

Knowledge is Power: Understanding git



STEVE SMITH | SENIOR DEVELOPER | @TARKASTEVE

> “Git is fundamentally a content-addressable filesystem with a VCS user interface written on top of it „

Pro Git Book, Section: Git Internals



@tarkasteve



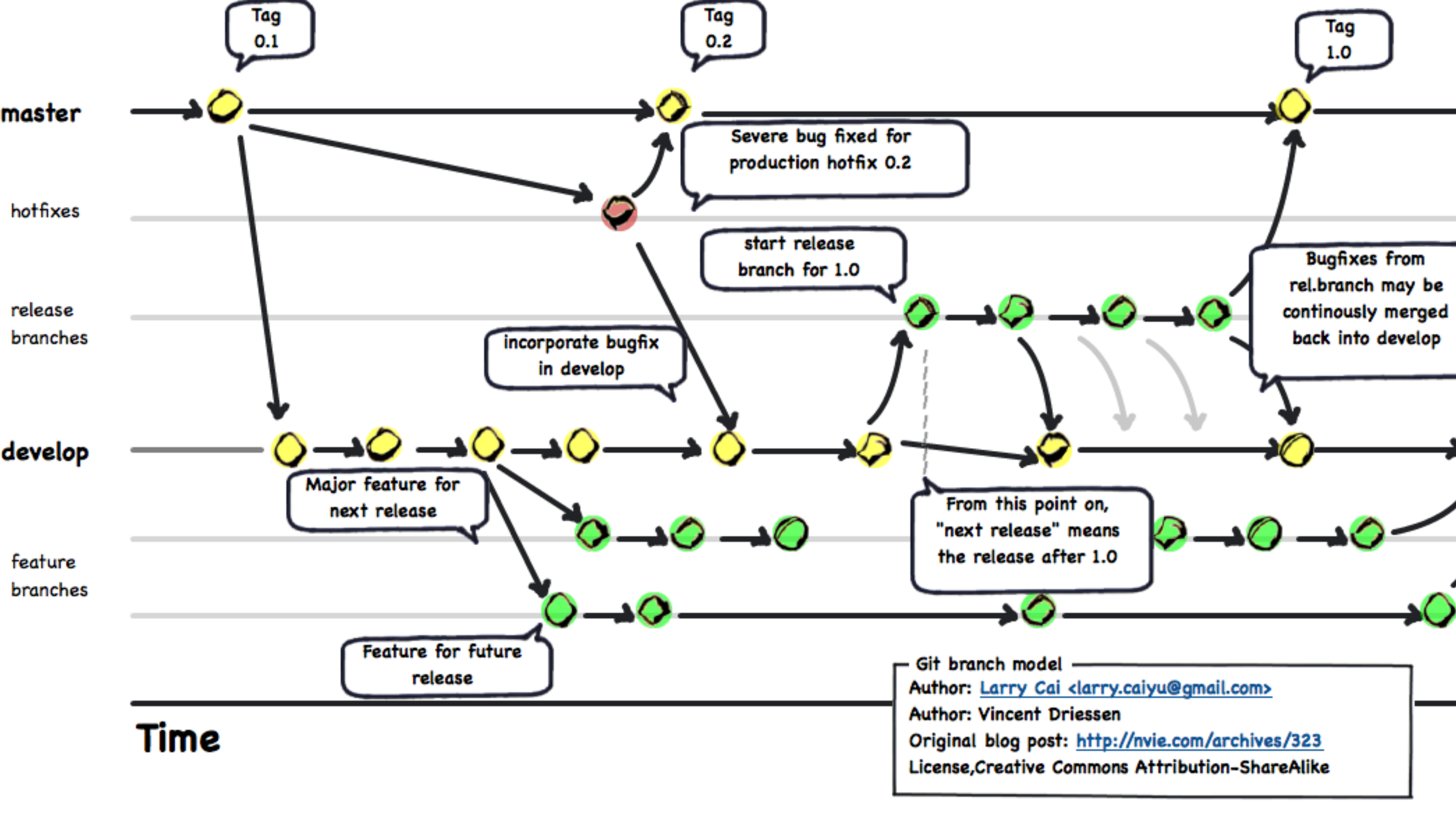
> “It's easy, it's just a directed acyclic graph!”
> If I hear that one more time I may have to punch something!

Emma Jane Hogbin Westby, Git-Merge 2015



@tarkasteve




Tag
0.1
Tag
0.2
Tag
1.0
master
Severe bug fixed for
production hotfix 0.2
hotfixes
start release
branch for 1.0
release
branches
Bugfixes from
rel.branch may be
continously merged
back into develop
incorporate bugfix
in develop
develop
Major feature for
next release
From this point on,
"next release" means
the release after 1.0
feature
branches
Feature for future
release
Time
Git branch model
Author: Larry Cai <larry.caiyu@gmail.com>
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License,Creative Commons Attribution-ShareAlike


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

@tarkasteve

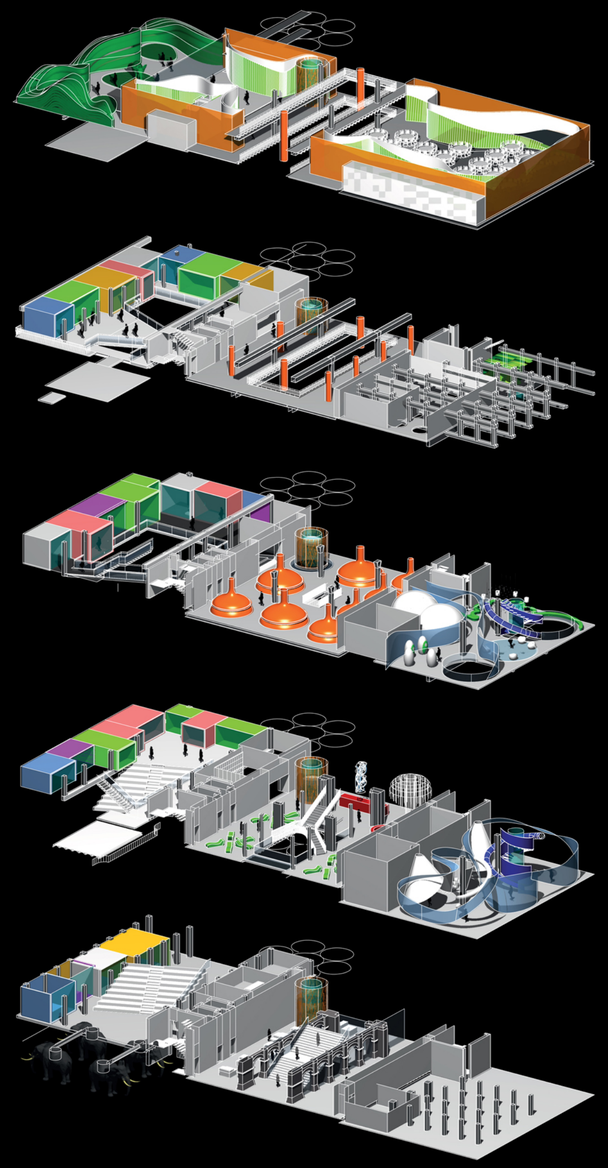
> If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.



@tarkasteve



git internals

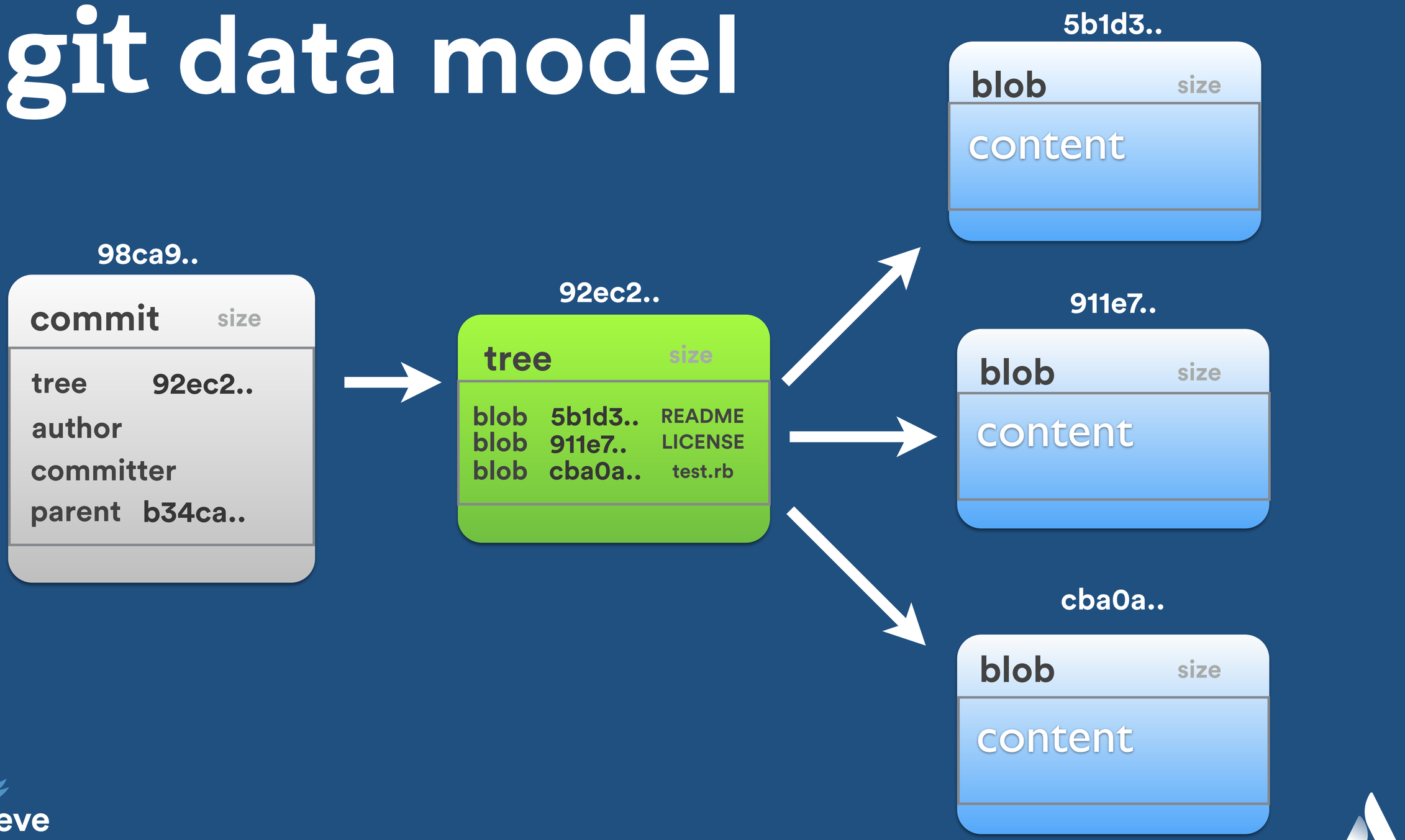


@tarkasteve

Carlsberg


THE BLOB

git data model



@tarkasteve

```
$> git init
$> tree .git/objects
.git/objects
├── info
└── pack

2 directories
```



@tarkasteve



```
$> touch some-file.txt
$> git add some-file.txt
```



@tarkasteve



```
$> tree .git/objects
.git/objects
├── e6
│   └── 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
├── info
└── pack

3 directories, 1 file
```



@tarkasteve

```
$> git commit -m "First commit"
```

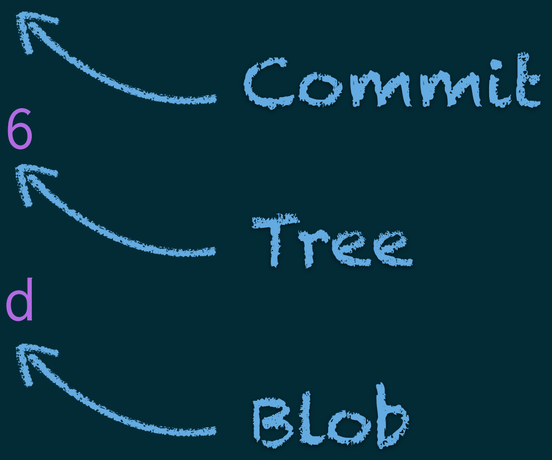


@tarkasteve



```
$> tree .git/objects
.git/objects
├── 13
|   └── 1e360ae1a0c08acd18182c6160af6a83e0d22f
├── 31
|   └── 995f2d03aa31ee97ee2e814c9f0b0ffd814316
├── e4
|   └── 3a6ac59164adadac854d591001bbb10086f37d
├── info
└── pack

5 directories, 3 files
```





@tarkasteve



```
$> echo "// Comment" >> some-file.txt
$> git add some-file.txt
```





```
$> tree .git/objects
.git/objects
├── 13
|   └── 1e360ae1a0c08acd18182c6160af6a83e0d22f
├── 31
|   └── 995f2d03aa31ee97ee2e814c9f0b0ffd814316
├── c1
|   └── 9e6823e34980033917b6427f3e245ce2102e6e
├── e4
|   └── 3a6ac59164adadac854d591001bbb10086f37d

6 directories, 4 files
```

Entirely new BLOB



@tarkasteve



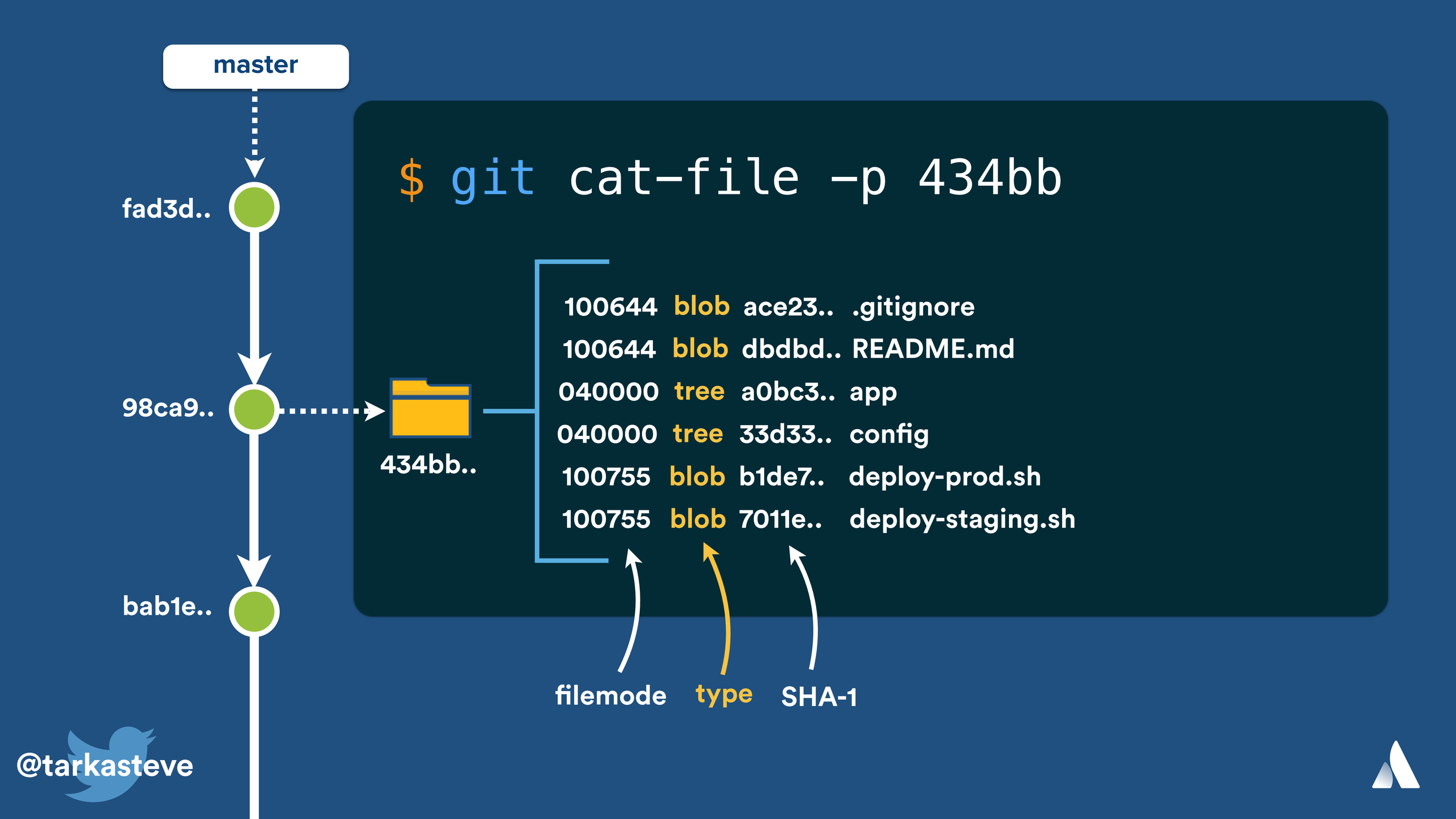


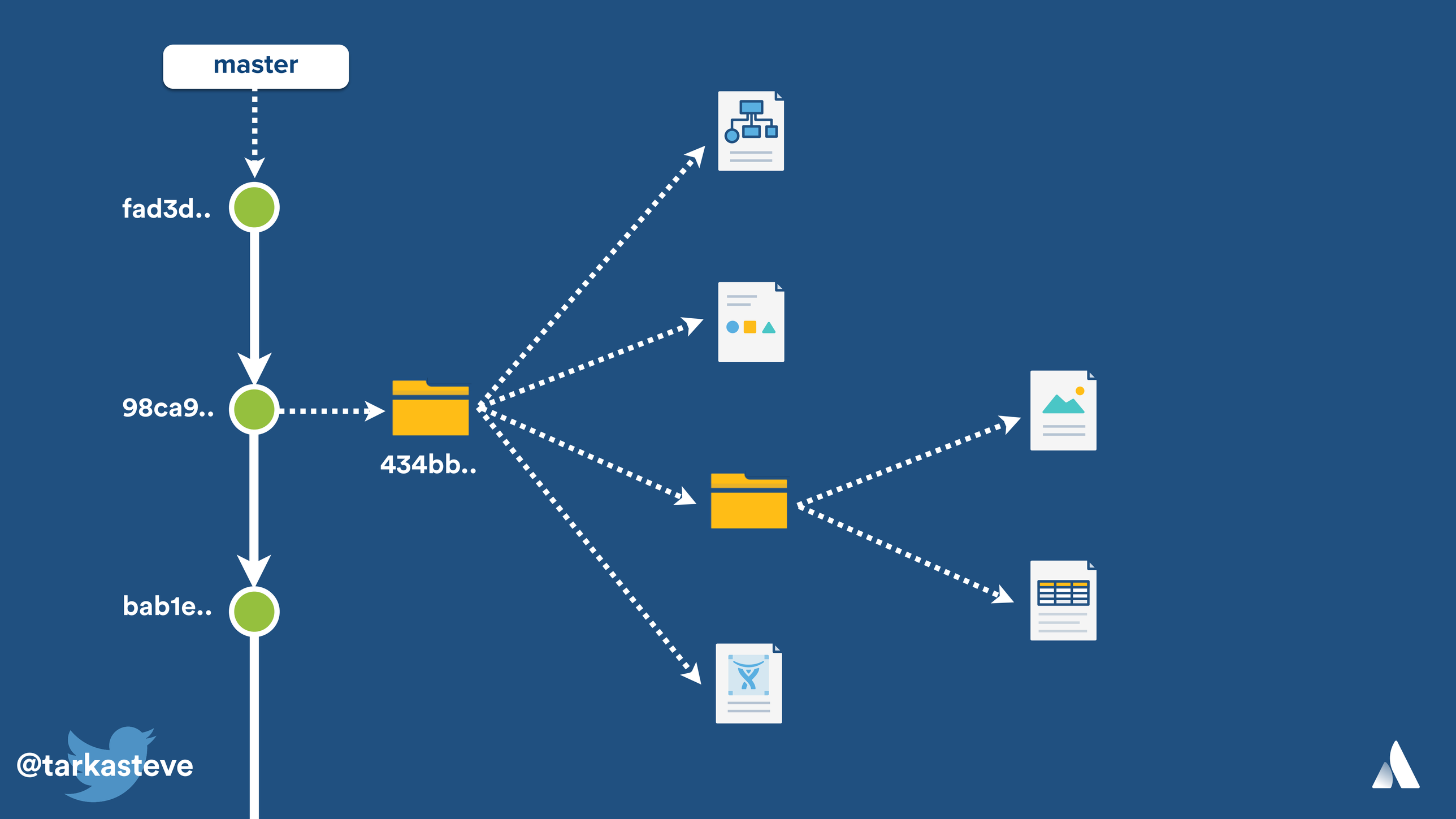

```
$ git cat-file -p 98ca9
```

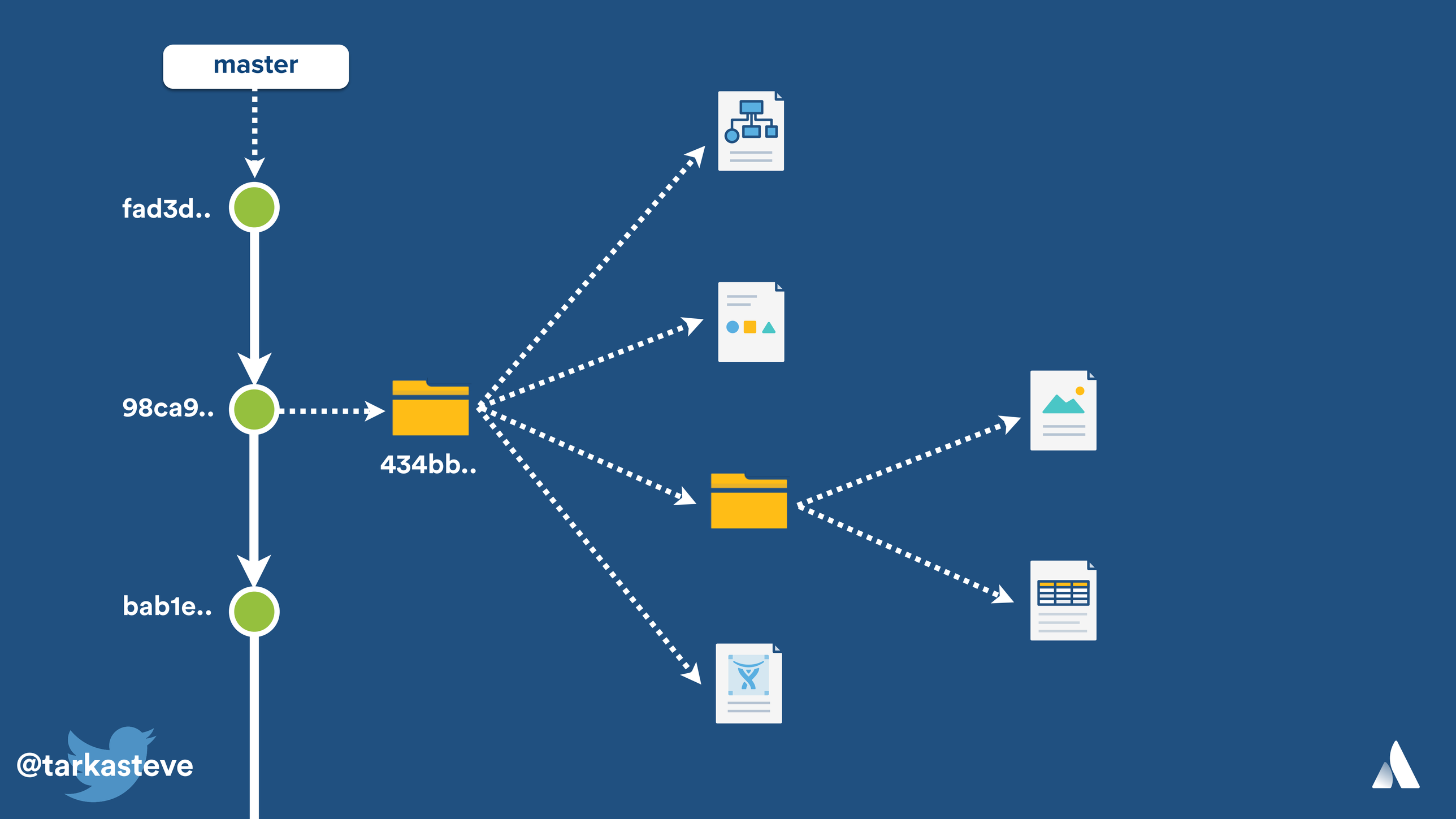
```
parent      bab1e..
tree        434bb..
committer   Tim P <kannonboy@...> 1455209277 -0800
author      Tim P <kannonboy@...> 1455209277 -0800

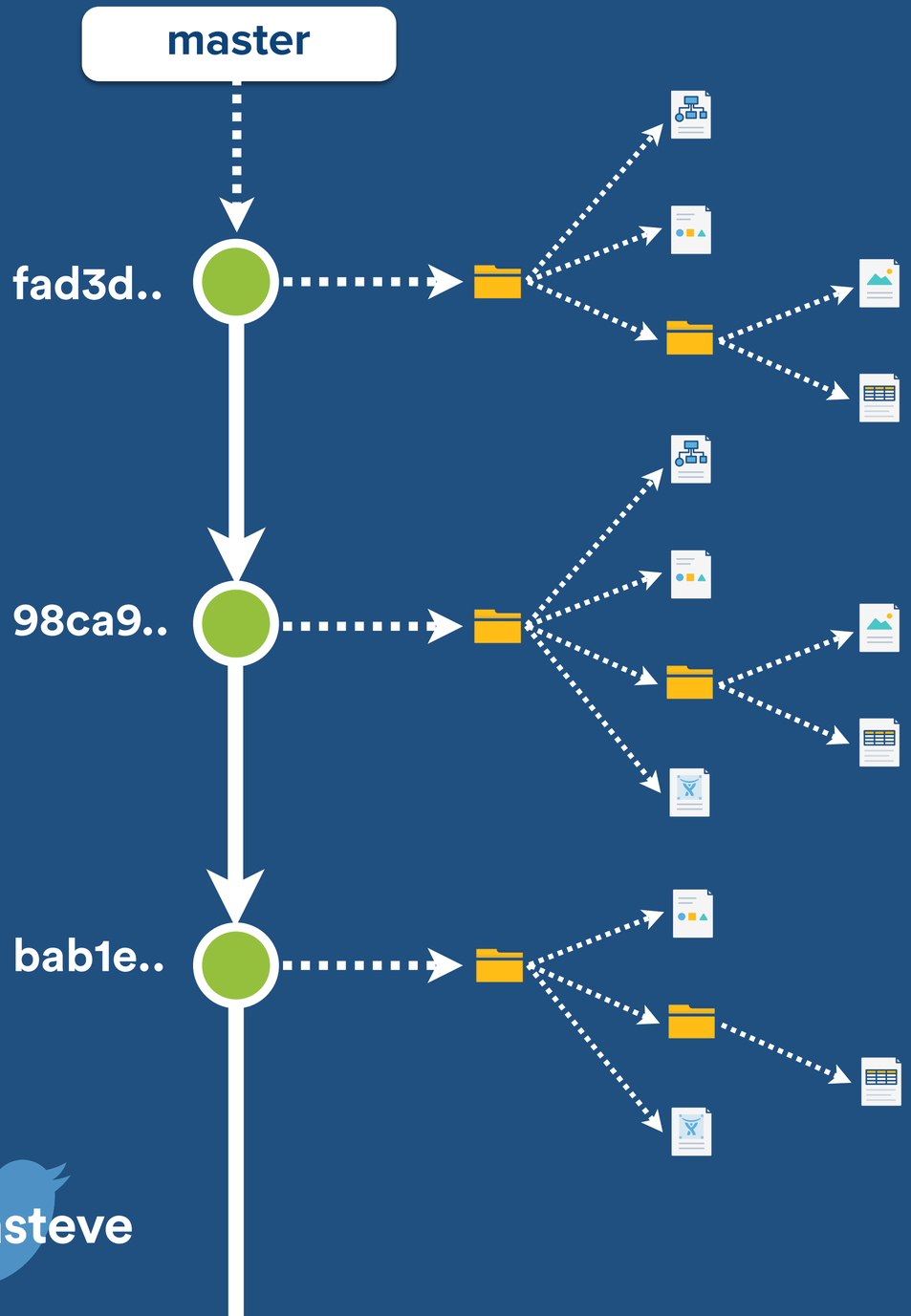
            My life is my commit message.
```

@tarkasteve

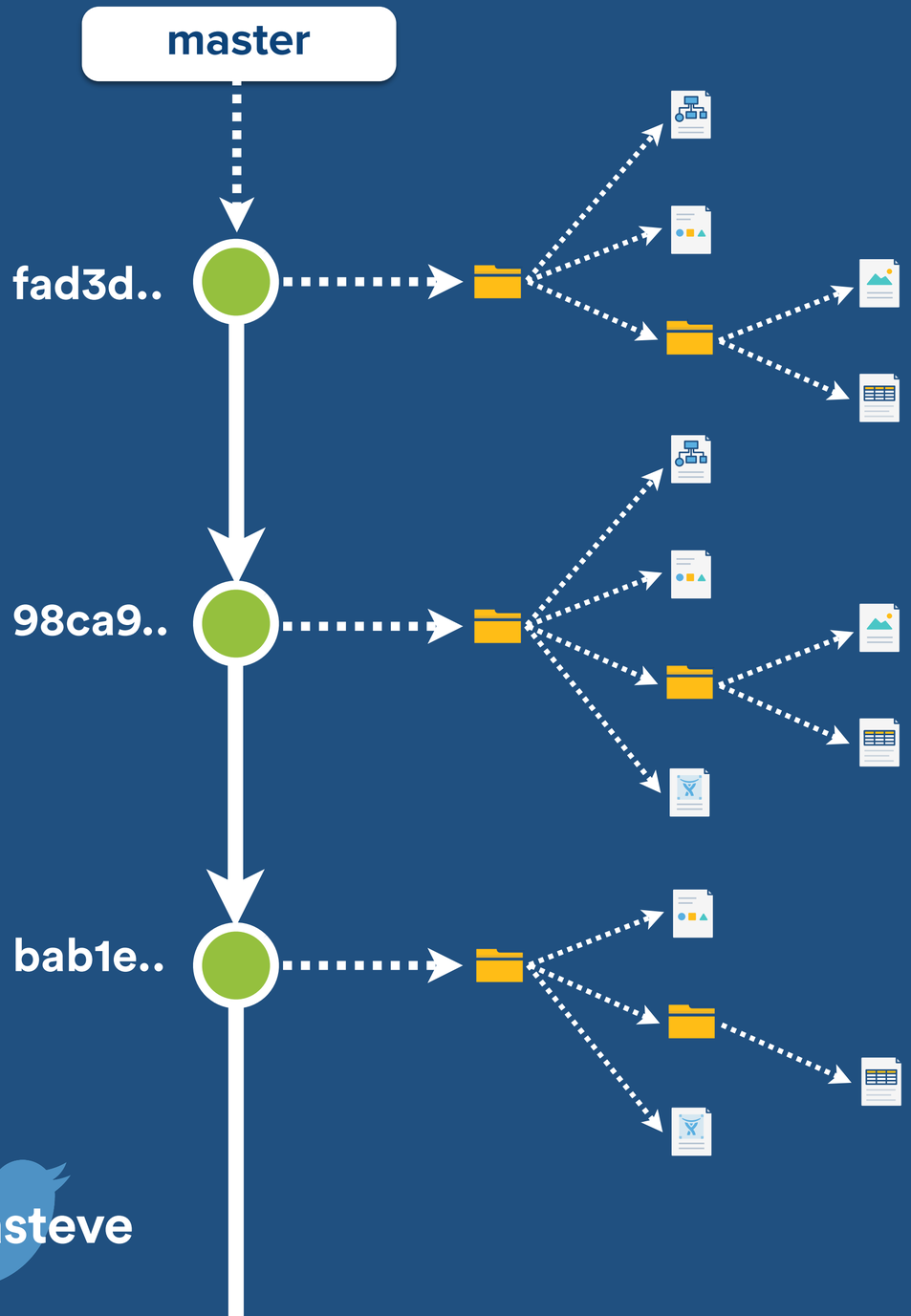

master
fad3d..
98ca9..
bab1e..
$ git cat-file -p 434bb
434bb..
100644 blob ace23.. .gitignore
100644 blob dbdbd.. README.md
040000 tree a0bc3.. app
040000 tree 33d33.. config
100755 blob b1de7.. deploy-prod.sh
100755 blob 7011e.. deploy-staging.sh
filemode
type
SHA-1
@tarkasteve


master
fad3d..
98ca9..
434bb..
bab1e..
@tarkasteve

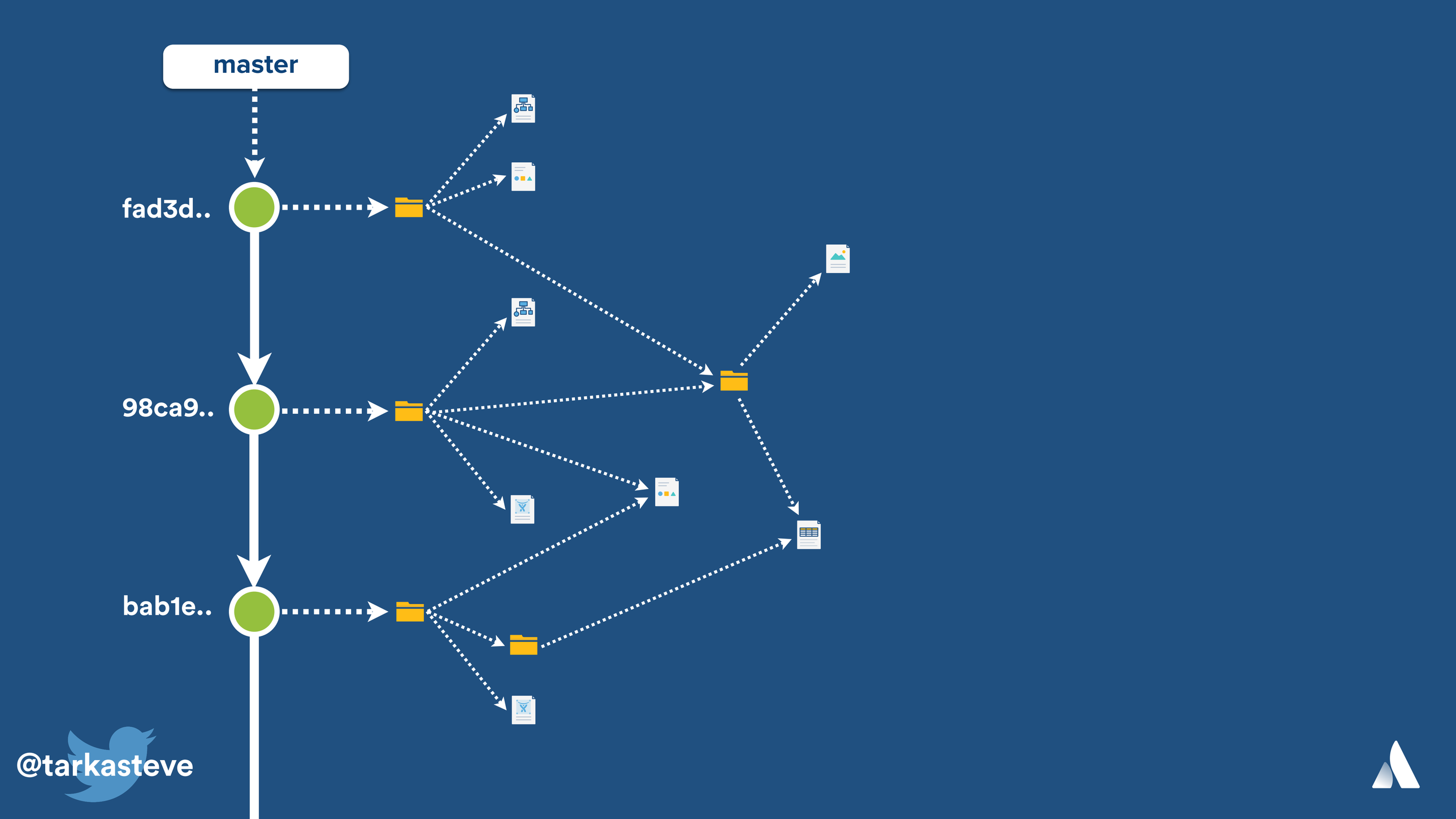

master
fad3d..
98ca9..
434bb..
bab1e..
@tarkasteve



@tarkasteve



@tarkasteve



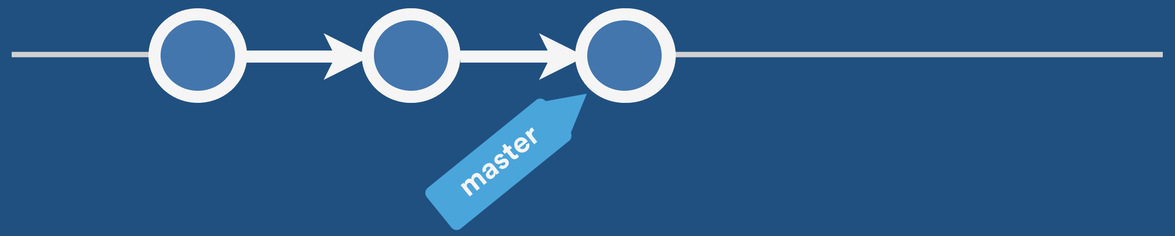
@tarkasteve



Refs and Branches And Tags

What is a ‘ref’?

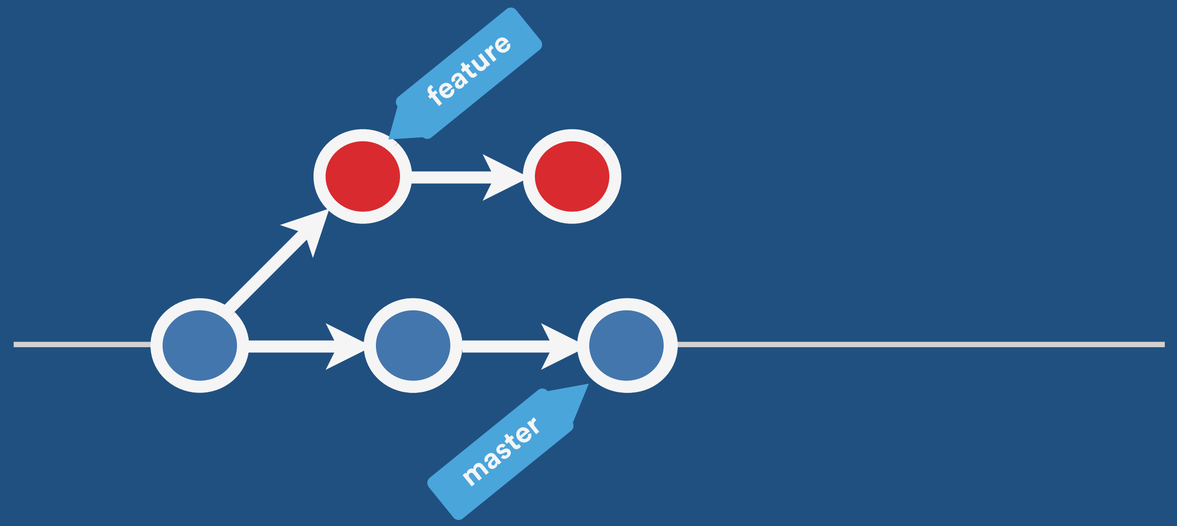
A ref is just a pointer to an object



@tarkasteve

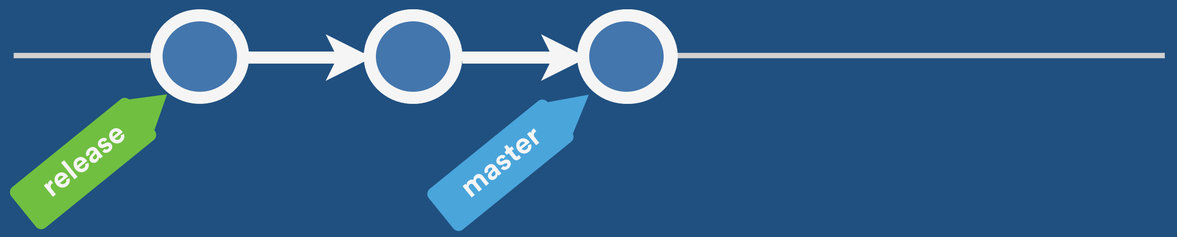
What is a ‘branch’?

A branch is divergence from a common point, and ref to a commit, the “HEAD”



What is a ‘tag’?

A tag is just special ref used a mark
a commit in the history



```
$> git tag a-tag -m”A tag”
$> git branch a-branch

$> tree .git/refs/
.git/refs/
├── heads
|   ├── a-branch
|   └── master
└── tags
    └── a-tag

$> cat .git/refs/heads/a-branch
c13e27cdfd15c5acdcd8b510eefed7be68c41c8e
```



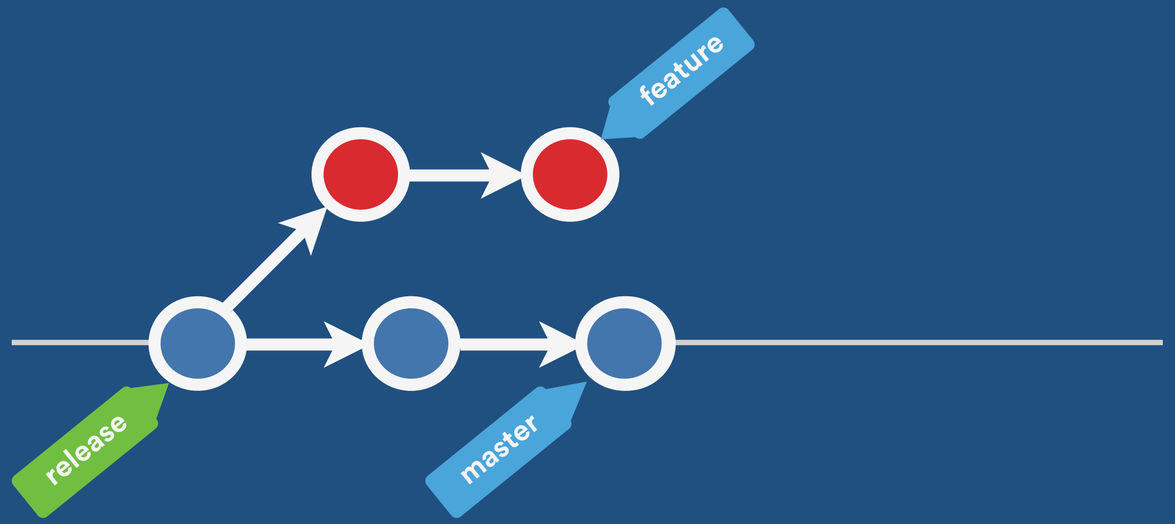
@tarkasteve



What is a ‘reset’?

Manipulates the branch ‘head’

```
$> git reset --hard feature^              # ‘^’ means ‘parent’
```



@tarkasteve

Reflog keeps a history

```
$> git reflog
0c35628 HEAD@{1}: reset: moving to HEAD^
6cc6637 HEAD@{2}: commit: Add B
0c35628 HEAD@{3}: merge: Merge made by the 'recursive' strategy.
e0c0d65 HEAD@{4}: cherry-pick: A
80bb854 HEAD@{5}: checkout: moving from alpha to master
5044136 HEAD@{6}: commit: A
80bb854 HEAD@{7}: checkout: moving from master to alpha
80bb854 HEAD@{8}: commit (initial): 1
```

(Only 90 days by default though!)



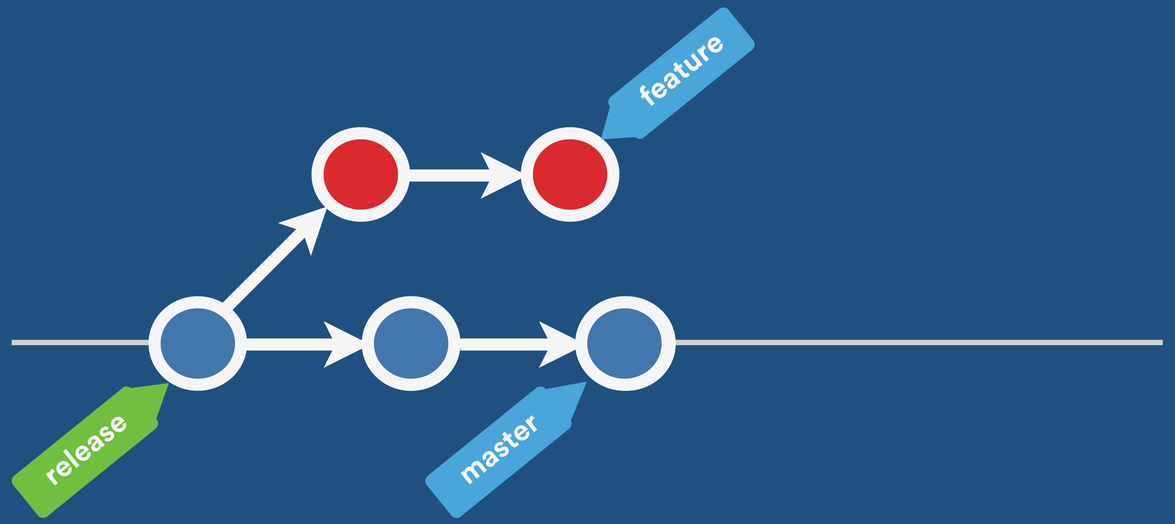



Keeping things
clean

What is a ‘gc’?

Orphaned objects are eligible for removal

```
$> git reset feature^                    # ‘^’ means ‘parent’
$> git gc --prune=all
```

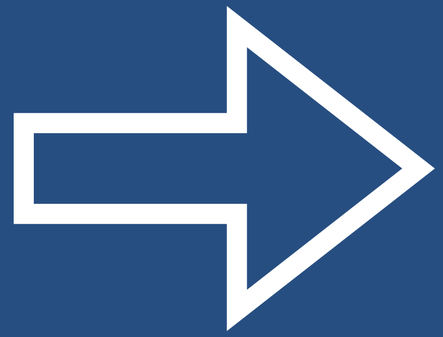


@tarkasteve

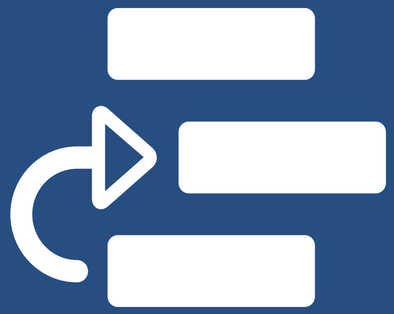
GC also packs objects

Loose Objects

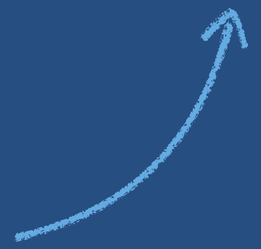




Packfile



1. zlib compressed
2. Delta encoded



@tarkasteve



```
$> tree .git/objects
.git/objects
├── info
|   └── packs
└── pack
    ├── pack-7475314b451a882d77b1535d215def8bad0f4306.idx
    └── pack-7475314b451a882d77b1535d215def8bad0f4306.pack

2 directories, 3 files
```



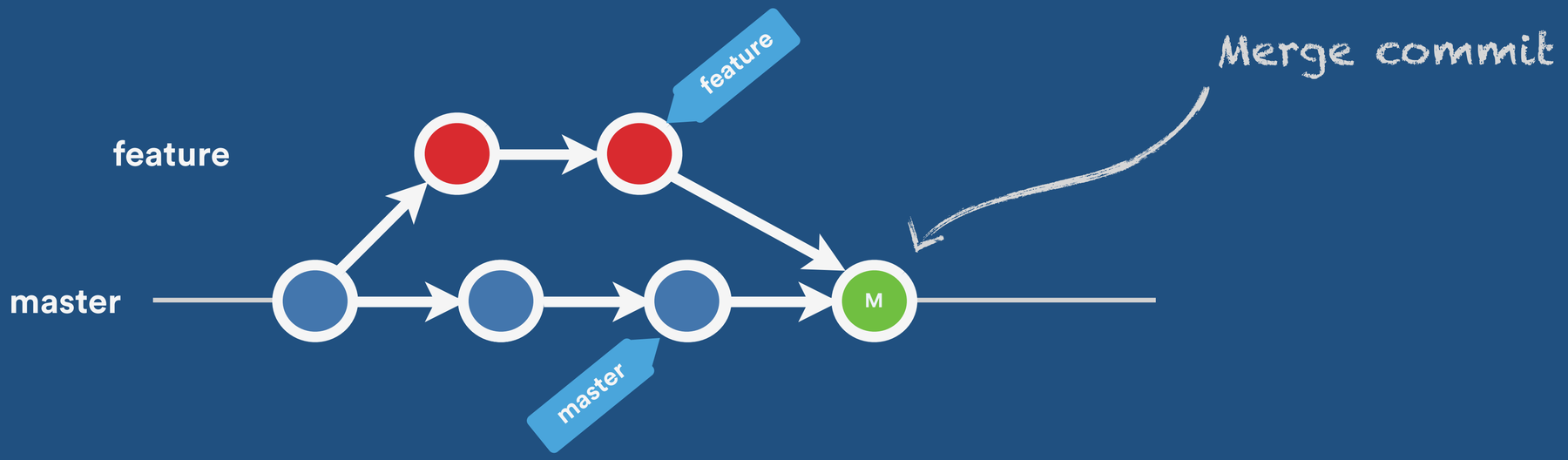
@tarkasteve




git merge

What is a merge?

merges keep the context of
the feature's commits



@tarkasteve

Anatomy of a merge

```
$> git cat-file 3680d8c8fd182f97cb0e75045e2fed5c7b7613ed
tree f362c42032aff677c1a09c3f070454df5b411239
parent 49a906f5722ad446a131778cea52e3fda331b706
parent bd1174cd0f30fe9be9efdd41dcd56256340f230e
author Marcus Bertrand <mbertrand@atlassian.com> 1409002123 -0700
committer Marcus Bertrand <mbertrand@atlassian.com> 1409002123 -0700

Merge branch 'foo/mybranch'
```





merge is better in **git**

1. The merge is **local**
2. **git** knows the **ancestry**
3. Powerful merge **strategies**

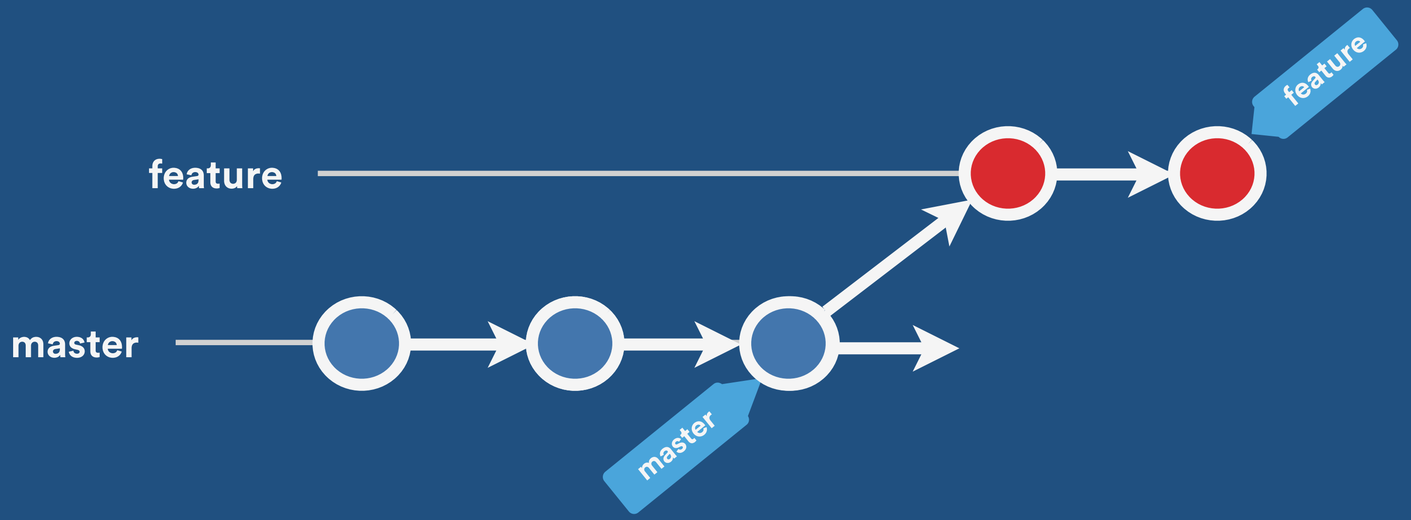


@tarkasteve



What is a fast-forward merge?

It will just shift the HEAD tag



@tarkasteve

What are ‘merge strategies’?

git has breadth of choice on
how to merge changes!

resolve **recursive** **octopus**

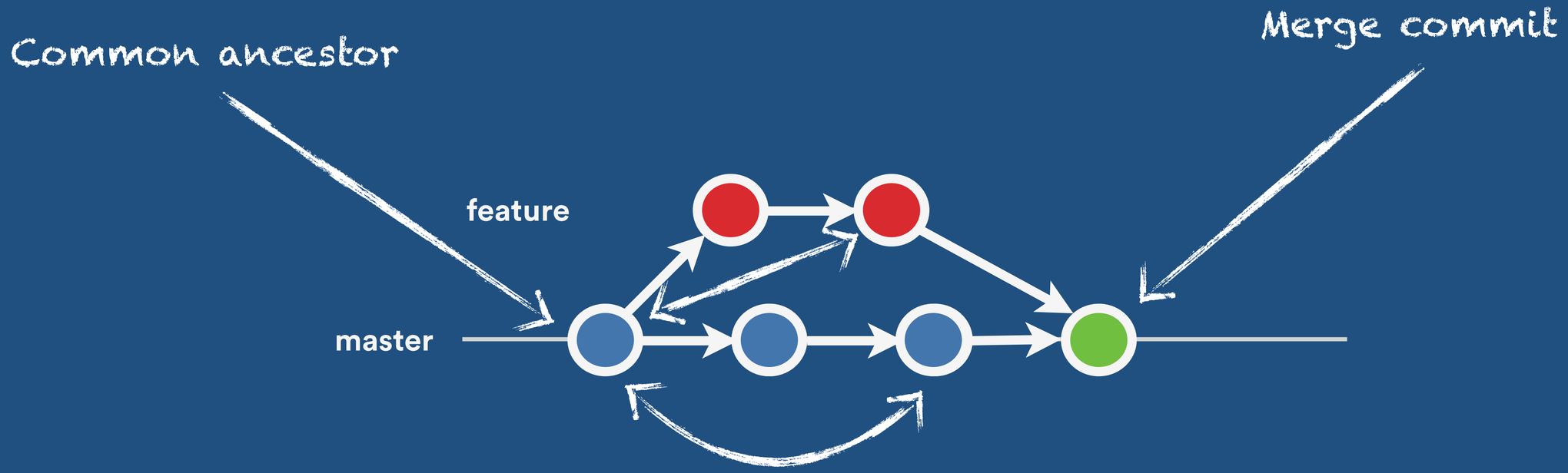
ours **subtree** **custom?**





merge strategy: resolve

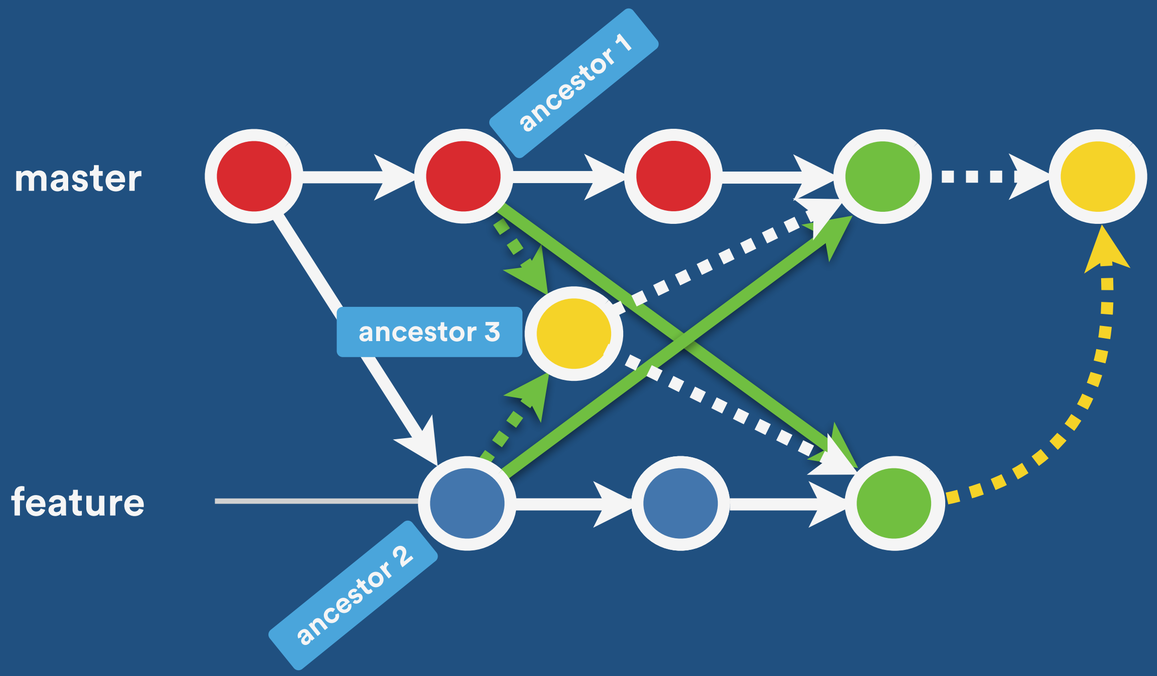
Three-way merge of the ancestors



@tarkasteve

merge strategy: recursive

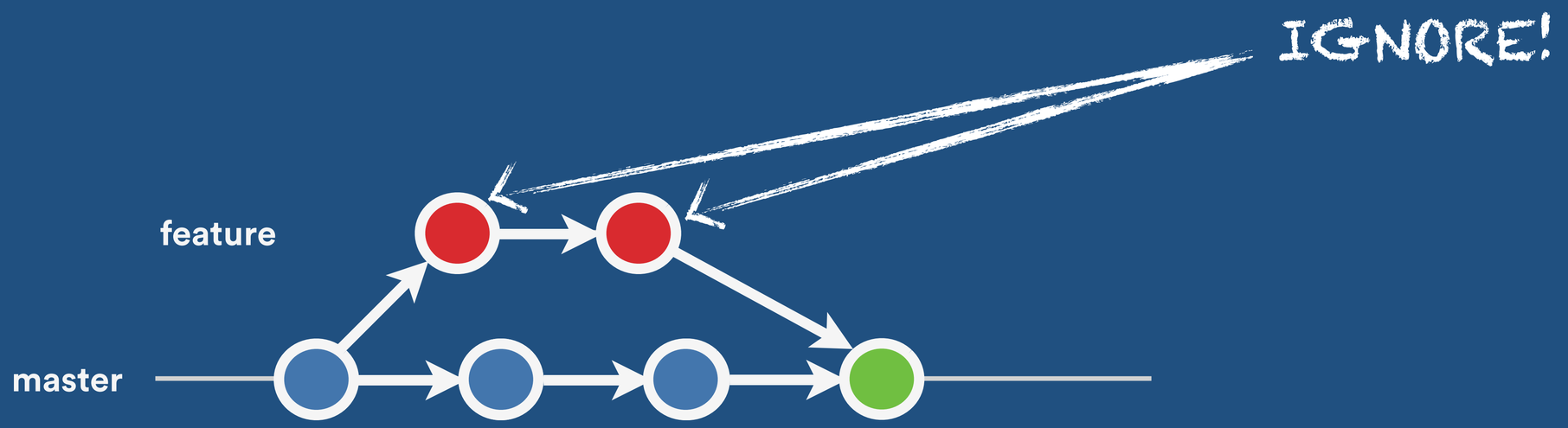
3-way merge but climbs the ancestry tree recursively if there are multiple ancestors



@tarkasteve

merge strategy: ours

Records a merge but skips incoming changes



@tarkasteve

use “ours” to promote more recent branches

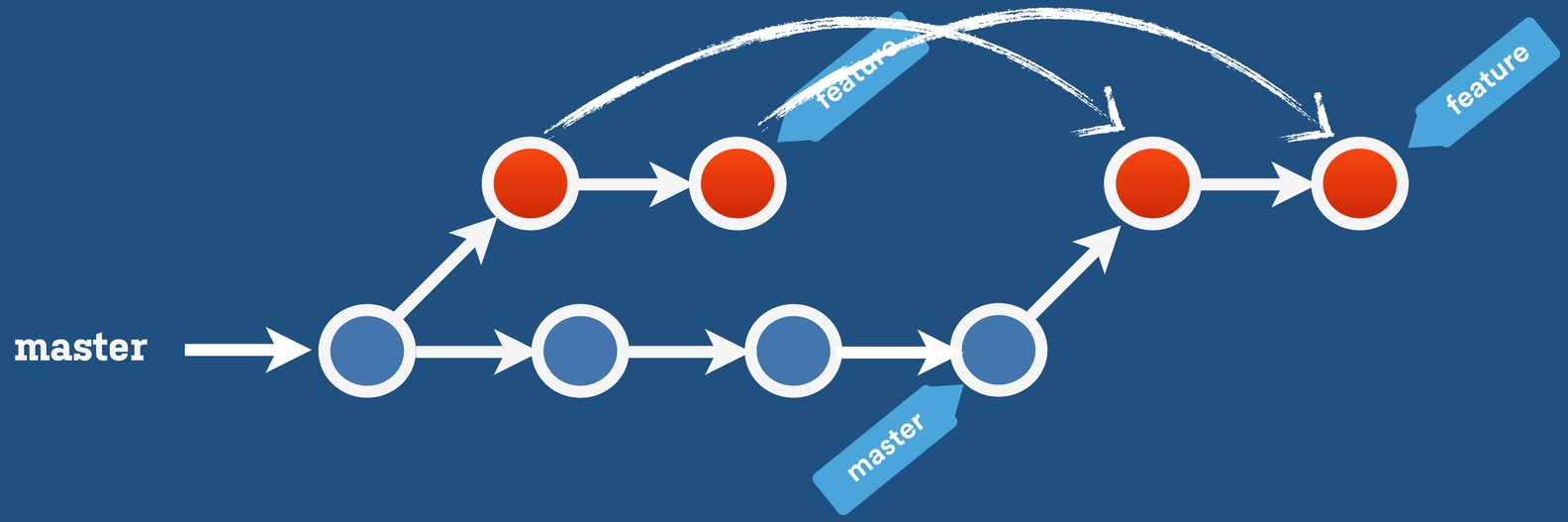


@tarkasteve



What is a Rebase?

It's a way to replay commits,
one by one, on top of a branch



@tarkasteve

What is an --interactive rebase?

Helps you clean up your private branches before publishing them

 **pick**

 **reword**

 **edit**

 **squash**

 **fixup**

 **exec**





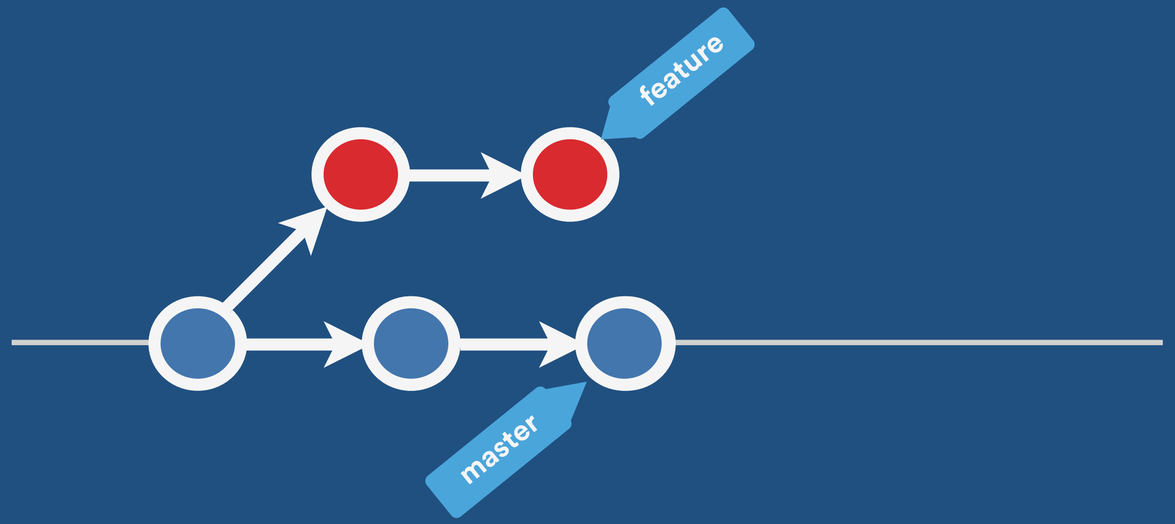
Getting out of trouble



reset —hard ohshit

—hard removes all staged and working changes!

```
$> git reset --hard feature^          # ‘^’ means ‘parent’
```



@tarkasteve

Reflog!

```
$> git reflog
0c35628 HEAD@{1}: reset: moving to HEAD^
6cc6637 HEAD@{2}: commit: Add B
0c35628 HEAD@{3}: merge: Merge made by the 'recursive' strategy.
e0c0d65 HEAD@{4}: cherry-pick: A
80bb854 HEAD@{5}: checkout: moving from alpha to master
5044136 HEAD@{6}: commit: A
80bb854 HEAD@{7}: checkout: moving from master to alpha
80bb854 HEAD@{8}: commit (initial): 1
```

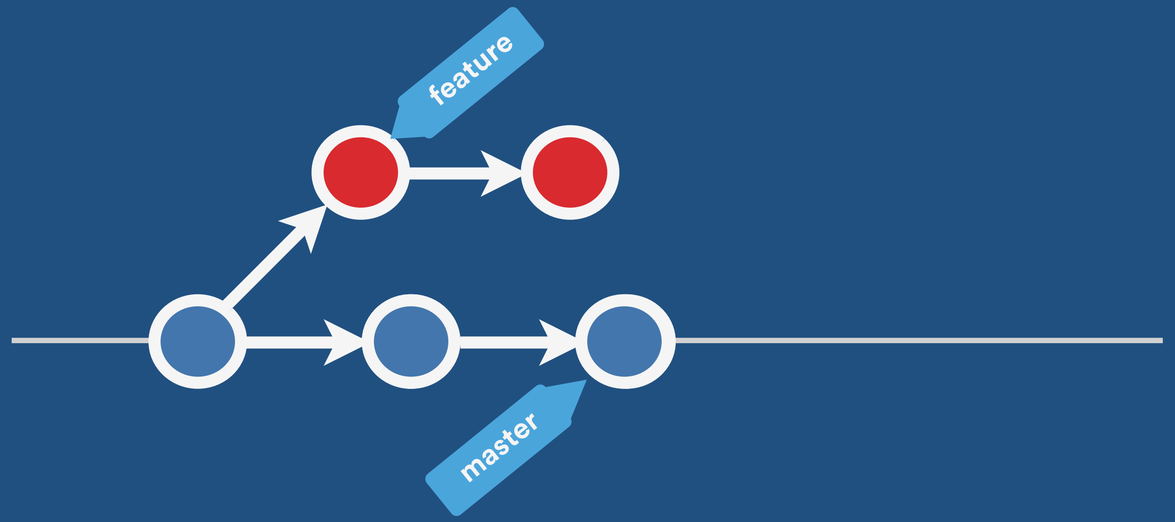




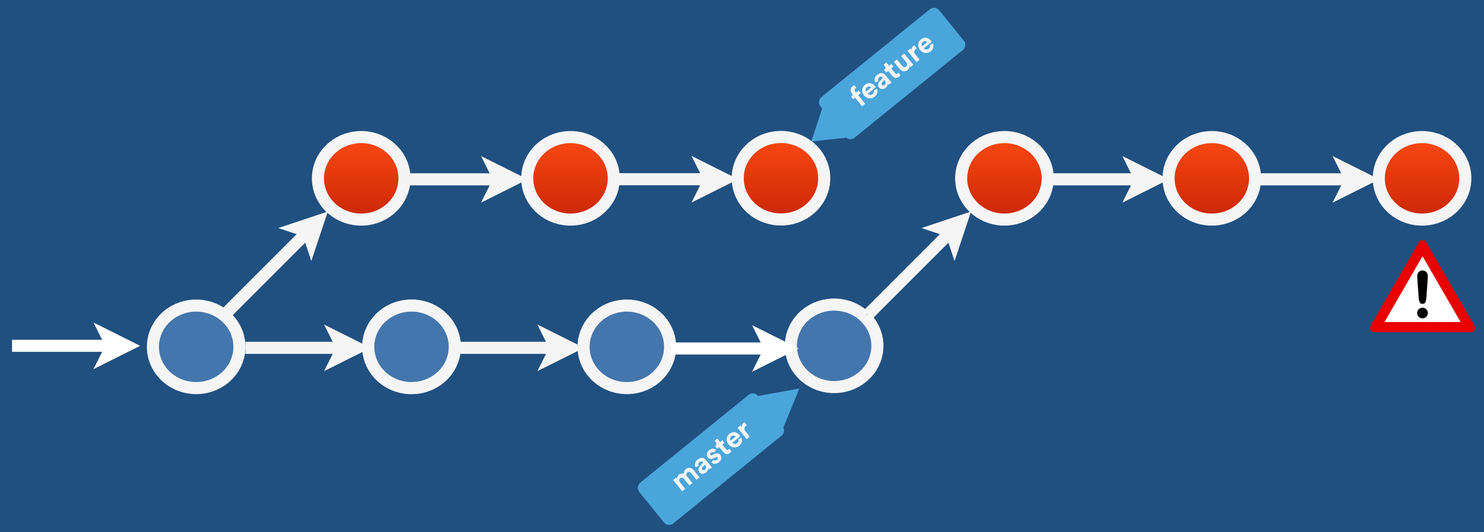
Reflog+Reset = Redo

Reset back to our commit!

```
$> git reset --hard 6cc6637
```



Rebase Broke The Build!



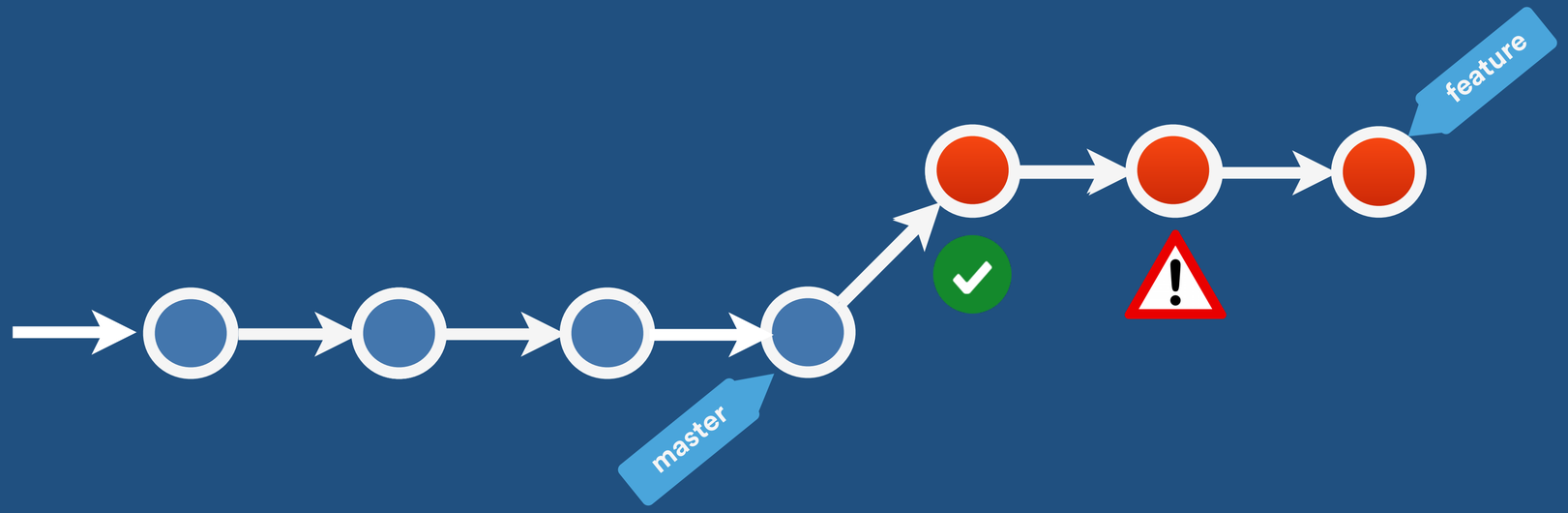
@tarkasteve

Fix with rewind/replay

Reflog to reset.

rebase --exec to test each step.

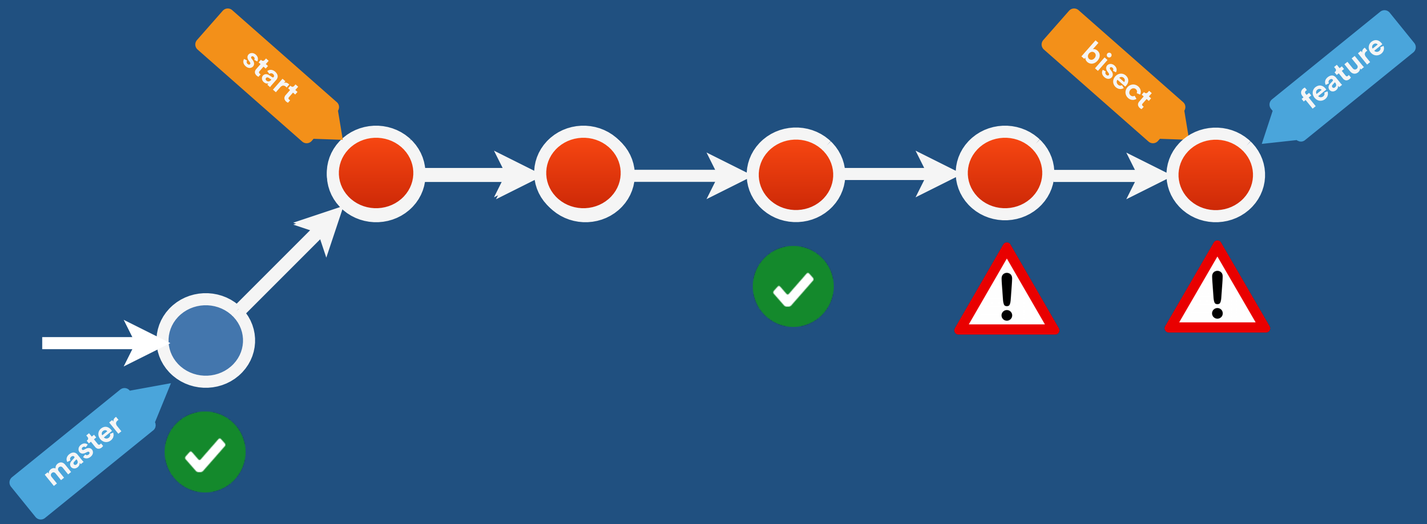
```
$> git rebase master  --exec “make test”
```



@tarkasteve

Fix in place: bisect

```
$> git bisect start
$> git bisect good master
$> git bisect run make test
```

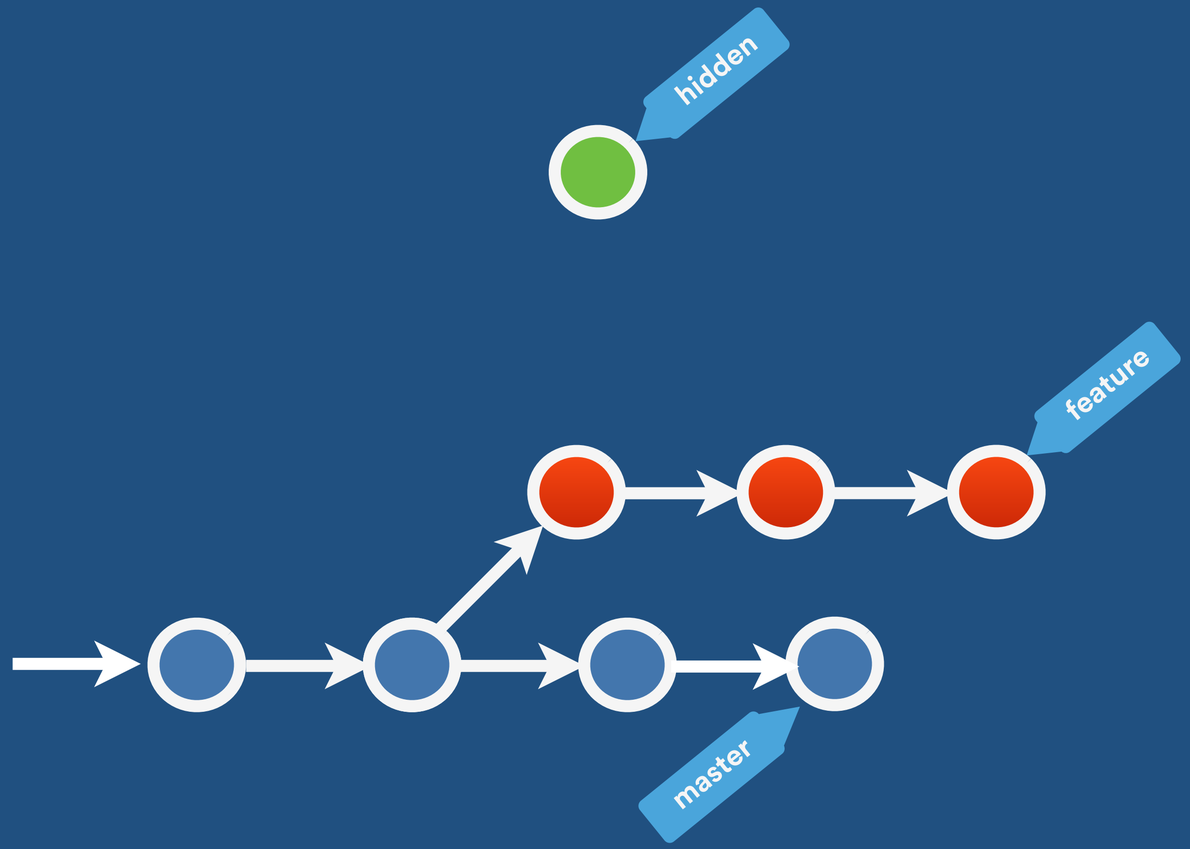


@tarkasteve



© http://www.amigosdosbichos.org/

Hiding files in Git



@tarkasteve

Hiding files in Git

```
$> git hash-object -w hidden.txt
929cb9f0f8bde8e8b9dc4a69cf3459d2326191c8
$> git tag hidden 929cb9f0f8bde8e8b9dc4a69cf3459d2326191c8
```



Retrieving

```
$> git cat-file -p hidden
Oh freddled gruntbuggly,
Thy micturations are to me, (with big yawning)
As plurdled gabbleblotchits, in midsummer morning
On a lurgid bee,
That mordiously hath blurted out,
Its earted jurtles, grumbling
Into a rancid festering confectious organ squealer.
Now the jurpling slayjid agrocrustles,
Are slurping hagrilly up the axlegrurts, le me,
```





Actual use...

```
$ ~/git> git cat-file -p junio-gpg-pub
```

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQIcBAABAgAGBQJUbnV7AAoJELC16IaWr+bLvjYP+wSOG8rp1y77ExHDJQj7HBLm
9PVQIb70tkiBxAGUpVNNbaoJQBuMAgFdT4Baj8CIo2jdzDeeqbjtcdy/XsHZ3IMl
7VRm9cI8Veh5nPc9U3z6XYTMmDhxxSsnCONQECYYFEJ5QGhnwkpOUTDYPRmnghWu
4JSDyWX0RLNXVPZkoP/zwVbv74Exx1Gz24p6fPlc4iETrKNHzzSM07CPKlAmXRHT
Ozh/NXfiE3OZSFDUNKkzHQJHtyFEPpVw/aUIyObvOcSYy6ZxQ+56U9h24D1mW8kY
w8BPdhzQ5d9nB/zpX83/cqsv6IXsK62ls8Qj51MfNM902D0pghVBInjR2xekWx+b
AO5VMaRiC+hgHvyzAM2nXuDfTUssd9ryYIx/plfw0gcKZ41S3HzHTiXYqRqd4O4I
lkFlWL11no7od+T9T2nap5wfdLliQpMxxZFLsqH67LzAvHxg/NjtIMXDxUfo++tV
qBR153FxcJX8AOcQo3zY7WotbmgS+8JPron3XvjXNOlz9HAxR2+orxhEC3pTVfeC
/aEZzs+ub9pV6ZQzVdfHzRZbtiN4Zw2ycW+EoXvlUCwNy72efaBRpKjQl4zJHKfD
LAwljdrenm8X6RjBKd/vPyTPSHx1AtSoXqseVw8Ly+qqoXnhsh2N0Xe2Ozx3D0lk
PtscHvD+diDMRLbLgK/k
=Sxgg
-----END PGP SIGNATURE-----
```



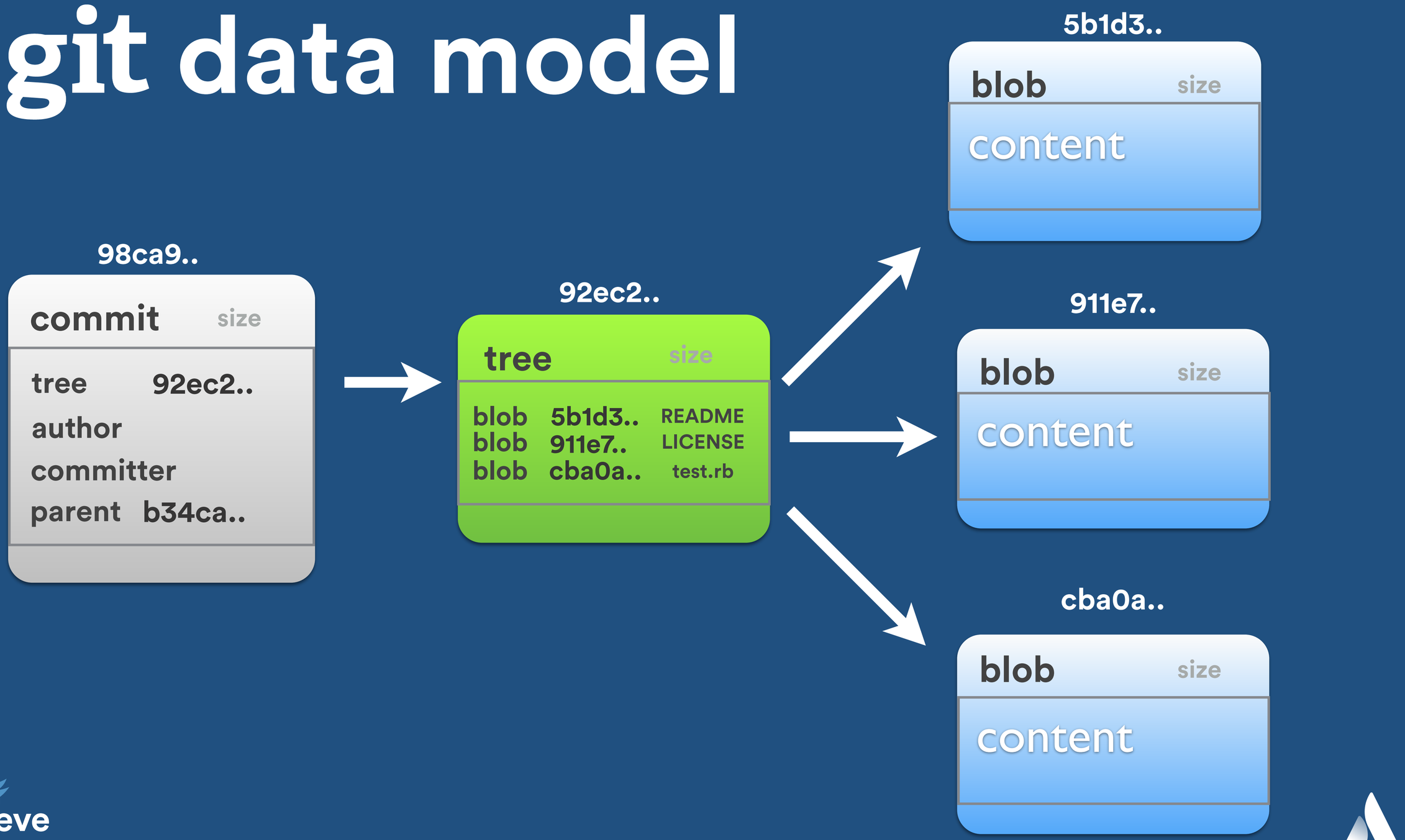
@tarkasteve



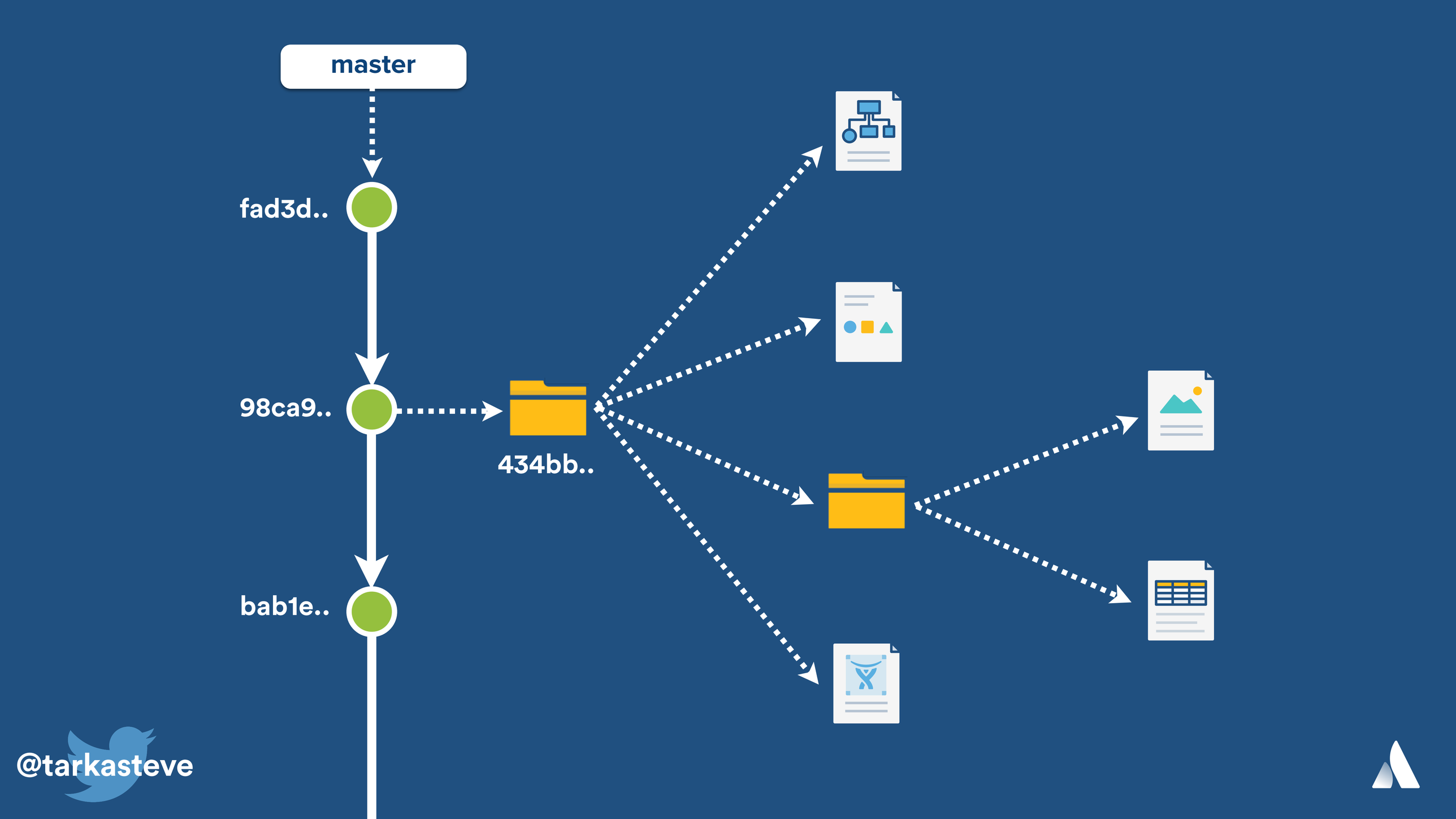


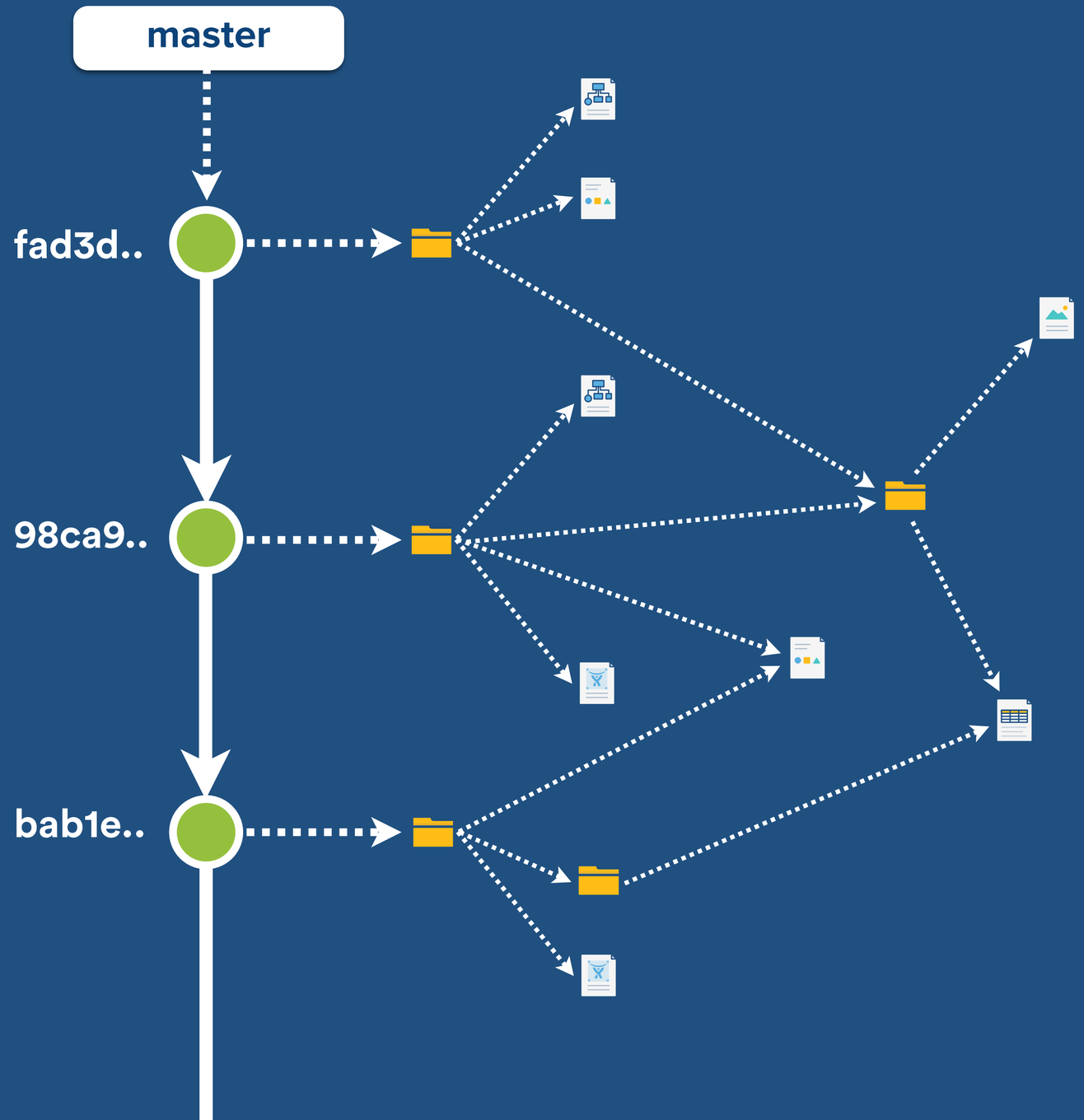
Out of Time

git data model



@tarkasteve


master
fad3d..
98ca9..
434bb..
bab1e..
@tarkasteve



@tarkasteve



STEVE SMITH | SENIOR DEVELOPER | @TARKASTEVE