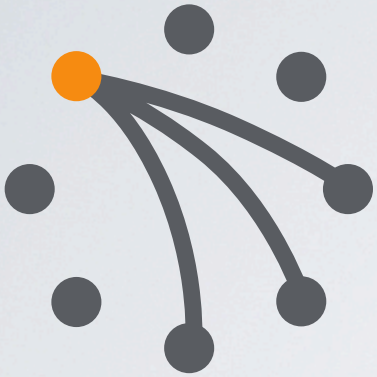


riak

from the inside

Justin Sheehy Basho Technologies

Riak is
a scalable, highly-available, distributed
open-source key/value store.

basho

riak

Riak is
a scalable, highly-available, distributed
open-source key/value store...

...but that's not what I'm here to tell you about.

basho

riak

Riak is
a system built using Erlang/OTP for
robustness, flexibility, and simplicity.

Riak is
a system built using Erlang/OTP for
robustness, flexibility, and simplicity...

...so today, I'll talk about how that helped us.

non-topics

Riak Core's Implementation Details

-- essential distributed systems infrastructure

Riak Features

-- anti-entropy: hinted-handoff and read-repair

-- javascript map/reduce

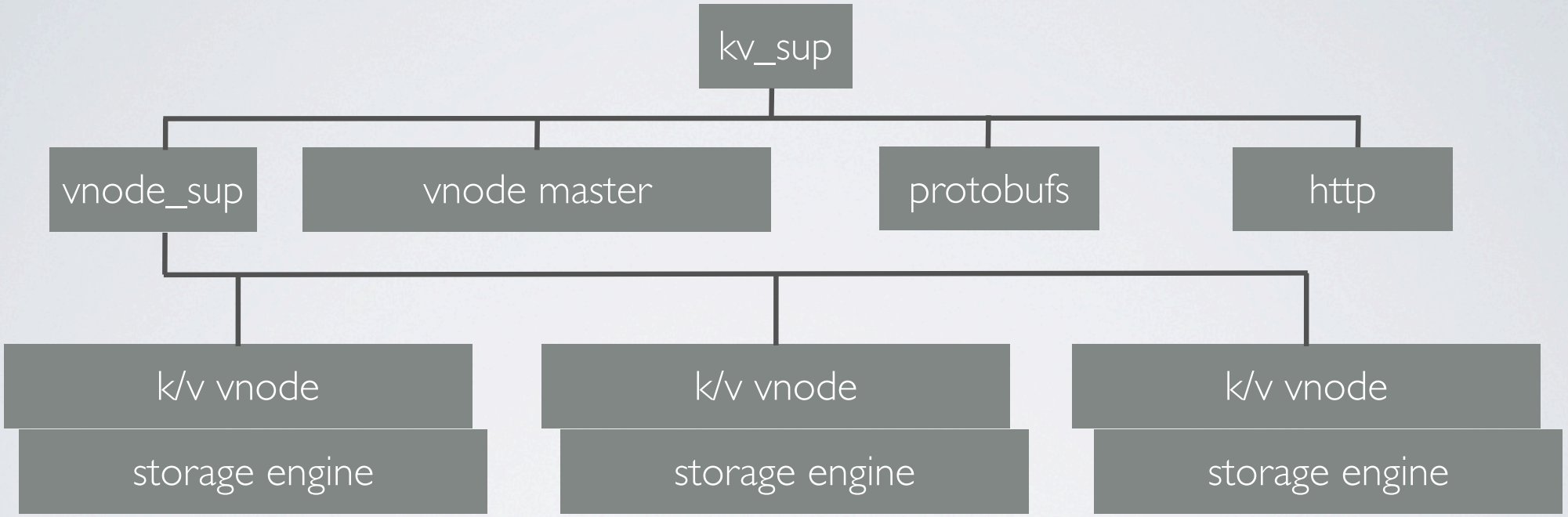
-- storage subsystems: bitcask, innostore...

The Riak key/value stack:

basho

riak

It's not only a "stack", of course:

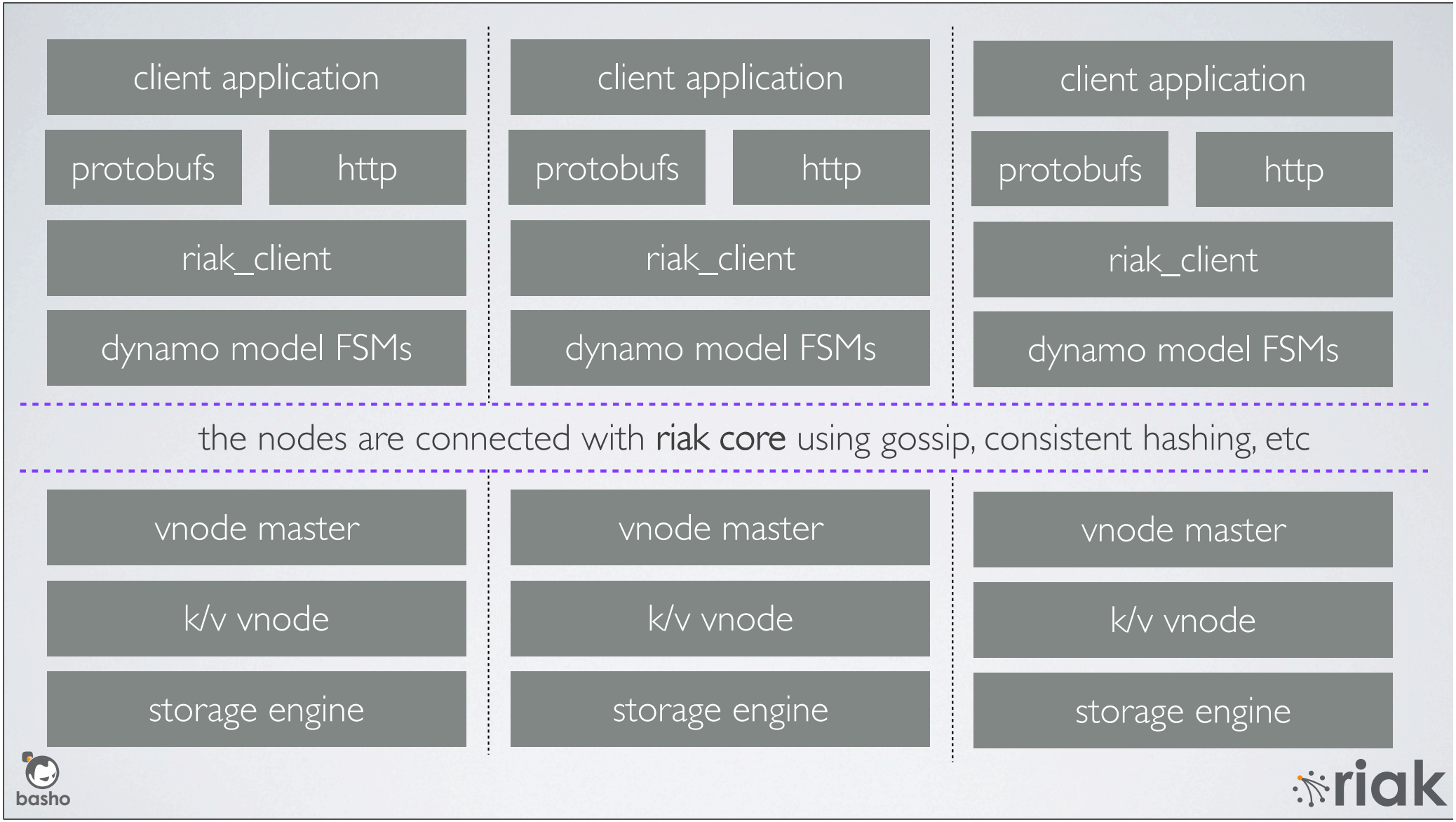


(this represents a part of the k/v section of the supervisor tree)

The Riak key/value stack:

basho

riak


client application
client application
client application
protobufs
http
protobufs
http
protobufs
http
riak_client
riak_client
riak_client
dynamo model FSMs
dynamo model FSMs
dynamo model FSMs
the nodes are connected with riak core using gossip, consistent hashing, etc
vnode master
vnode master
vnode master
k/v vnode
k/v vnode
k/v vnode
storage engine
storage engine
storage engine
basho
riak

let's start at the top

more than one way to access key/value data

interchangeable: use them both

get/put is representation transfer

HTTP is a great transfer protocol

- ubiquity
- flexibility
- interoperability

This is why we wrote webmachine!

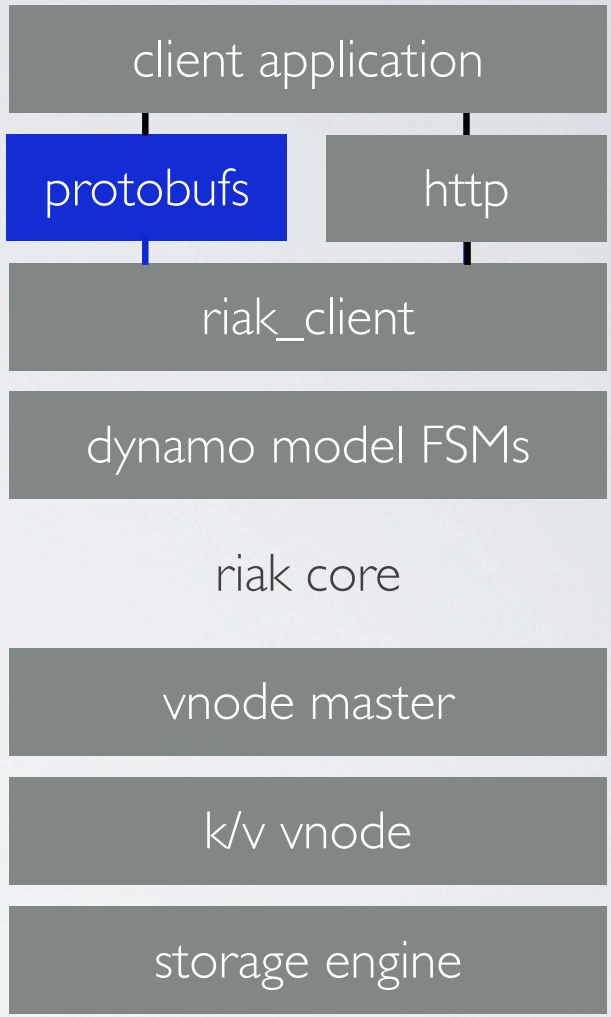
basho

riak

throughput matters!

protobufs are fast, simple, compact

- {packet,4}
- {active,once}
- socket owner handles both
  TCP packets and
  internal response messages



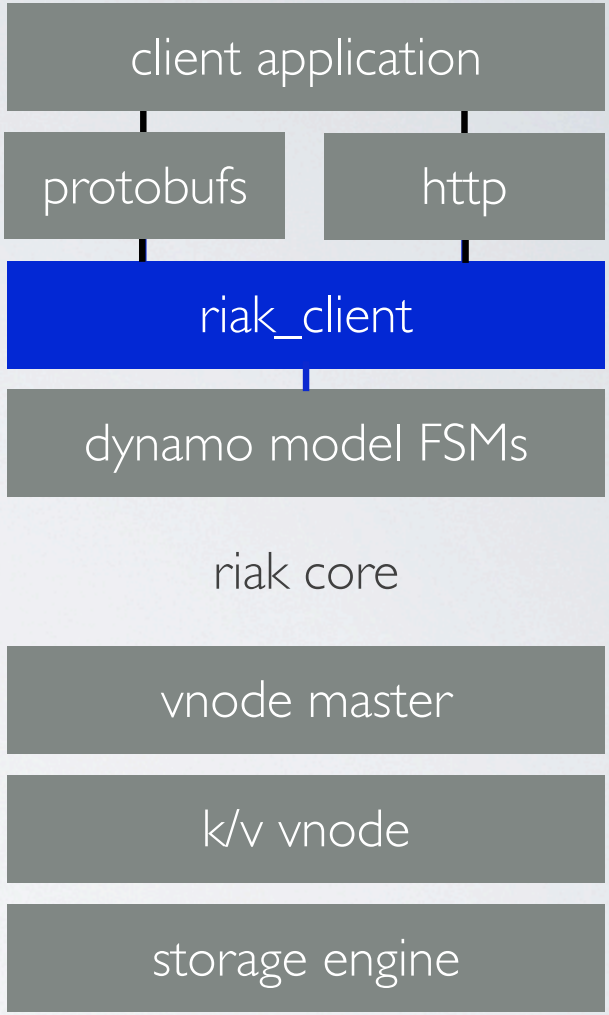
basho

riak

both the protobuf and HTTP entry points use the same interface

erlang native interface as general API

- parameterized module over
  - coordinator node
  - client instance id
- all access into Riak defined here



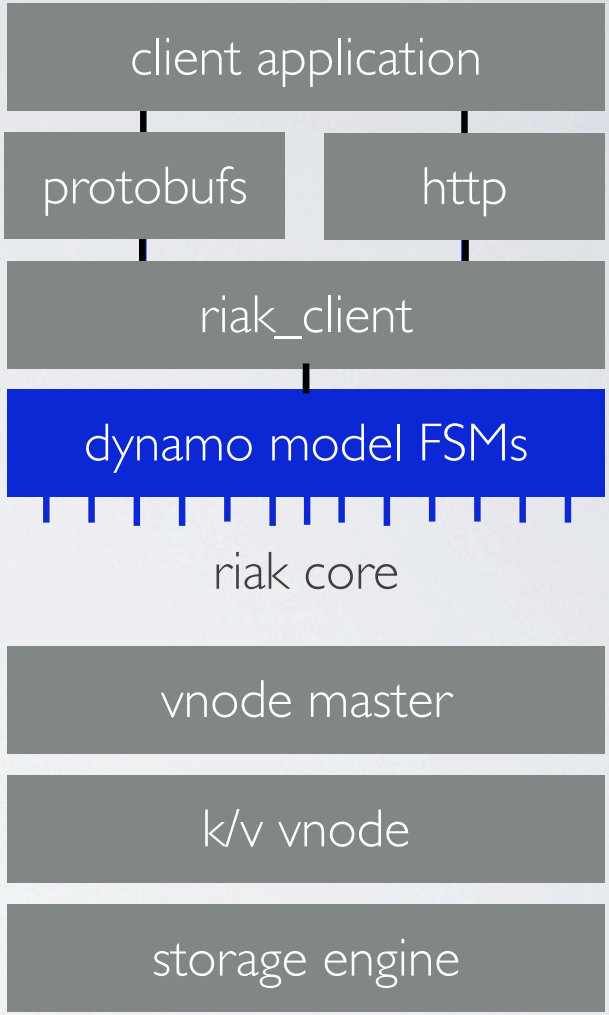
basho

riak

simple interface, complex semantics

direct inspiration: Amazon's Dynamo

- gen_fsm helps a lot here
- interactions with N other nodes
- multiple phases of interaction
- version vector resolution



basho

riak

digression: testing tricky FSMs is tricky

(dynamo model FSMs)

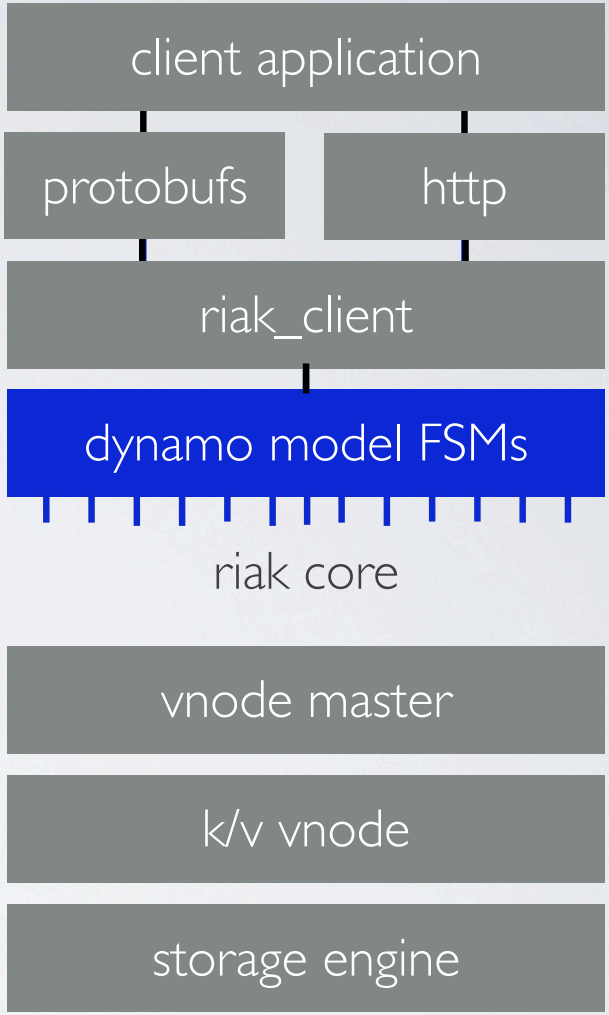
QuickCheck to the rescue!

- property-based / model-based tests
- bugs may only appear with unexpected combinations of events
- shrinking these combinations helps find minimal failure cases

simple interface, complex semantics

direct inspiration: Amazon's Dynamo

- gen_fsm helps a lot here
- interactions with N other nodes
- multiple phases of interaction
- version vector resolution



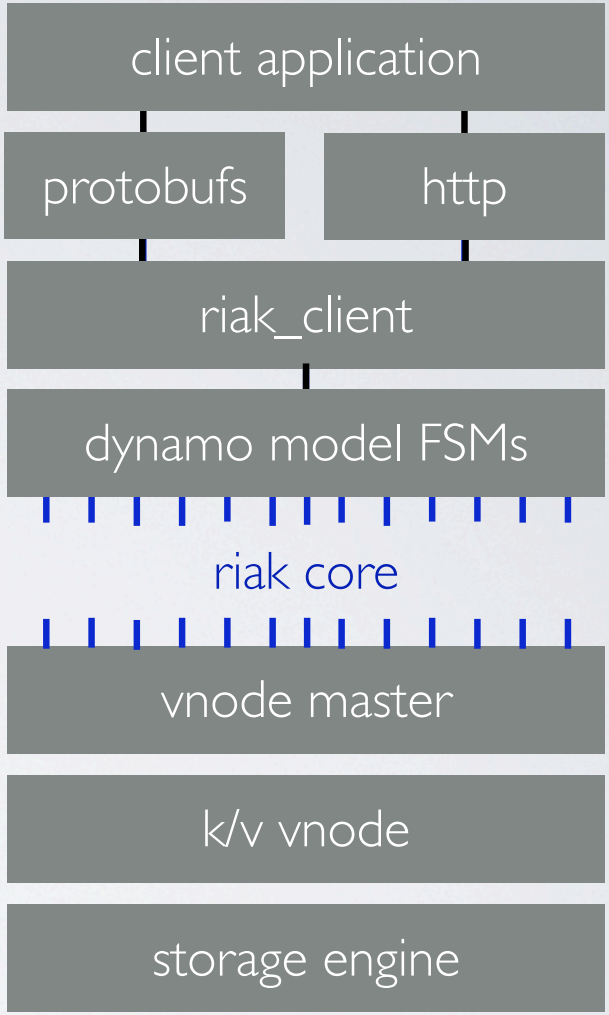
basho

riak

FSMs run anywhere, use everything

Riak Core: fundamental distribution

arbitrary number of storage nodes
each contributing to the whole



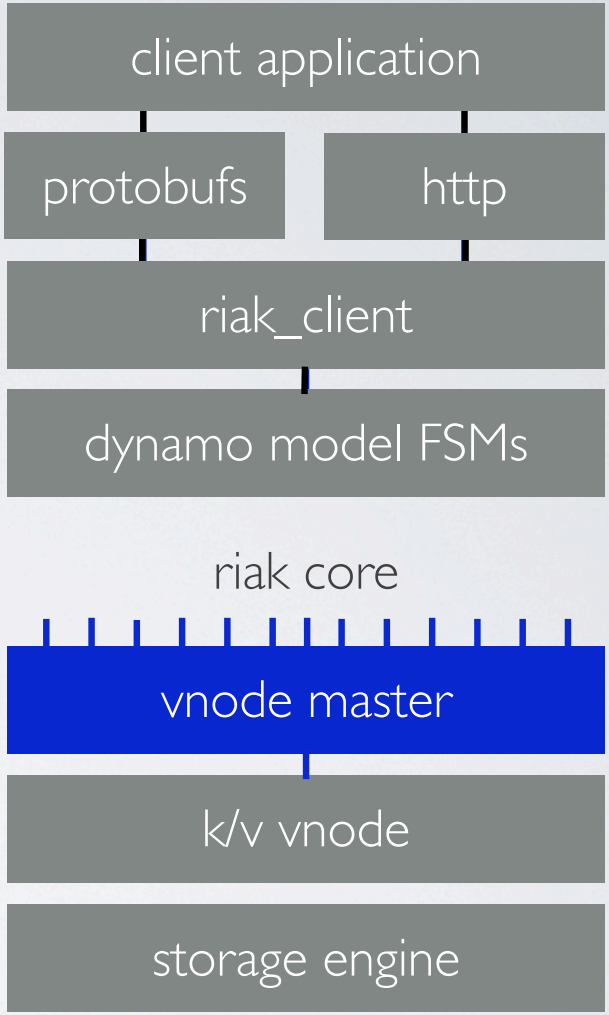
basho

riak

gen_server as a multiplexer
and well-known entry point

one host, many virtual nodes

instantiate vnodes as needed



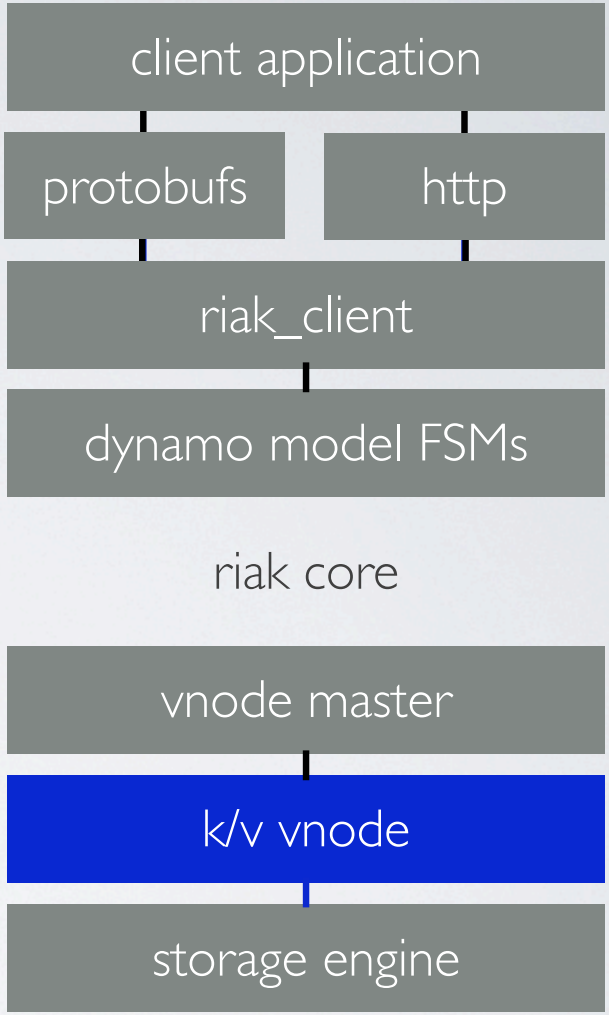
basho

riak

disposable, per-partition actor
for access to local data

enable parallelism & fault-tolerance
the Erlang way (per-process)

node-local k/v storage abstraction



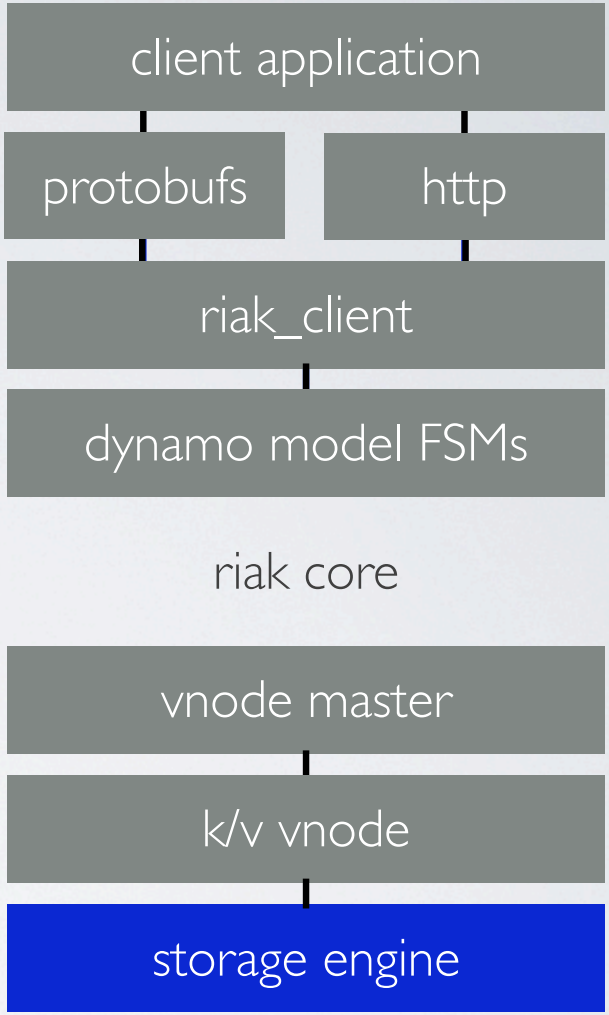
basho

riak

like an Erlang "behaviour"
separating development of disk
or other storage from distribution

steps forward w/o breaking code
(innostore, bitcask, ...)

all storage systems look the same

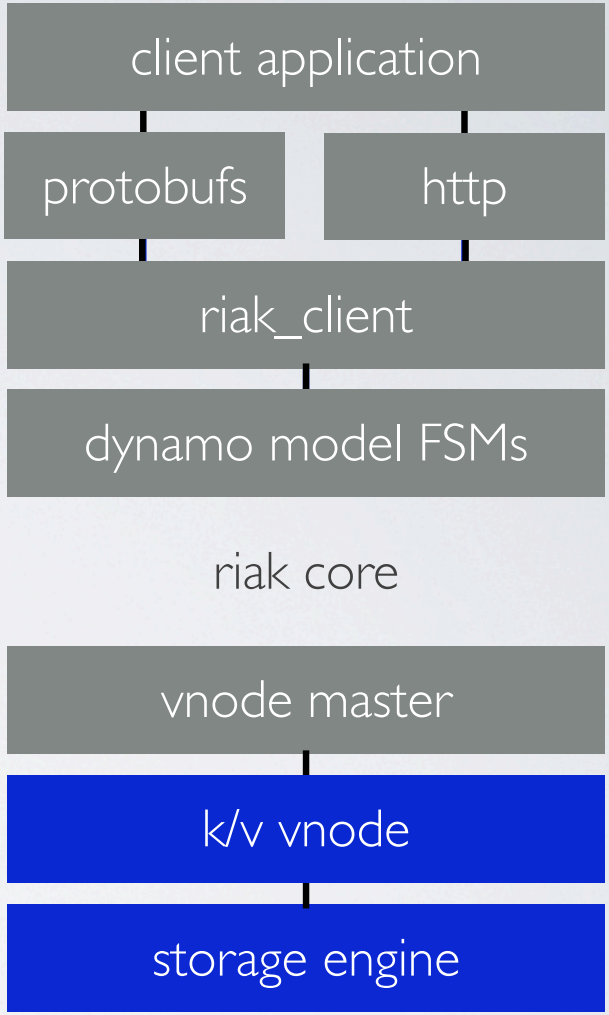


basho

riak

it's just a key/value store

from the bottom

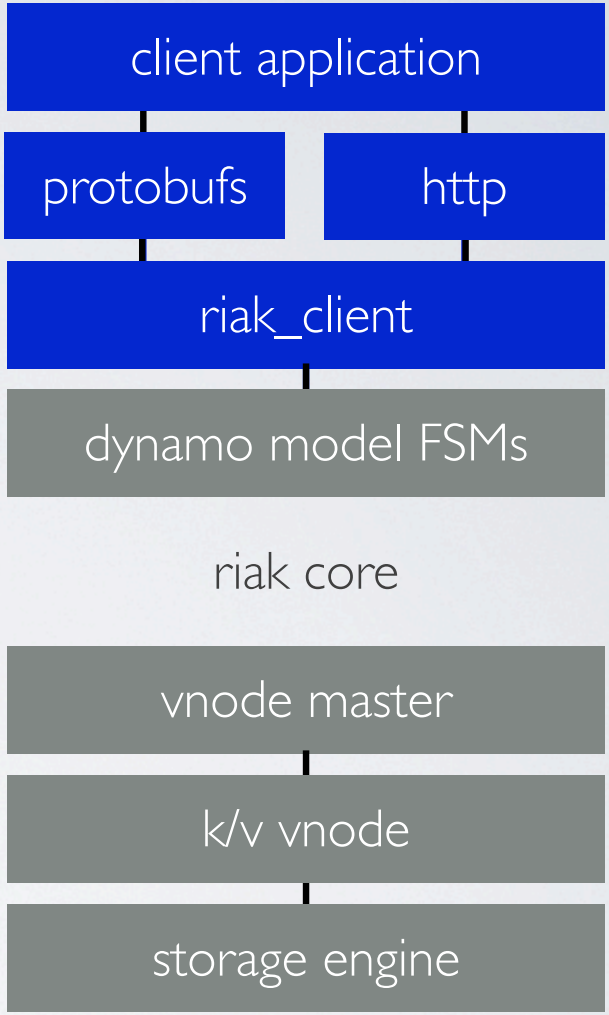


basho

riak

from the top

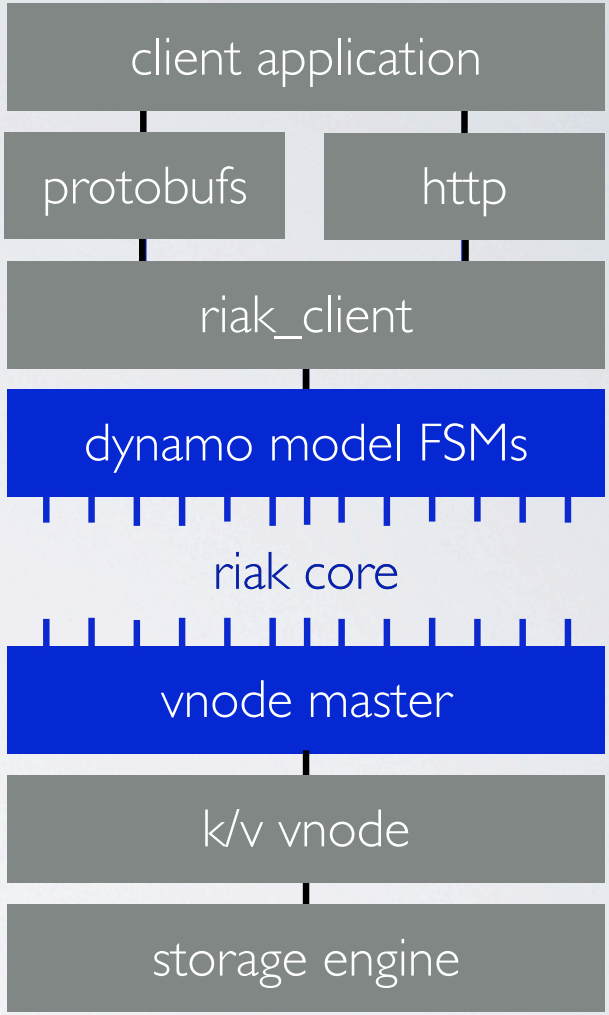
it's just a key/value store



basho

riak

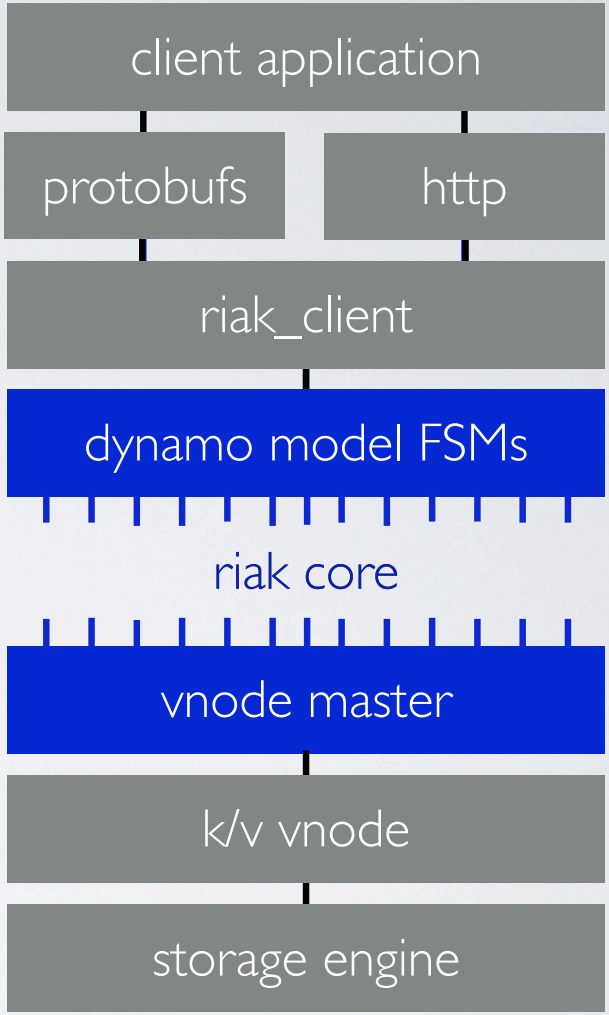
it's a distributed system at heart



basho

riak

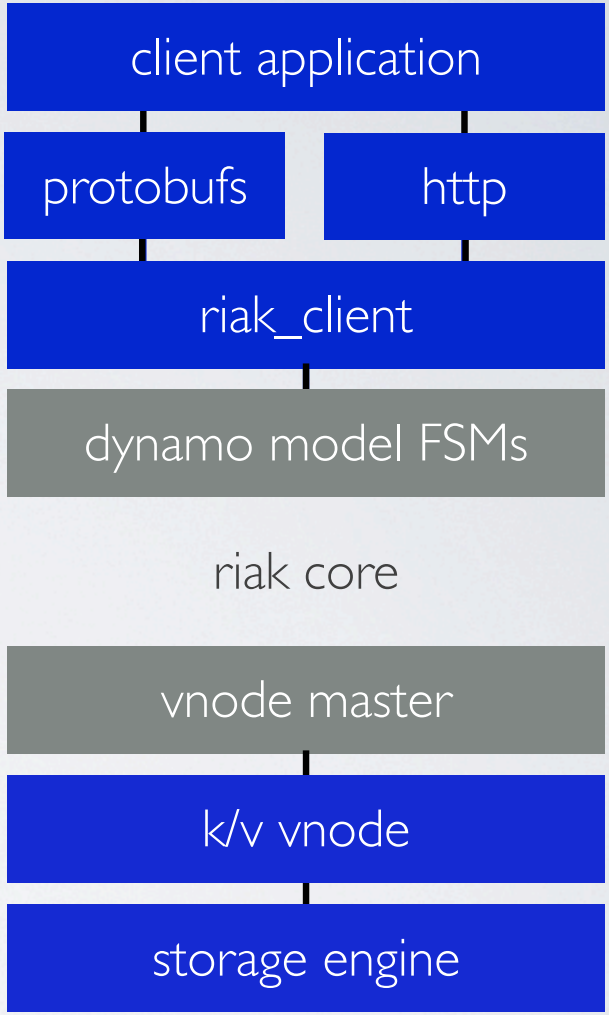
carefully managed complexity...



basho

riak

allows simplicity at the edges



basho

riak

http://www.basho.com

follow twitter.com/basho/team

riak-users@lists.basho.com

#riak on Freenode