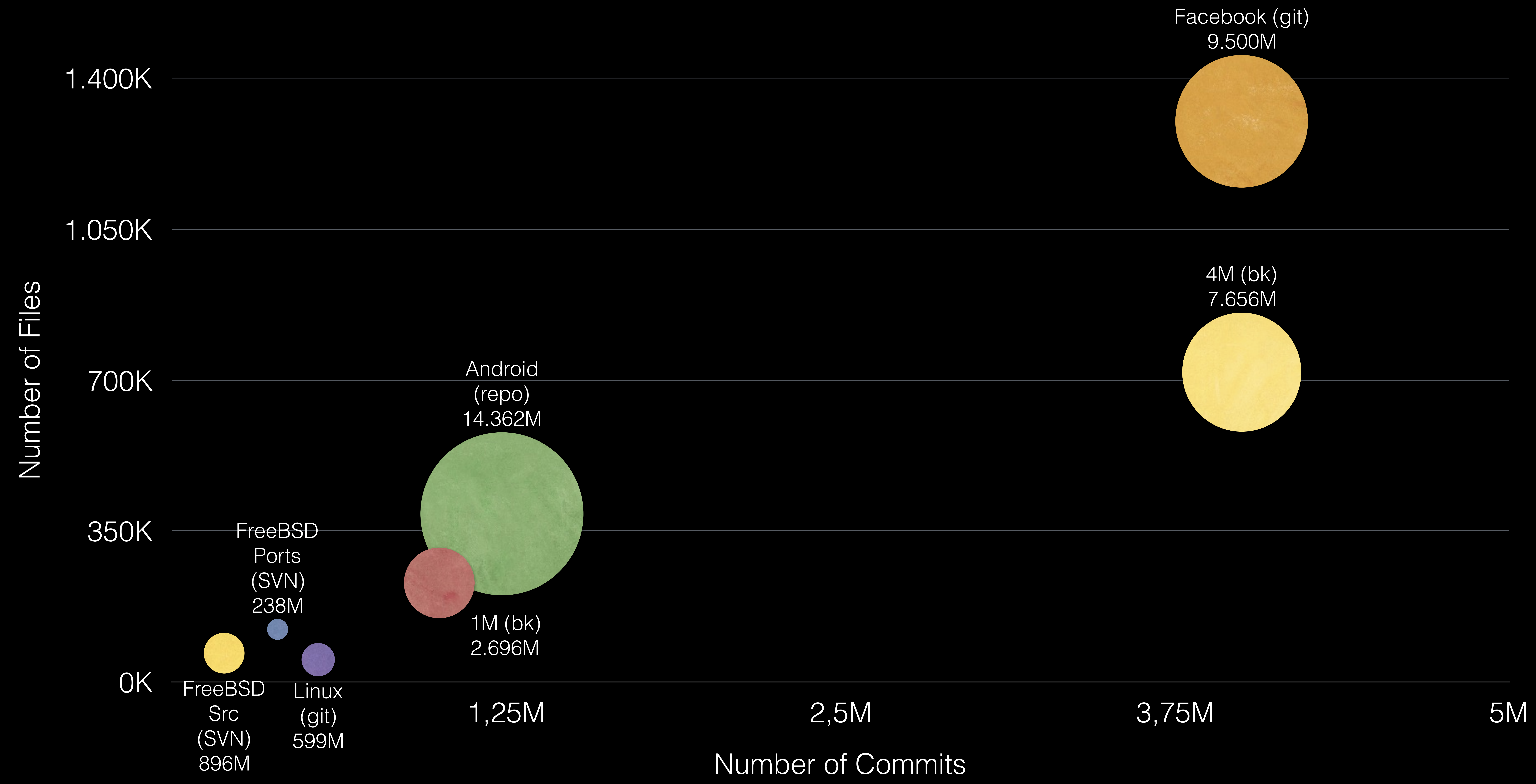
Binary Files

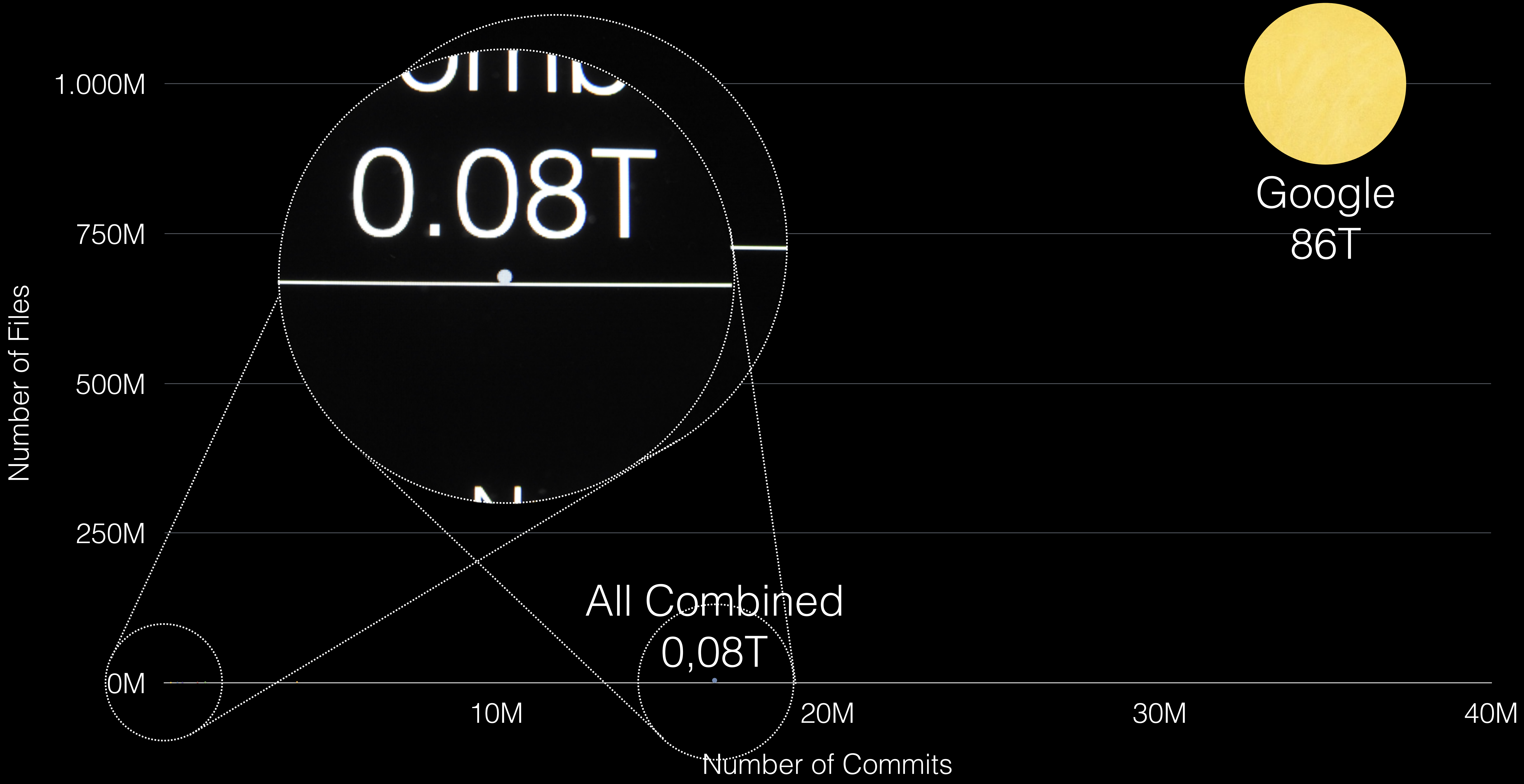
```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

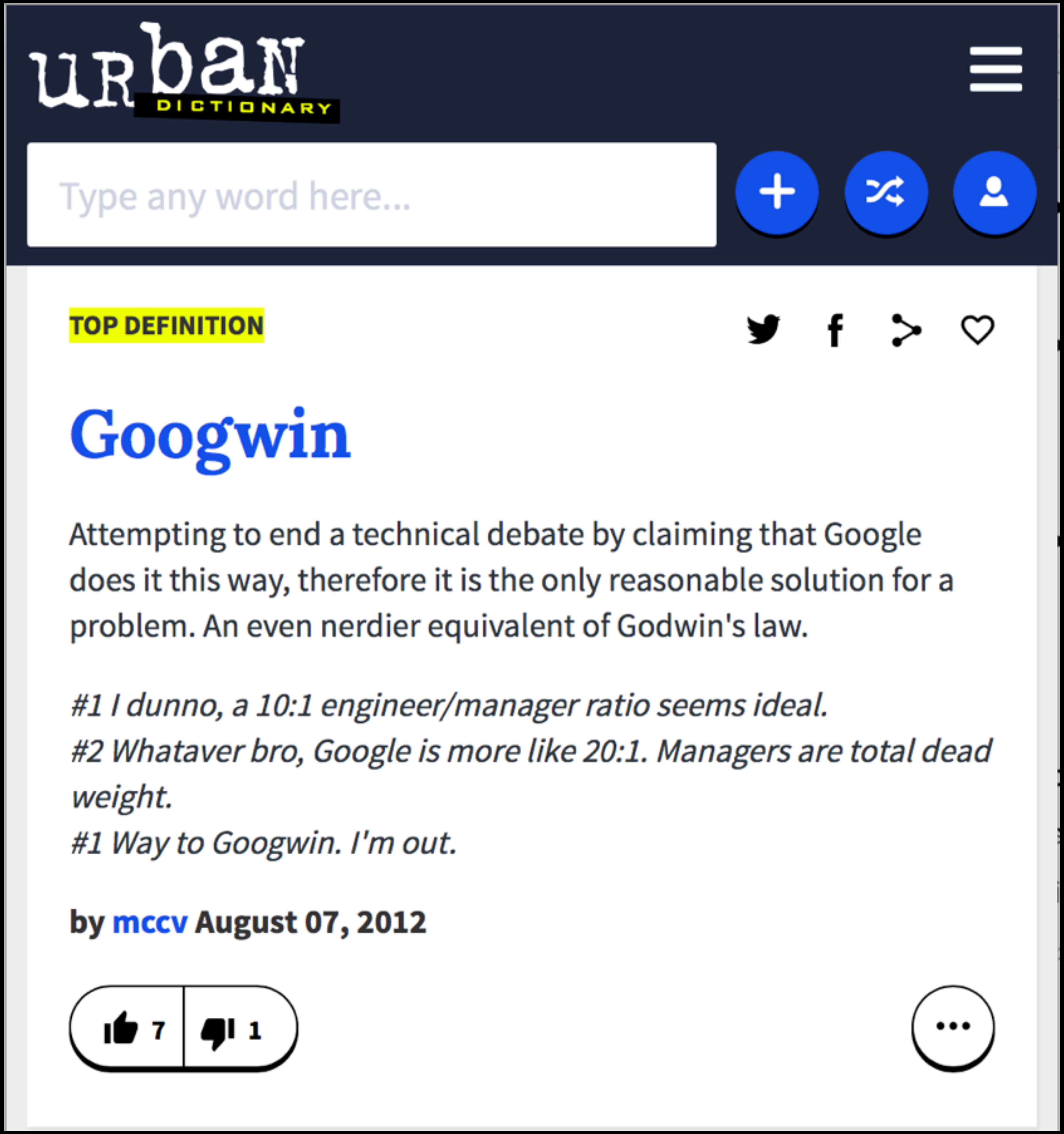
Security

Large Source Bases

LARGE source bases


Facebook (git)
9.500M
4M (bk)
7.656M
Android
(repo)
14.362M
1M (bk)
2.696M
FreeBSD
Ports
(SVN)
238M
FreeBSD
Src
(SVN)
896M
Linux
(git)
599M
1.400K
1.050K
700K
350K
0K
Number of Files
1,25M
2,5M
3,75M
5M
Number of Commits


Google
86T
All Combined
0,08T
0.08T
1.000M
750M
500M
250M
0M
10M
20M
30M
40M
Number of Files
Number of Commits



TOP DEFINITION

Googwin

Attempting to end a technical debate by claiming that Google does it this way, therefore it is the only reasonable solution for a problem. An even nerdier equivalent of Godwin's law.

#1 I dunno, a 10:1 engineer/manager ratio seems ideal.
#2 Whataver bro, Google is more like 20:1. Managers are total dead weight.
#1 Way to Googwin. I'm out.

by mccv August 07, 2012

7 1

Monorepo
vs
Multirepo

Advantages of a monolithic repository

- Unified versioning, one source of truth
- Extensive code sharing and reuse
- Simplified dependency management
- Atomic changes
- Large scale refactoring, codebase modernization
- Collaboration across teams
- Flexible team boundaries and code ownership
- Code visibility and clear tree structure providing implicit team namespacing

Some Disadvantages

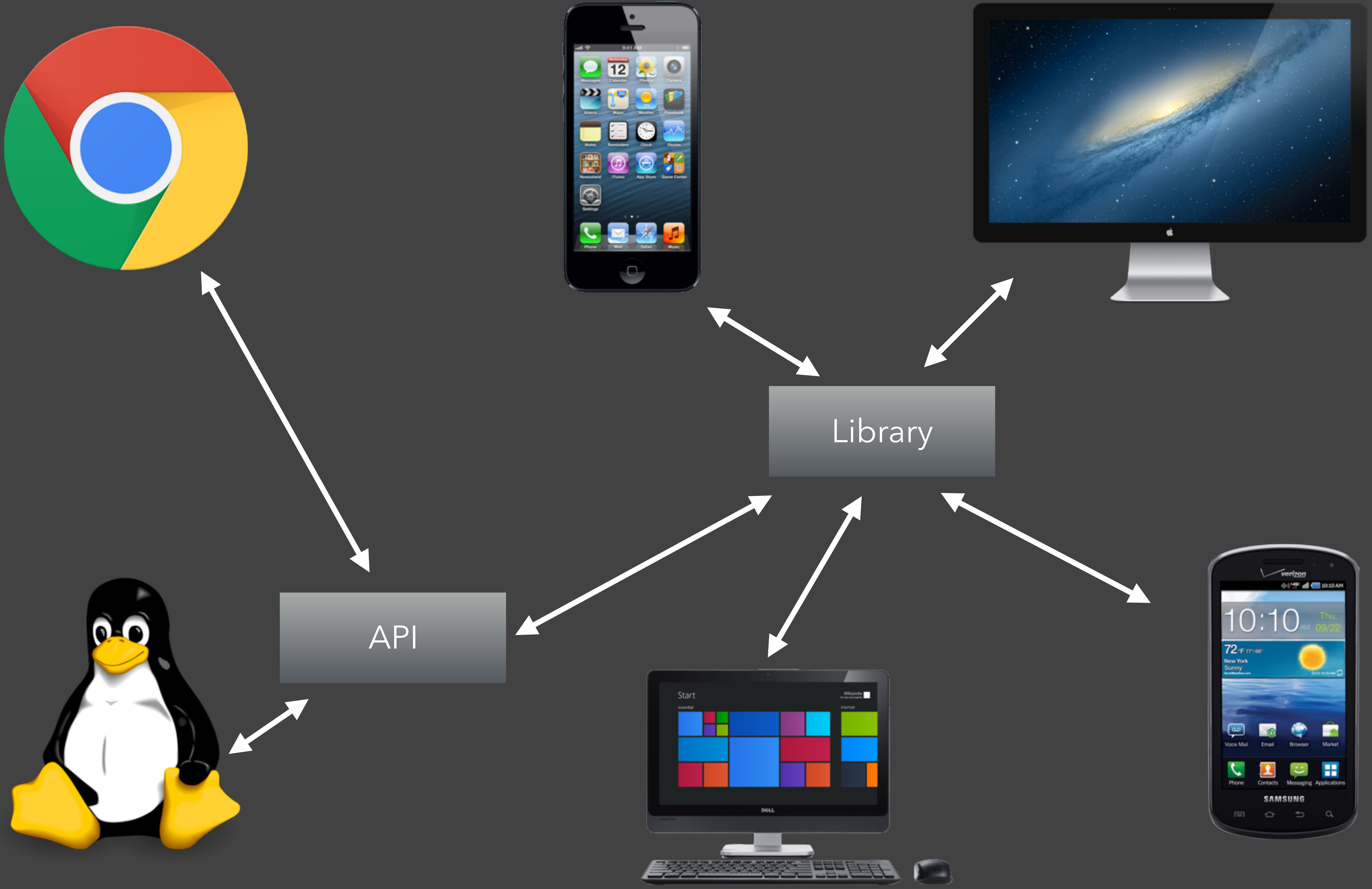
- A little *too easy* to share.
- Access control. (E.g. Outsourcing.)
- Noisy commit messages.
- Cloning no longer an option.

monorepi
@monorepi

software engineering

4:18 PM - 19 Nov 2013

Not just LARGE
also COMPLICATED


Library
API

What about multirepo?

webapp.git
app.git
macApp.git
restapi.git
Library
libglue.git
server.git
API
WinApp.git
droid.git

ONE DOES NOT SIMPLY
CHANGE A PUBLIC API

Problems of Multirepo

- Loss of atomicity.
- Loss of the ability to use SCM tools.
- That feeling of "*Never change anything*".
- Having multiple repositories breaks tools that interact with the SCM.

Mono vs Multi?
How about a Hybrid?

- Partial Checkouts.
- Preserves Atomic Commits.
- You can decouple and reuse components.

Solution: Stitch together multiple repositories into one.

Case Study: Git Submodules

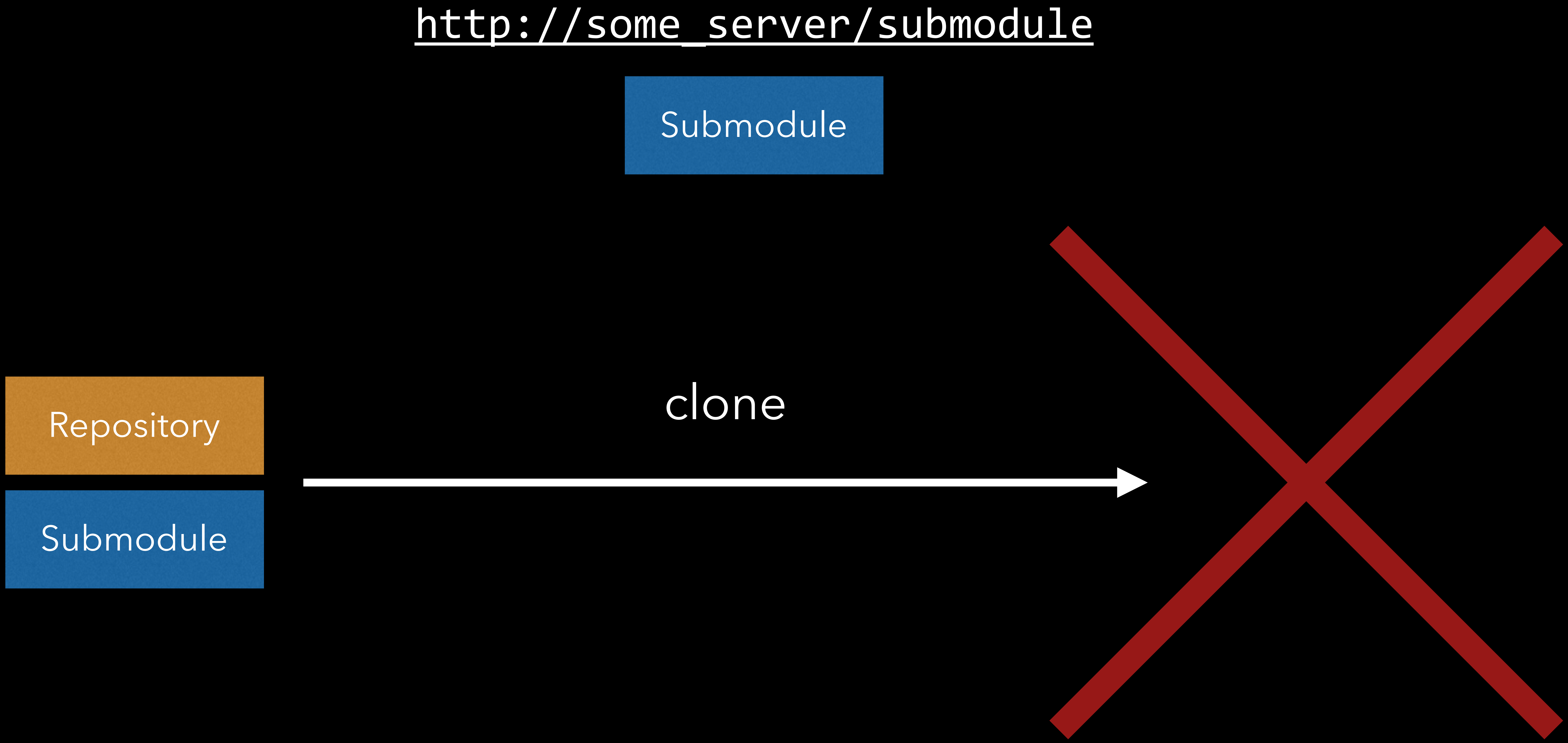
Repository

```
.gitmodules
        /submodule/path/in/repo
        http://some_server/submodule
```

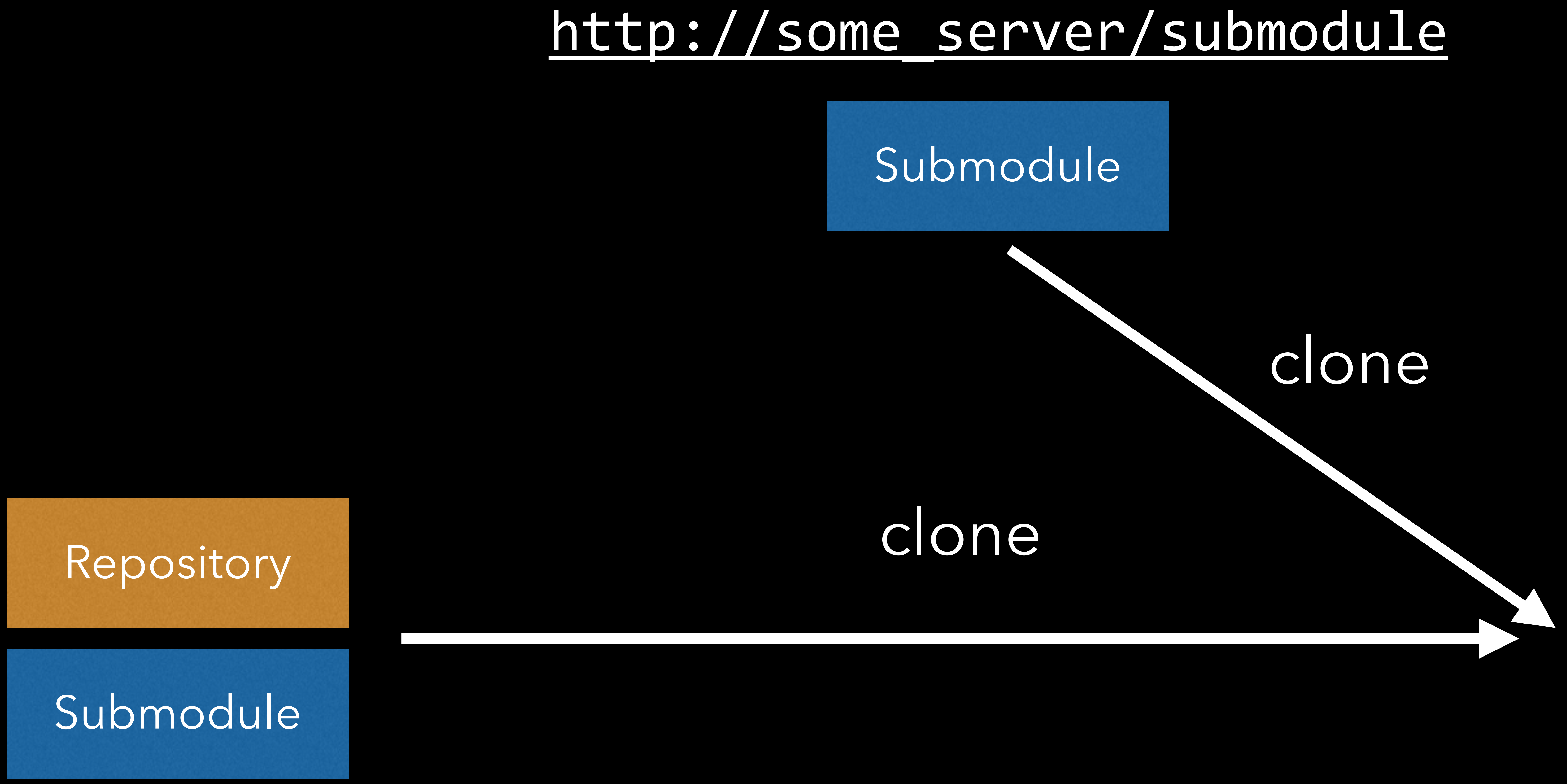
Submodule

```
e46fe3df01435bf523d2ab4f2755556c0e4e6f78
```

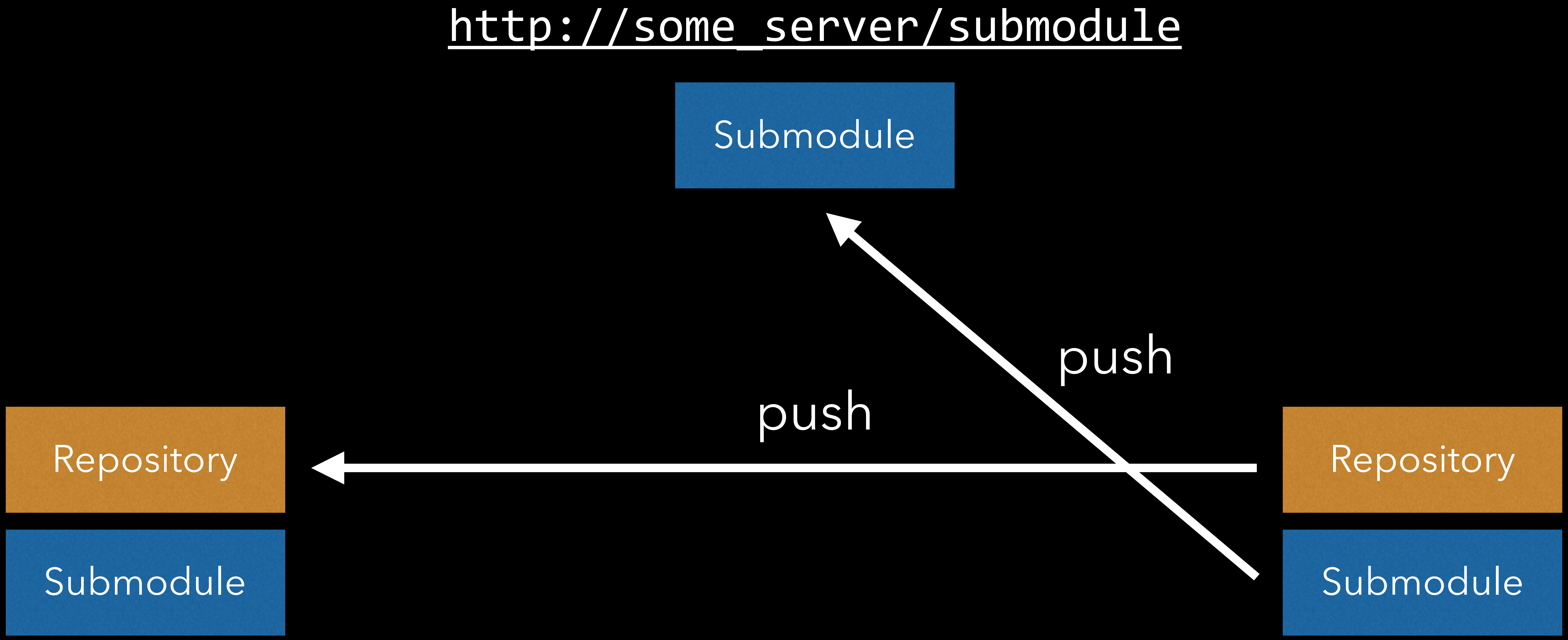
Case Study: Git Submodules



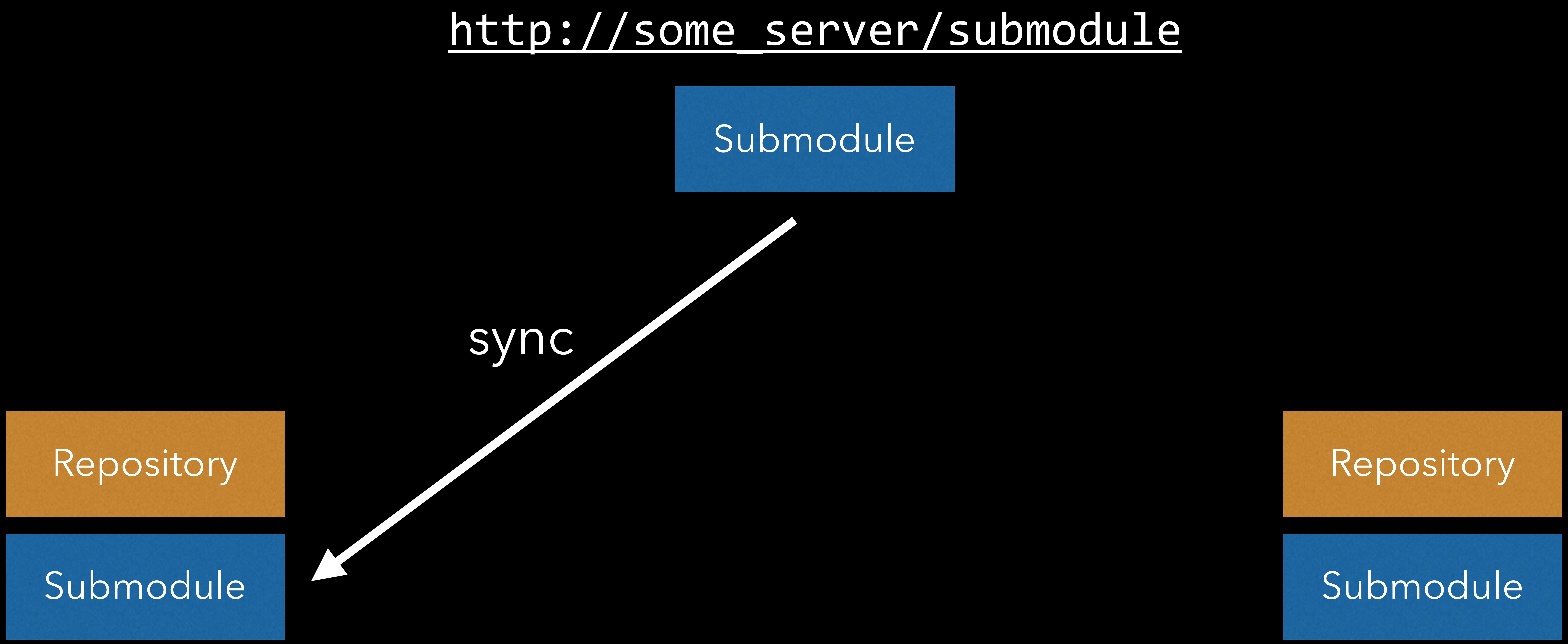
Case Study: Git Submodules



Case Study: Git Submodules



Case Study: Git Submodules



Case Study: Git Submodules

```
fatal: reference isn’t a tree: 6c…e0
Unable to checkout '6c…e0' in submodule path 'sub'
```

Means

Someone forgot to push the submodule ‘sub’.

Case Study: Git Submodules

```
submodule $ git push
Everything up-to-date
```

Means

You made a commit in the submodule while it was in a *detached head* state (the default). You will cause the problem outlined in the previous slide.

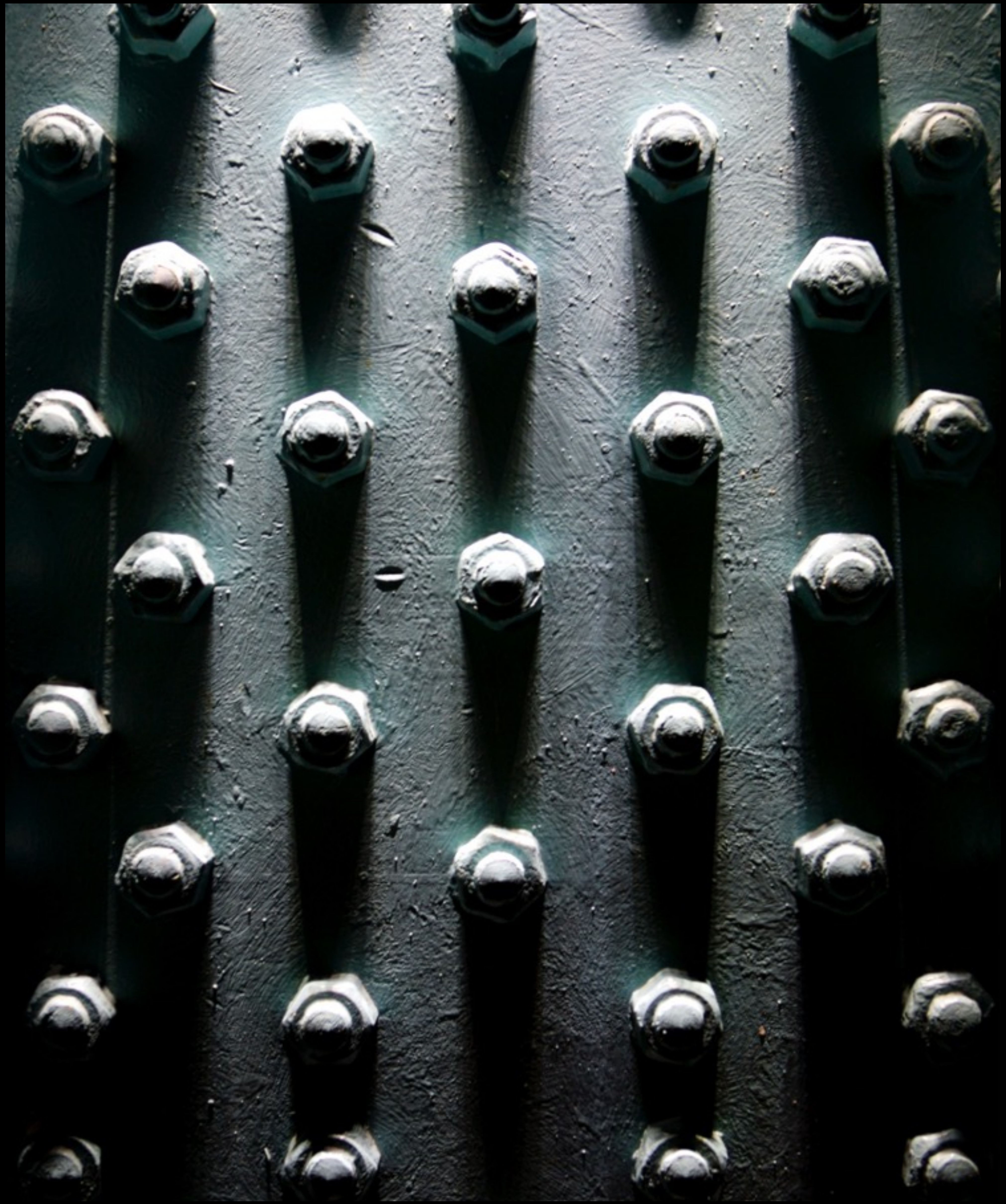
MY BRAIN HURTS

Git Submodules are *too loosely coupled* with the main repo.

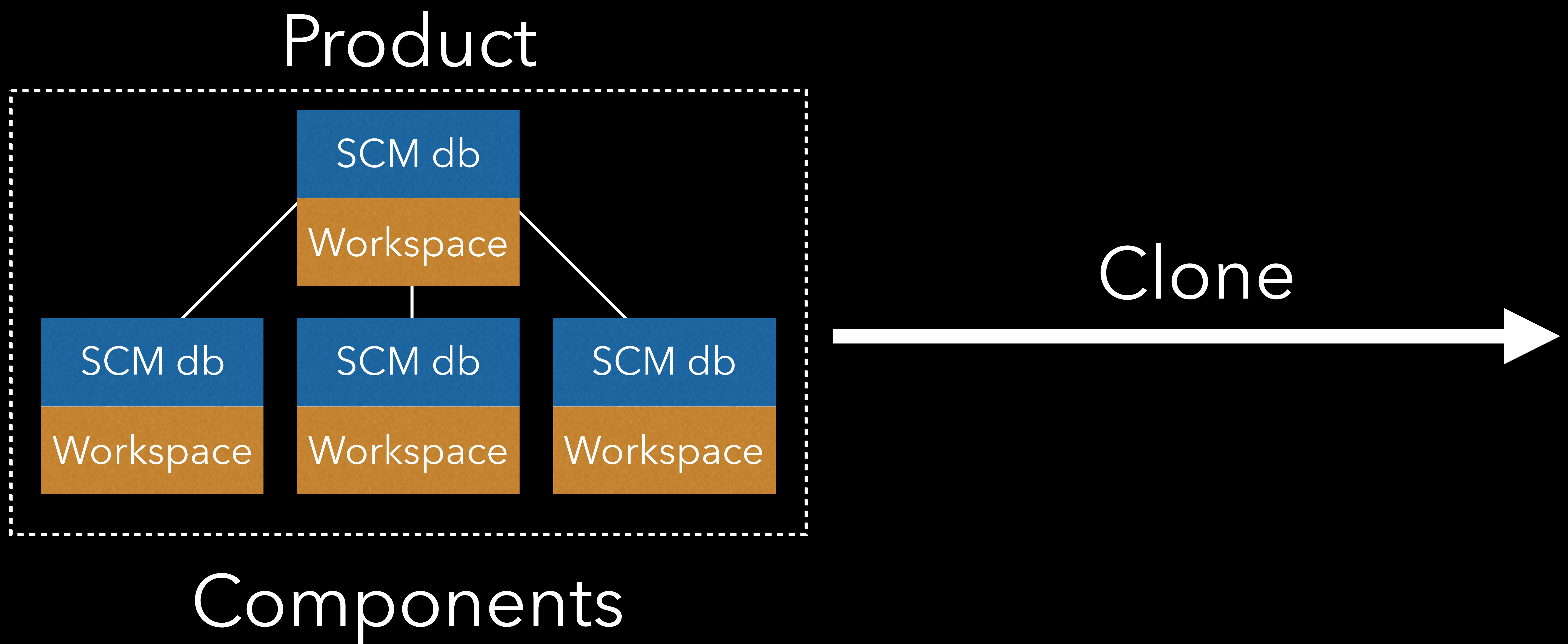
Key Insight

- We’ve seen this problem before:
  CVS
- We’ve solved this problem before:
  ChangeSets bind changes to independent files together.
- What if we treat repositories the same way we treat files?

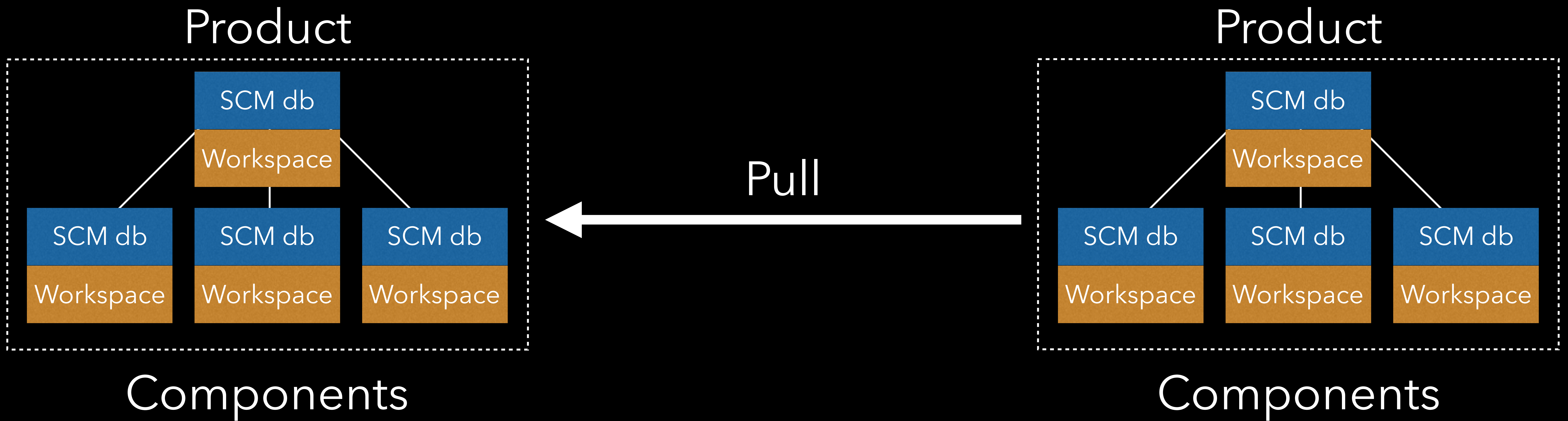
A component is to a product like a file is to a repository



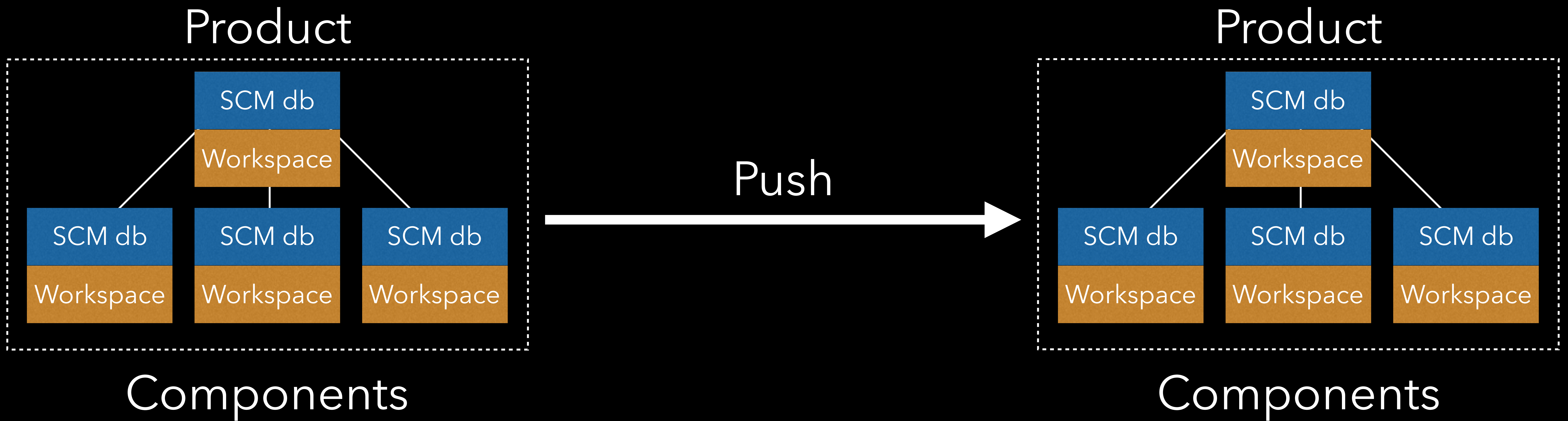
BitKeeper Nested



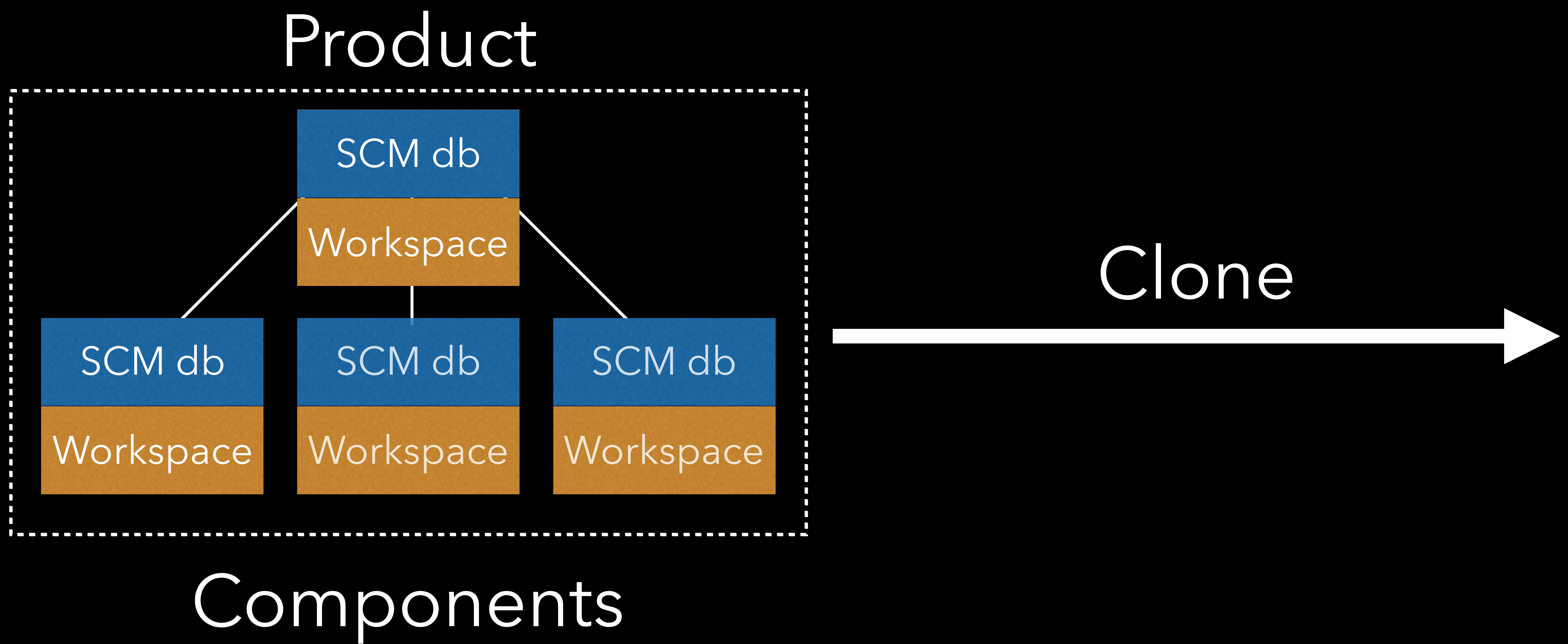
BitKeeper Nested



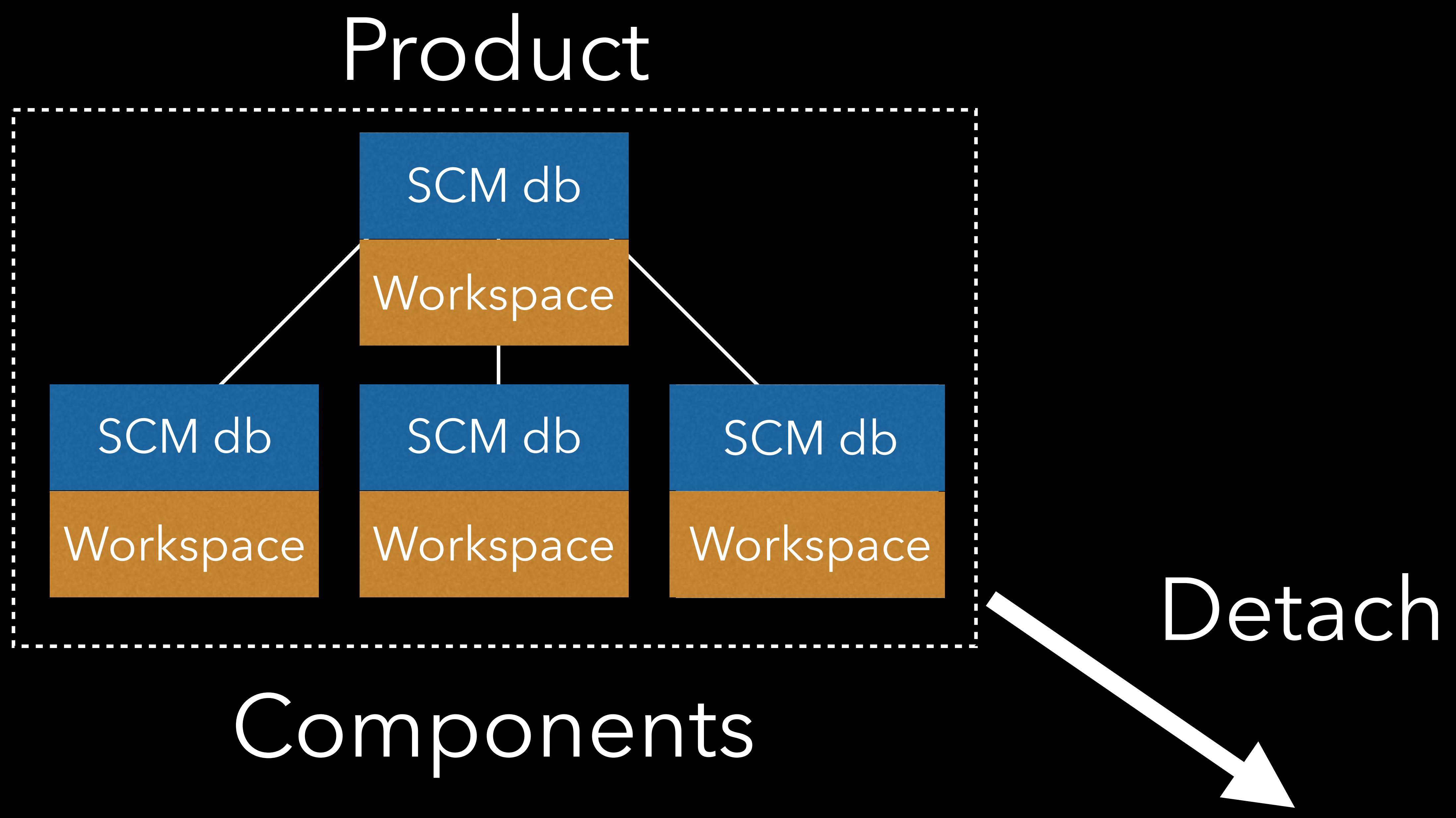
BitKeeper Nested



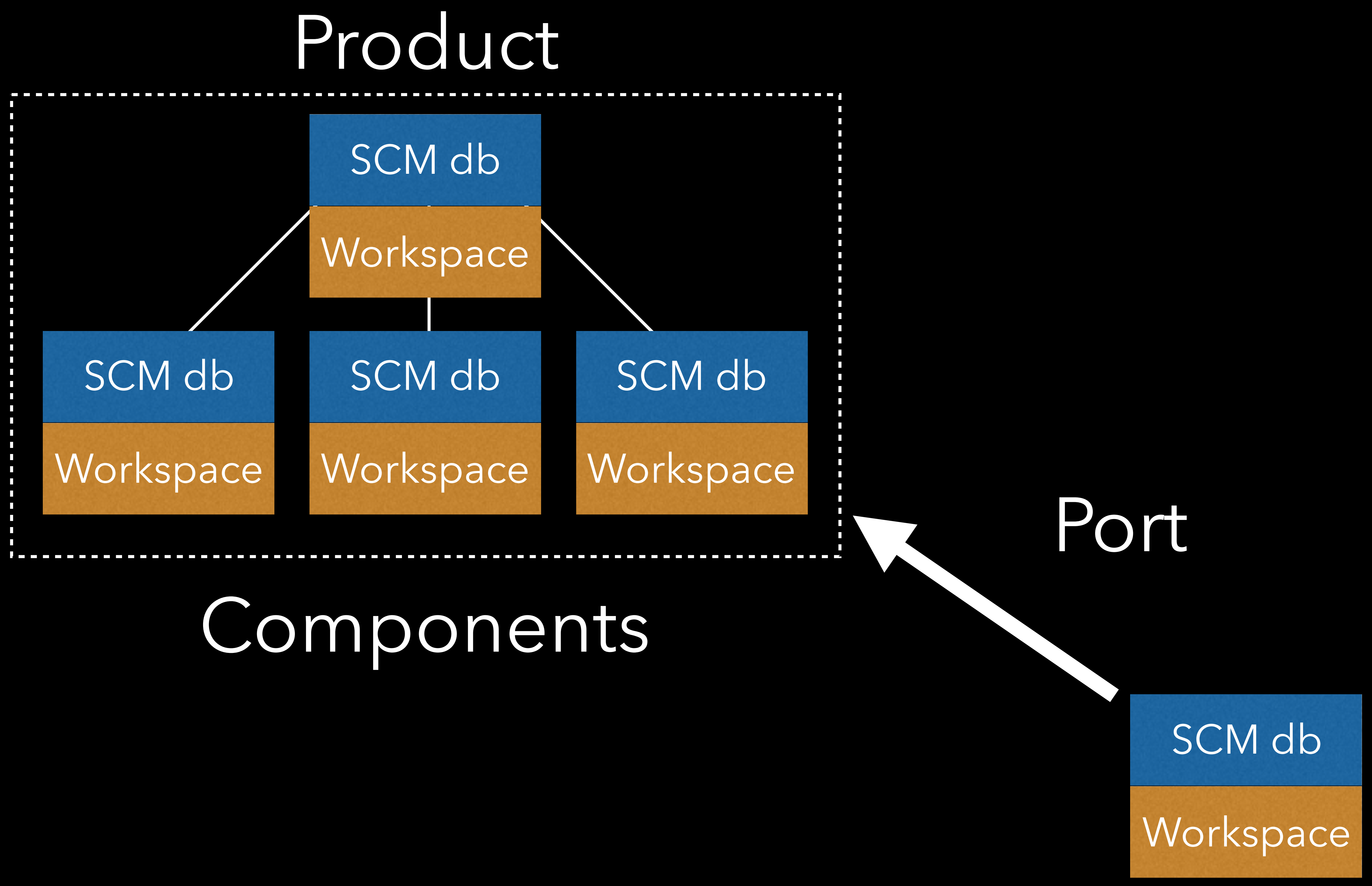
BitKeeper Nested



BitKeeper Nested



BitKeeper Nested



So?

Monorepo

- Goes better with centralized.
- Project boundaries are not clear (files move around).
- Lots of reuse, origin doesn't matter.
- Huge source base and need most of it. No natural boundaries.

Hybrid

- Goes better with distributed.
- Takes atomic commits from monorepo.
- Takes conceptual boundaries from multirepo.
- You can clone components but still work within overall structure.

Multirepo

- Goes better with distributed.
- Project has conceptual boundaries.
- You can work with a small number of components.
- Outsourcing, working with partners.

Don’t let your **tools** determine your **workflow**

Distributed SCM workflows are

MORE FLEXIBLE

And can be sprinkled with enough

centralized JUJU

to make them scale

Ahhh, people ask me questions,
lost in confusion
Well, I tell them there’s no problem,
only solutions

The End

MY BRAIN STILL HURTS