Riak in the Cloud

Tom Santero
Technical Evangelist
@tsantero

GOTO Copenhagen
May 21, 2012

Distributed Systems

Distributed Systems

Are Difficult

Complexity

- predictable
- unpredictable / chaotic

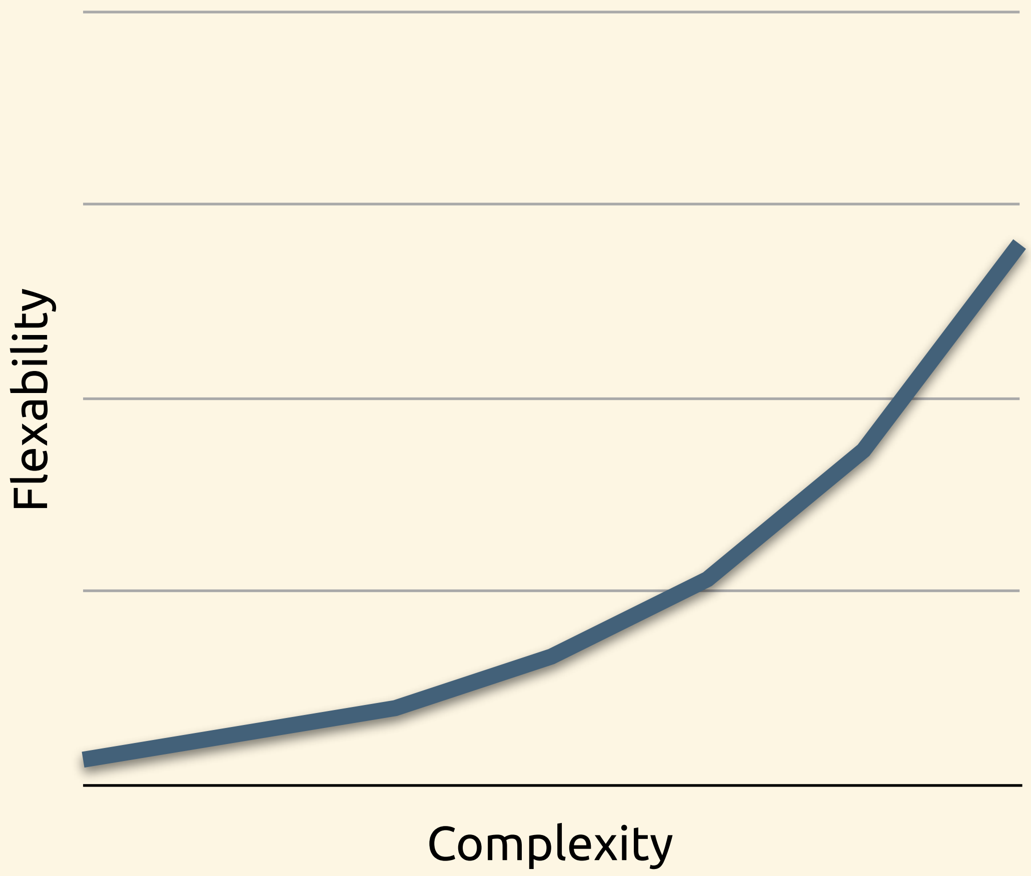
Failures

- deterministic, predicable failures
- unpredictable failures

Issues in Scaling

- limit to vertical scaling
- solution: horizontal scaling
  - increased network chatter

Tradeoff



riak

Riak in a Nutshell

- key/value datastore
- distributed
- highly available
- written in Erlang
- inspired by Amazon’s Dynamo

The Dynamo Bits

- decentralized - no master
- homogenous - all nodes participate equally
- consistent hashing
- data replication
- horizontally scalable
    - more nodes = more x

riak_core

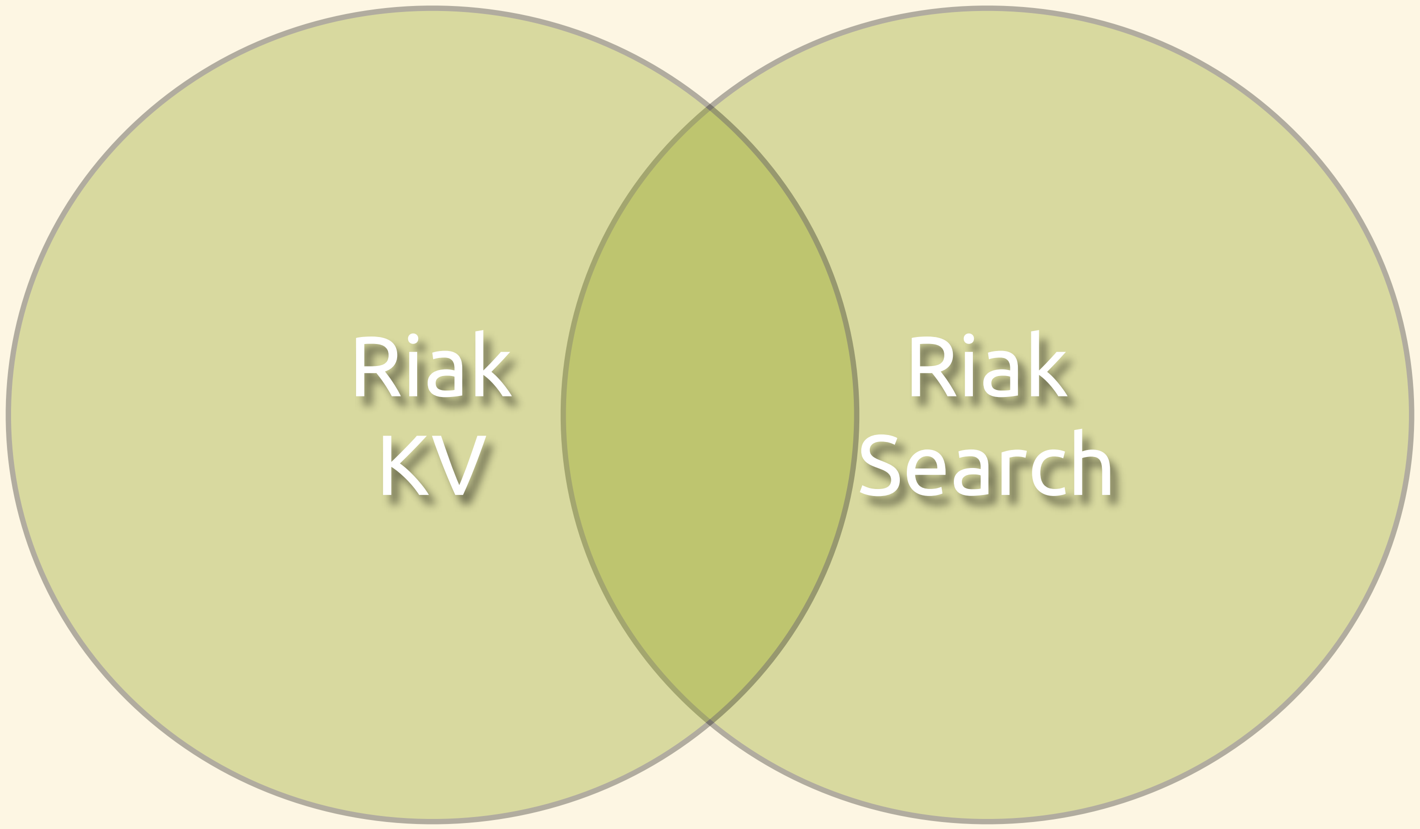
Riak Core

- open-source Erlang/OTP library
- node liveness and cluster membership
- enables partitioning and distribution
- stores cluster state
- foundation for distributed applications

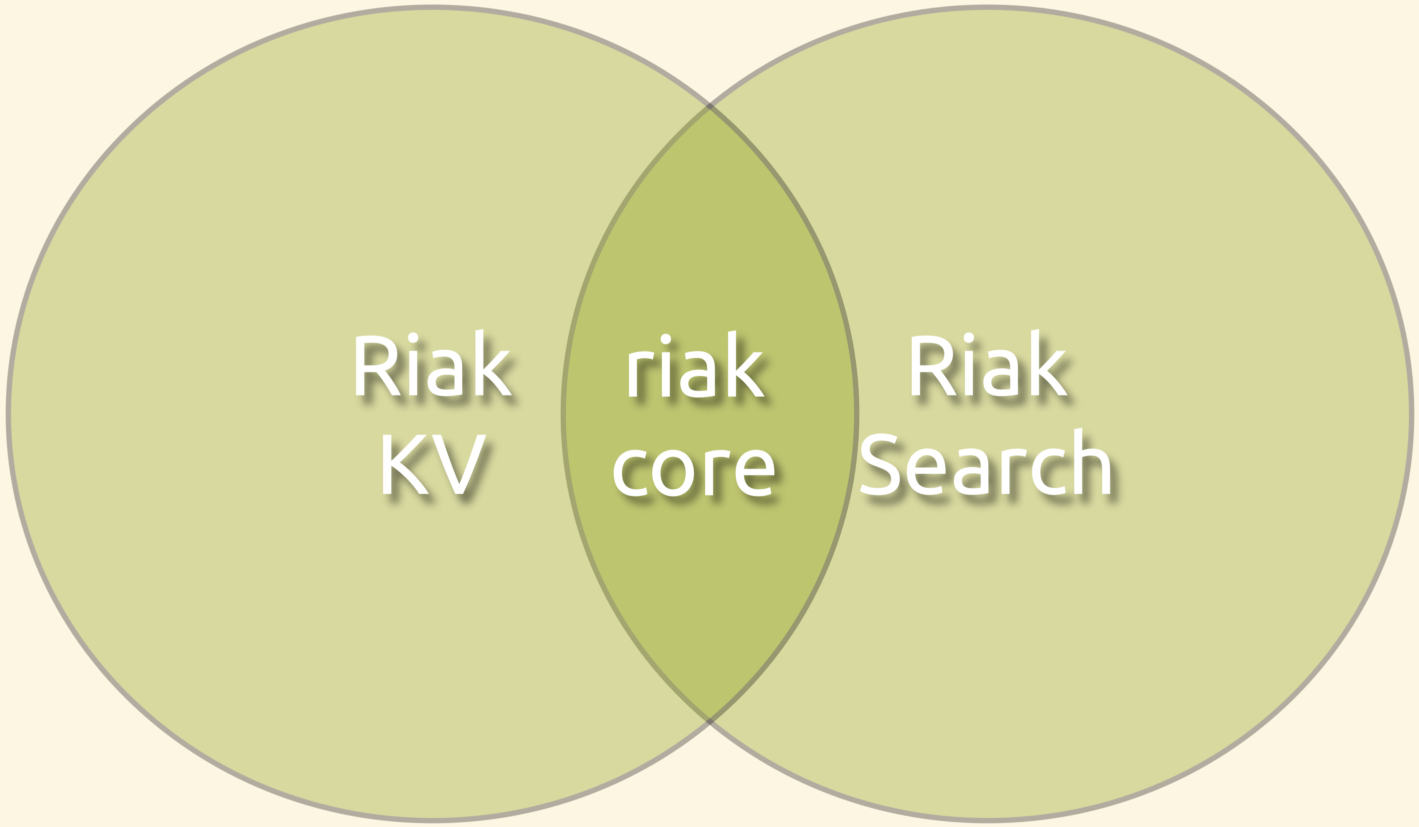
Origins

Origins

Origins



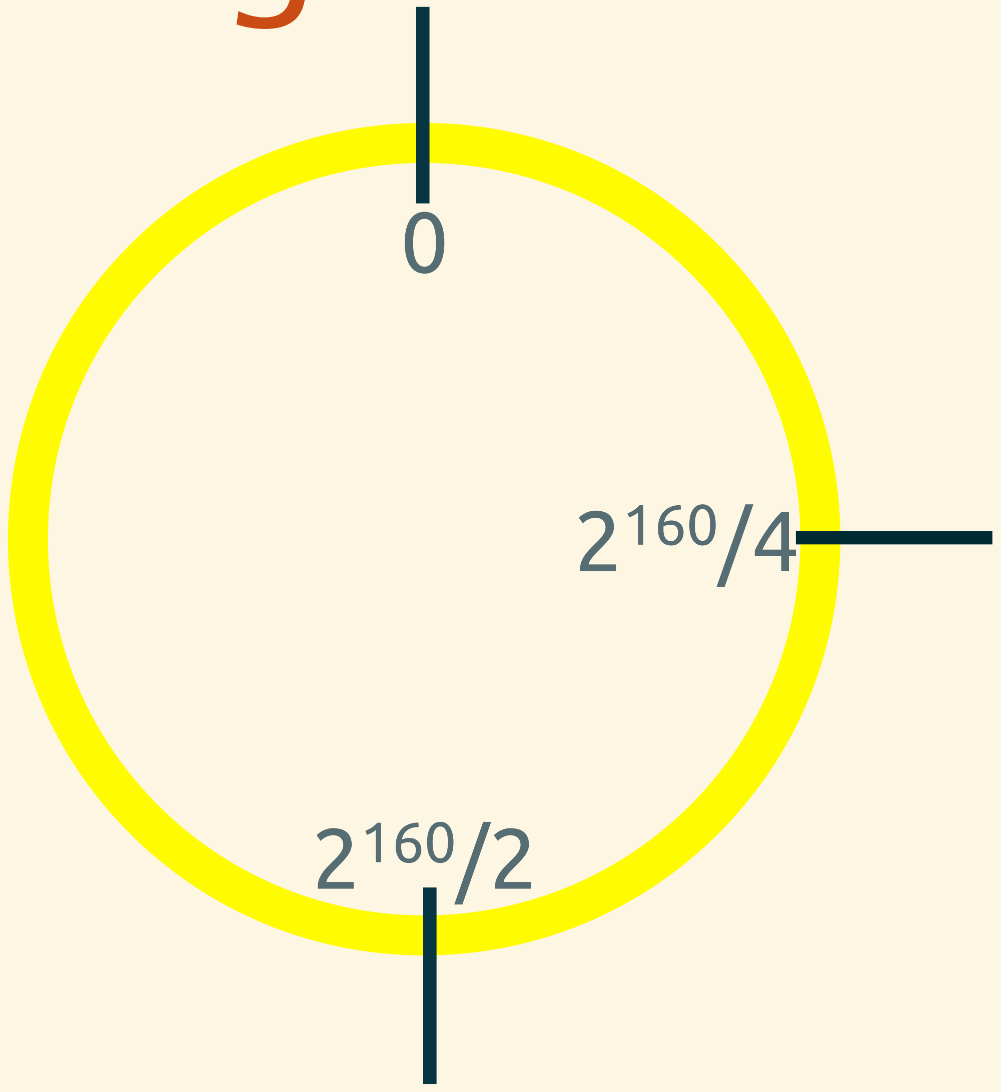
Origins



Consistent Hashing & The Ring

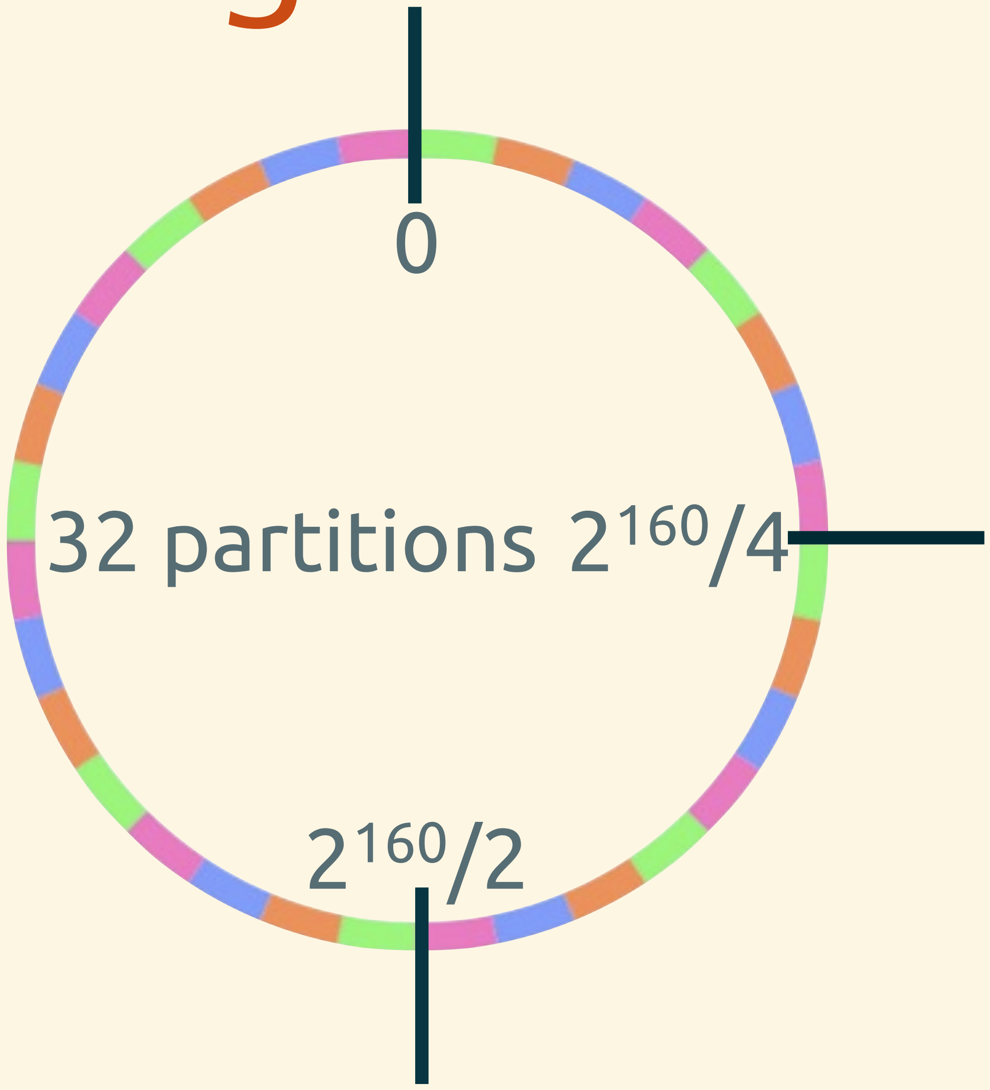
Consistent Hashing & The Ring

- 160-bit integer keyspace



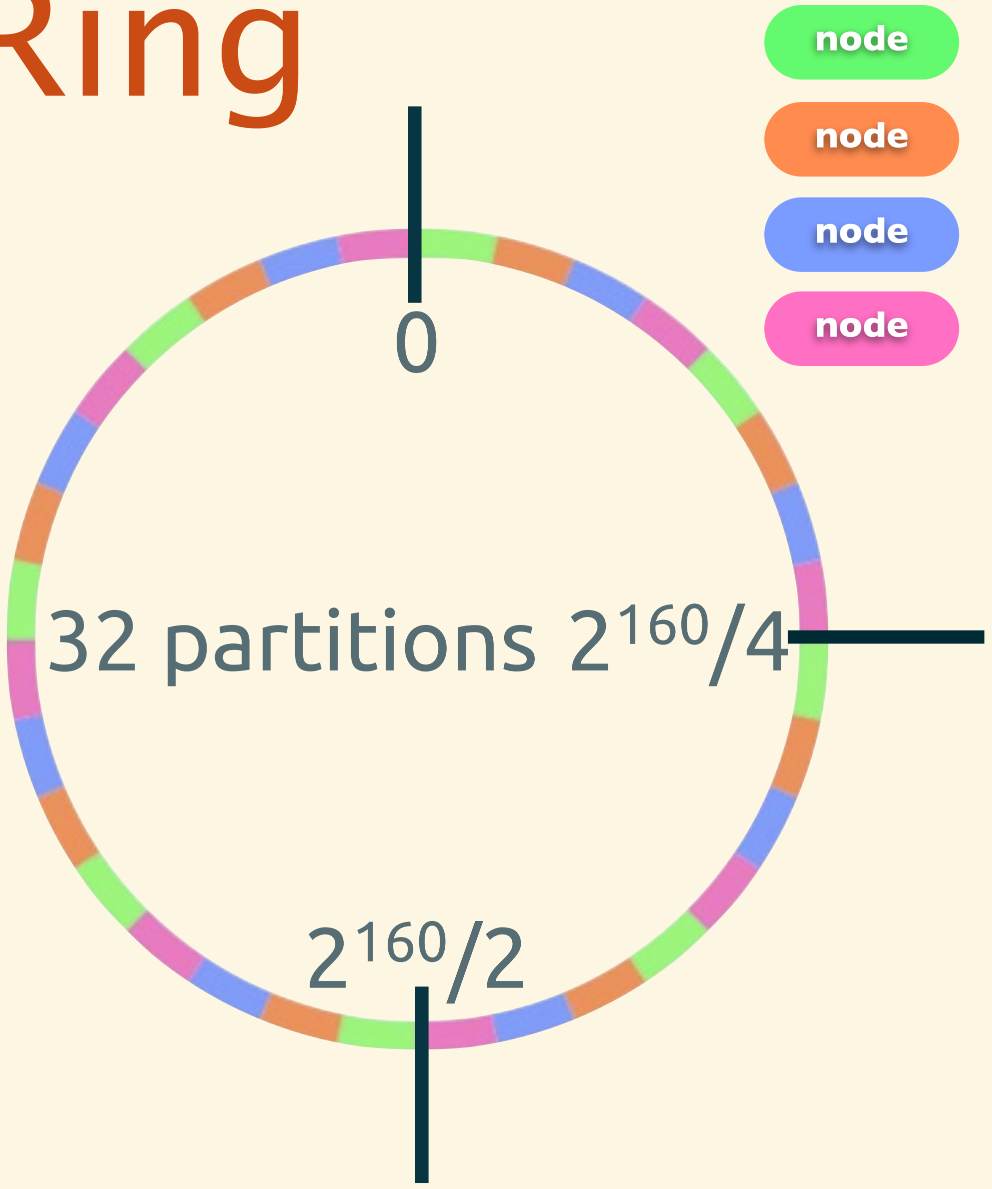
Consistent Hashing & The Ring

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions



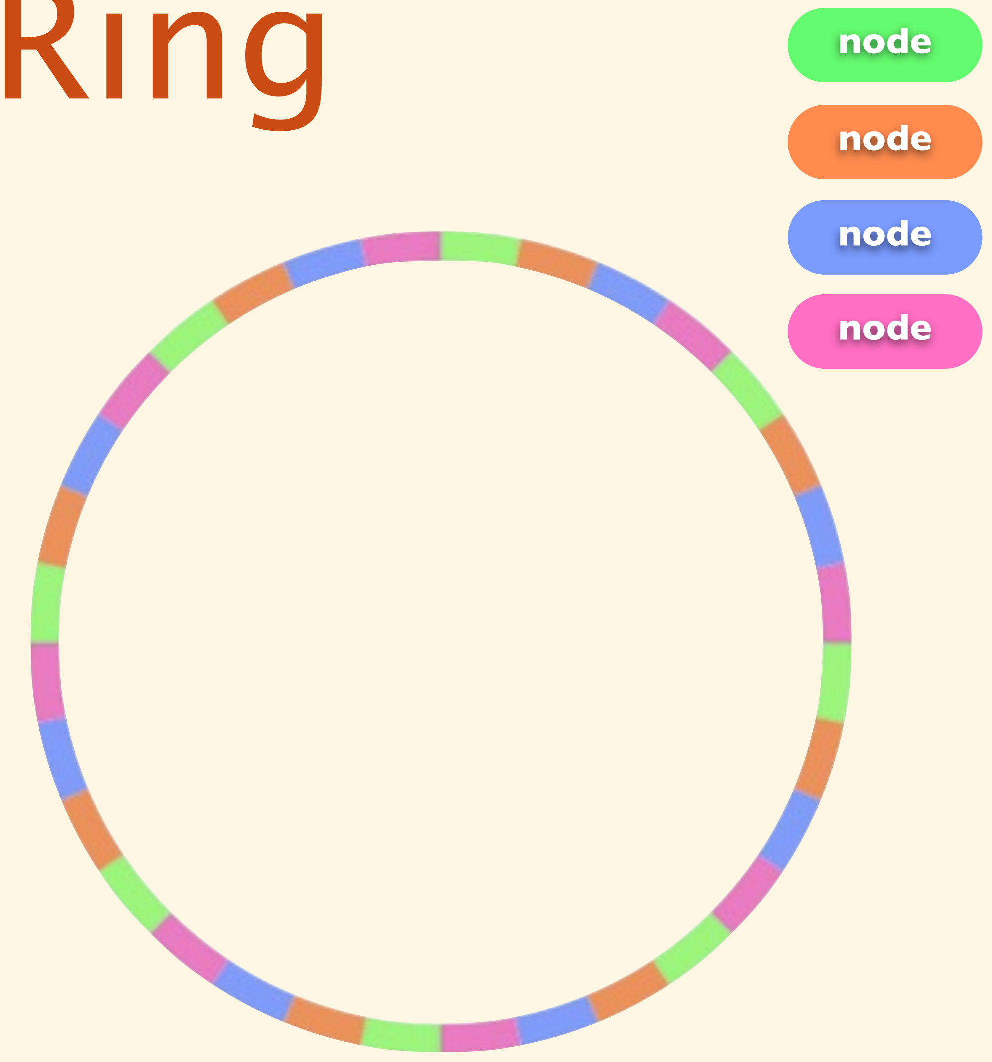
Consistent Hashing & The Ring

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster



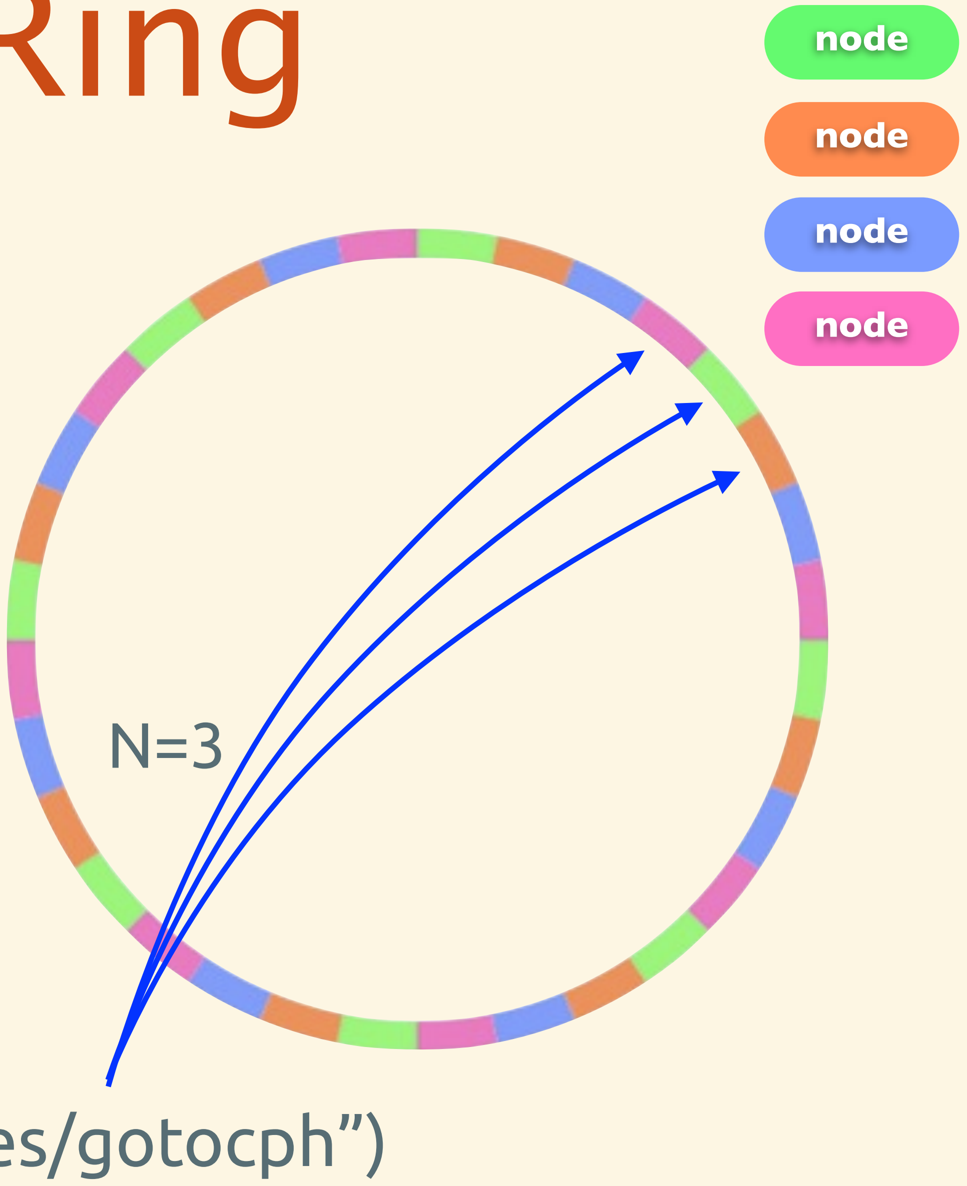
Consistent Hashing & The Ring

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



Consistent Hashing & The Ring

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



Node Liveness & Membership

- **riak_core_node_watcher**
  - tracks status of nodes in cluster
  - API advertising and locating nodes
- **riak_core_node_watcher_events**
  - generates events based on activity (joining, leaving cluster...etc)

Partitioning & Distribution

- master/worker configuration
- riak_core processes are vnodes
- **riak_core_vnode_master**: coordinator
  - starts worker vnodes + routes requests
- **riak_core_vnode**: workers

Cluster State

- **riak_core_ring**
  - create and change ring state data
- **riak_core_ring_manager**
  - manages cluster data for node
  - main entry point for applications
- **riak_core_gossip**
  - ensures ring is consistent

NoSQL Complexity

- tunable CAP
- quorum controls: R, W, DW, PW, PR
- numerous backend options
- unfamiliar query model
- immature client libraries (sometimes)
- and more....

Eliminate Complexity

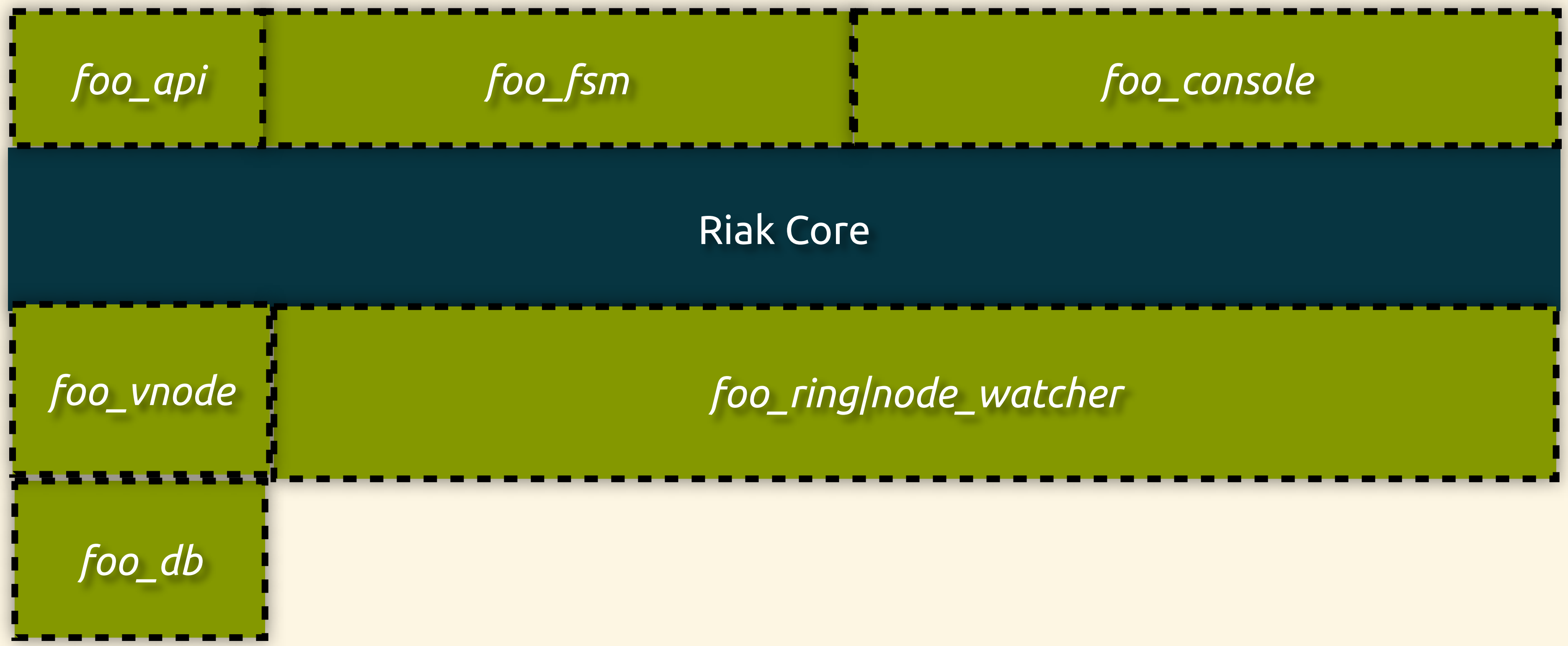
Vertical Services

- abstract away NoSQL complexity
- provide simpler API
- fewer configuration knobs
- existing client libraries

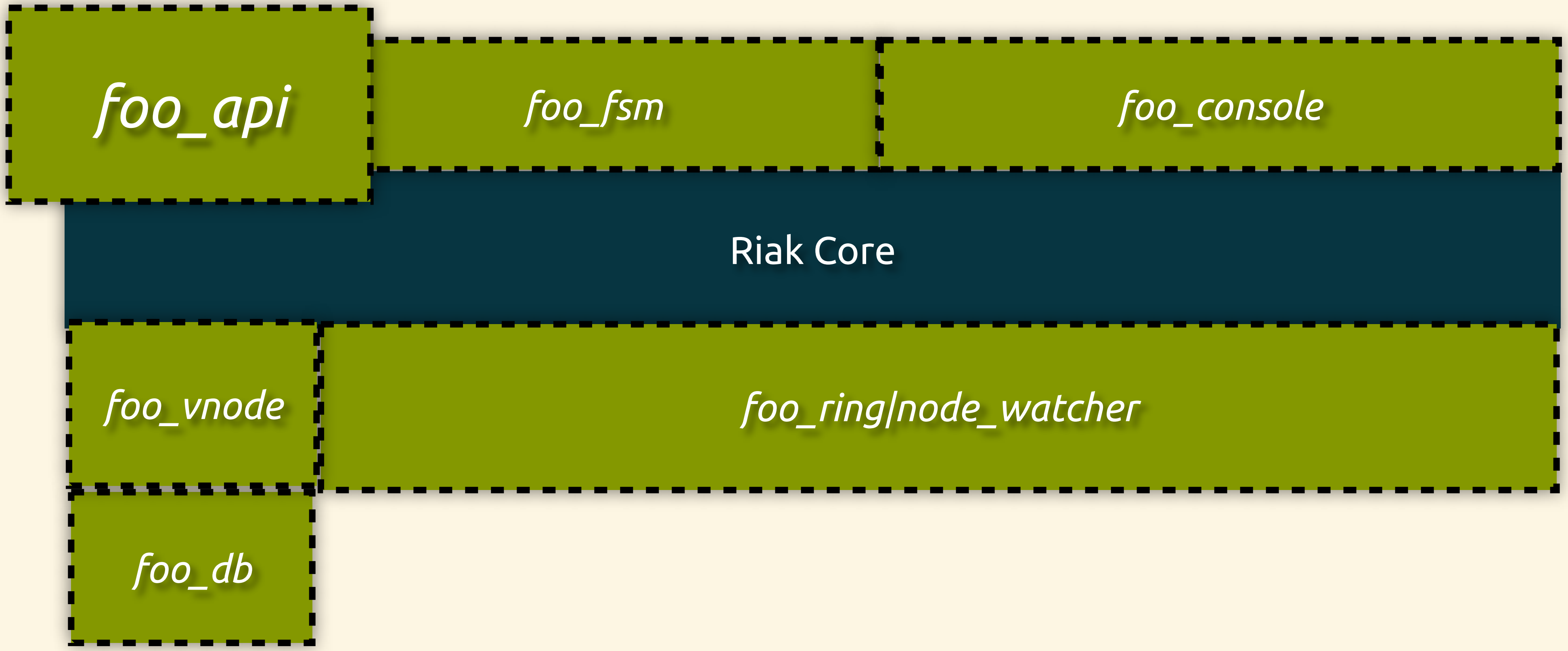
Riak Service Pattern

- Riak Core
- Stateless Proxy
- Clustered Proxy

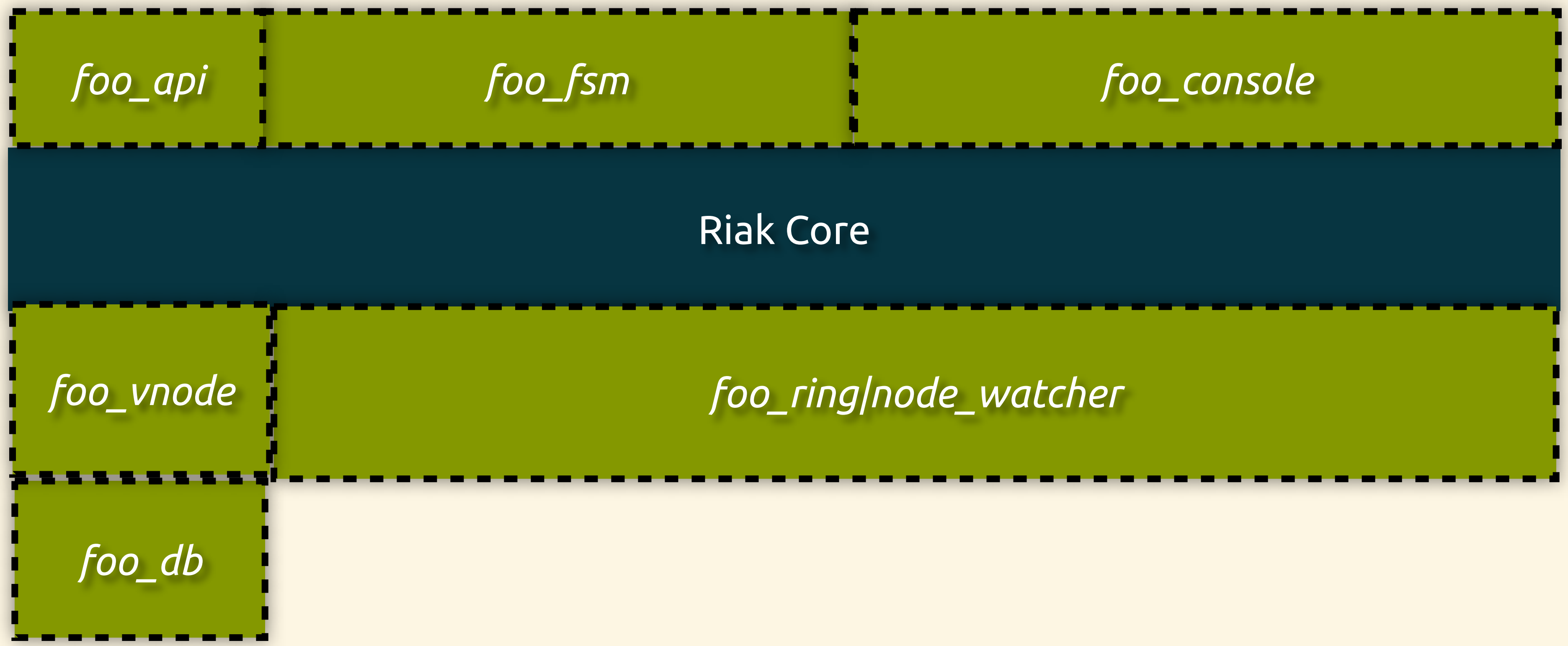
Riak Extension Points



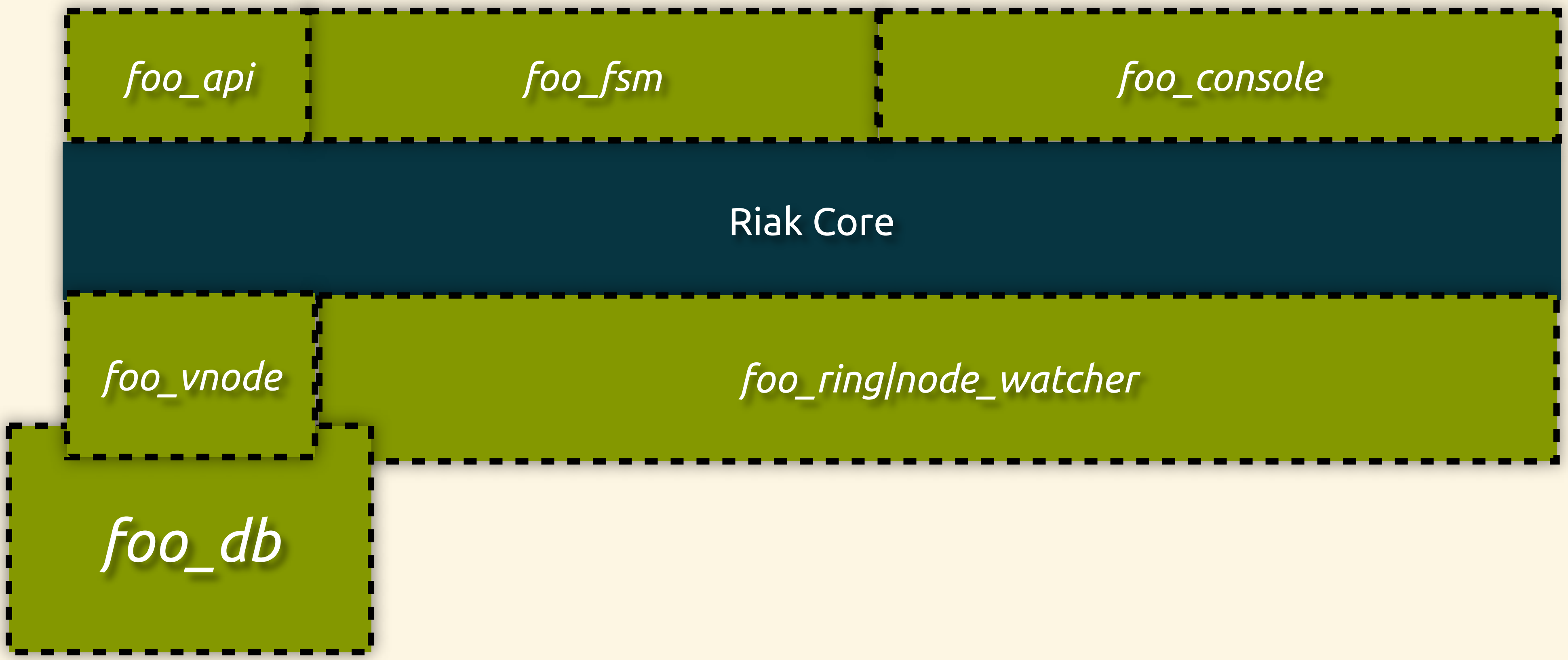
Riak Extension Points



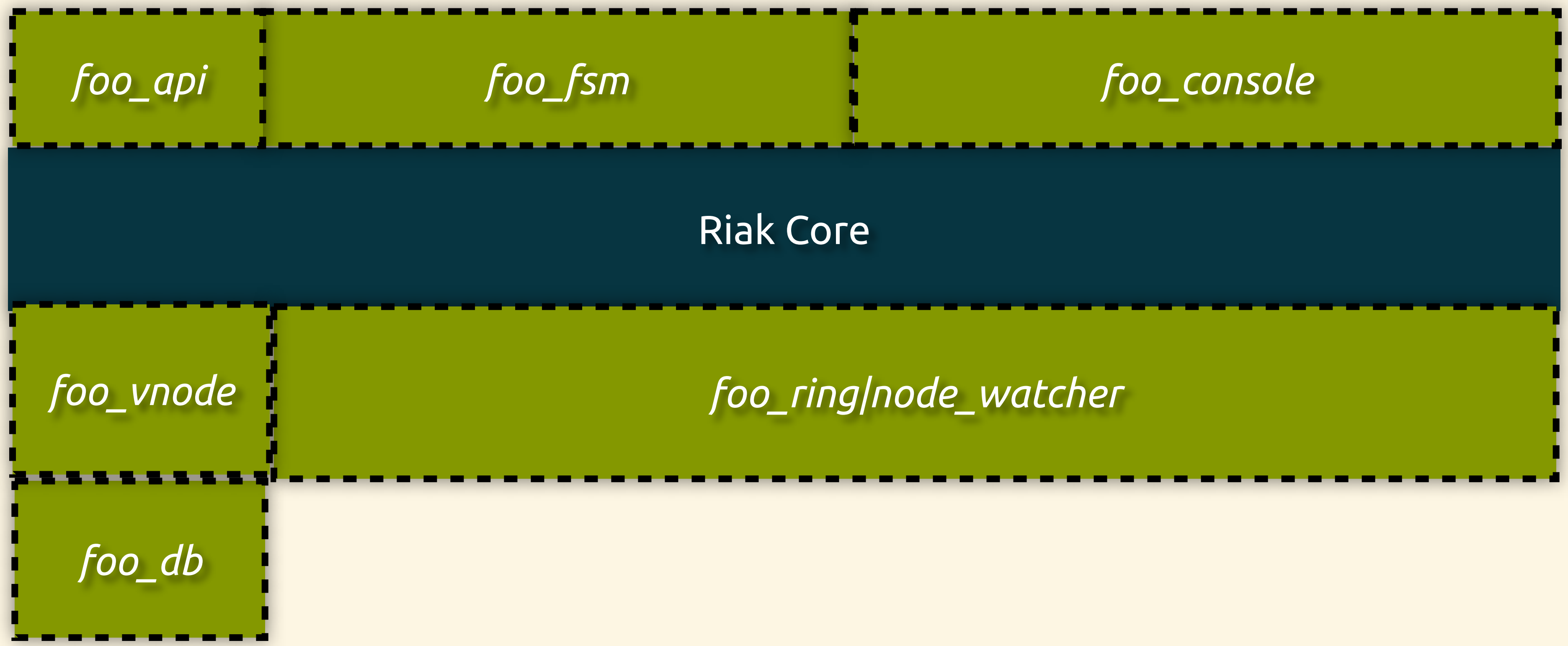
Riak Extension Points



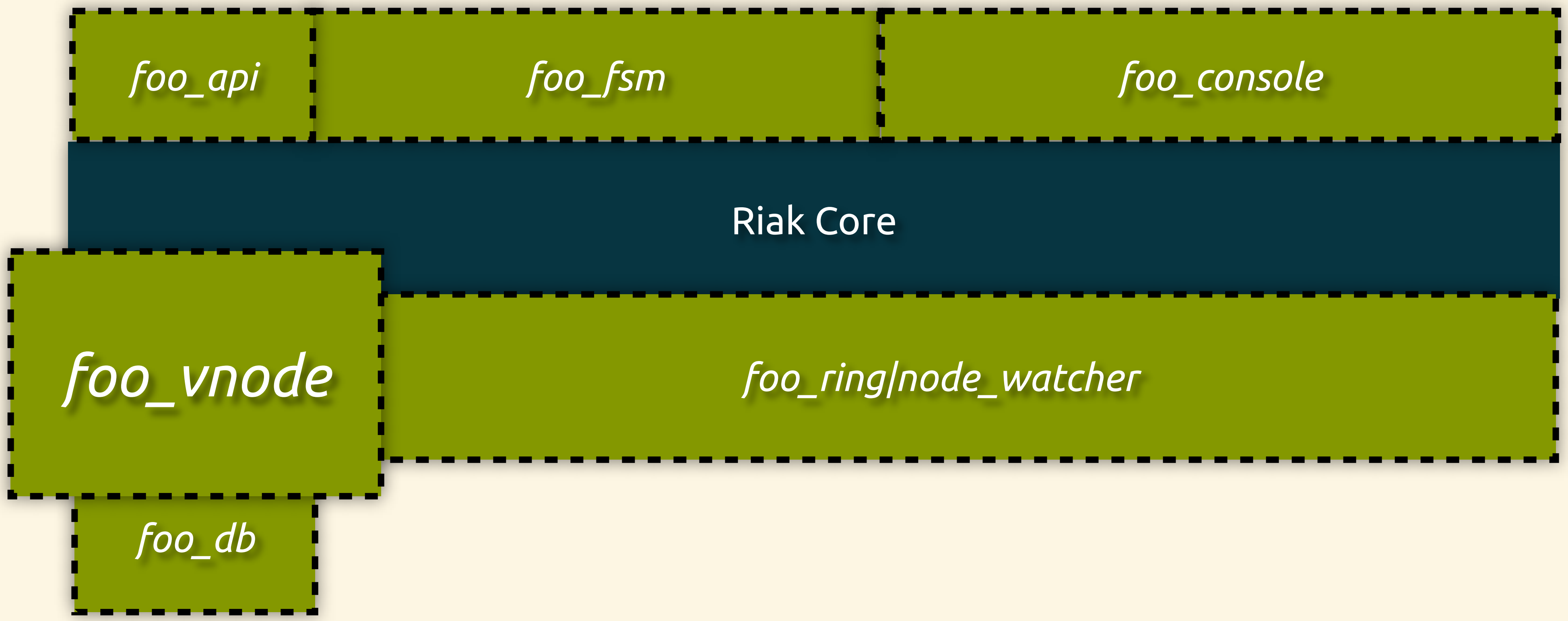
Riak Extension Points



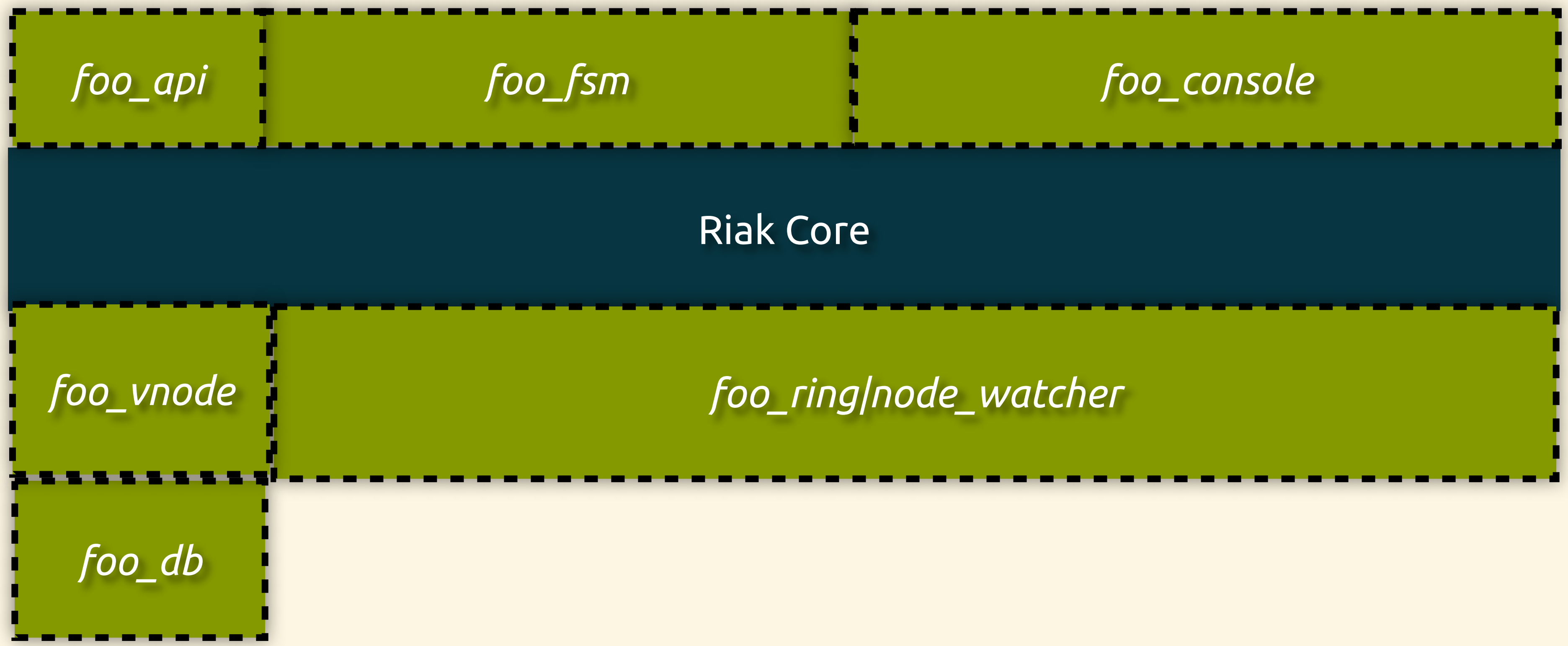
Riak Extension Points



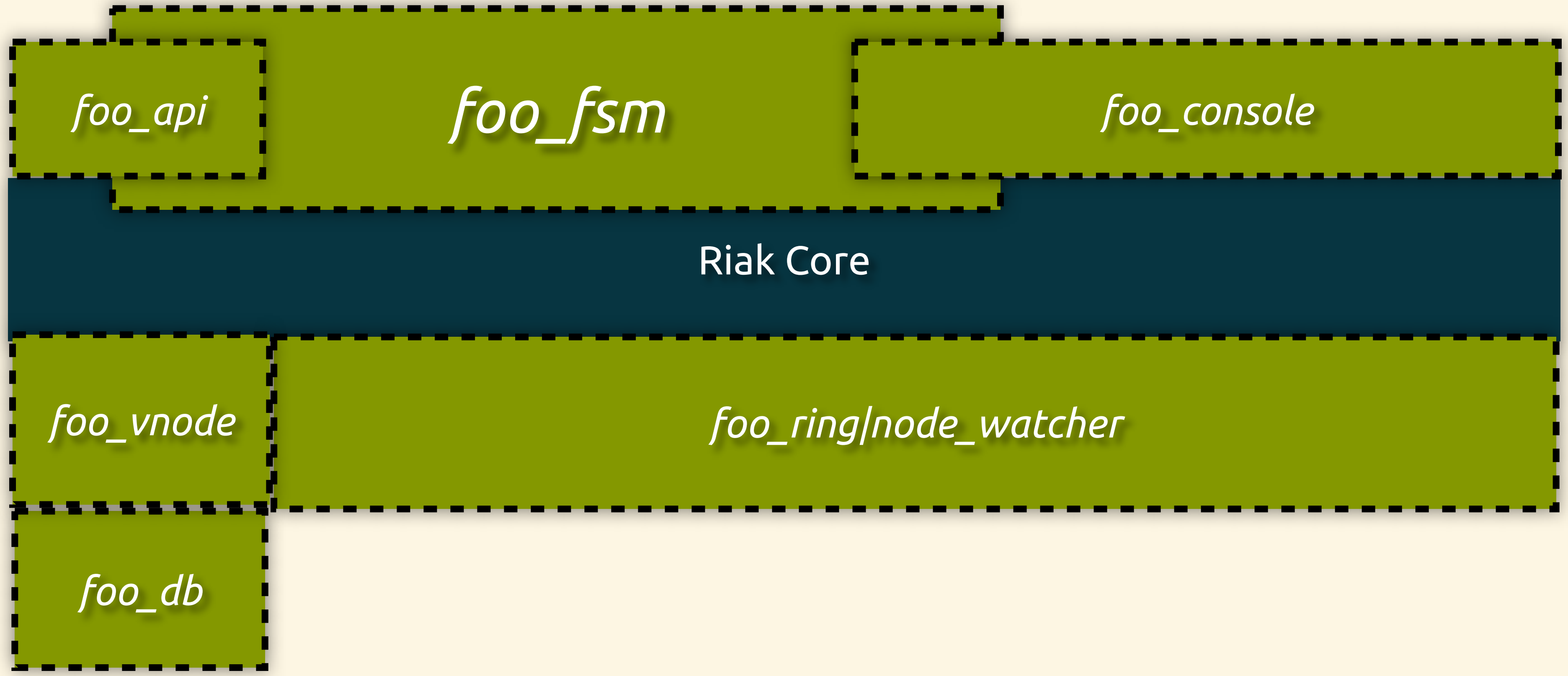
Riak Extension Points



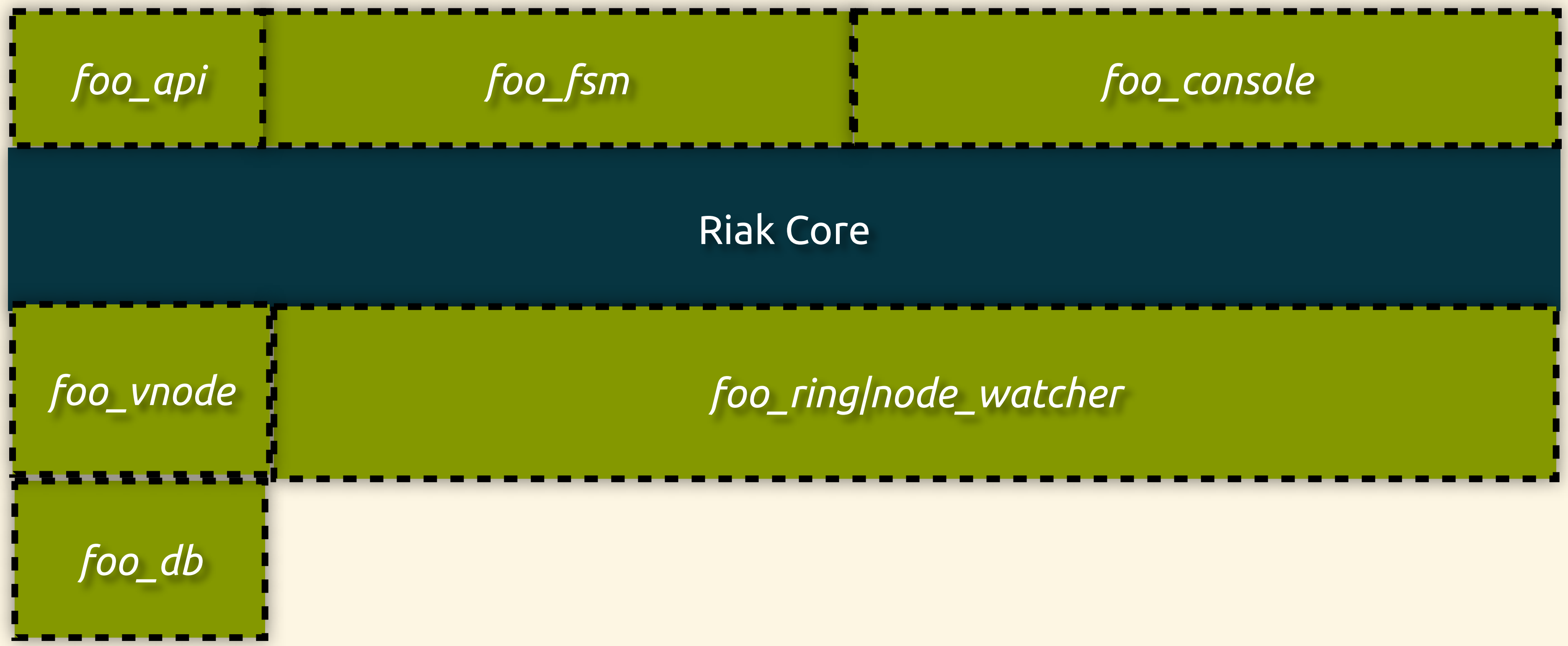
Riak Extension Points



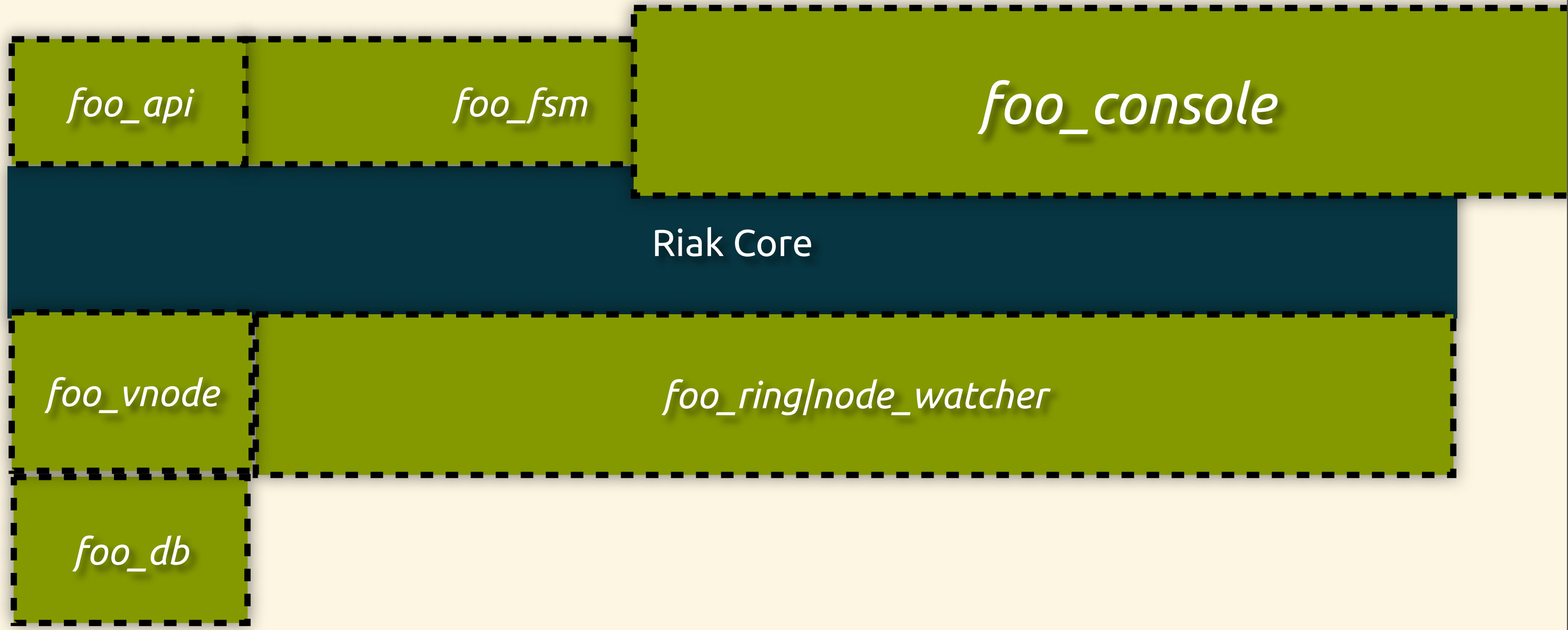
Riak Extension Points



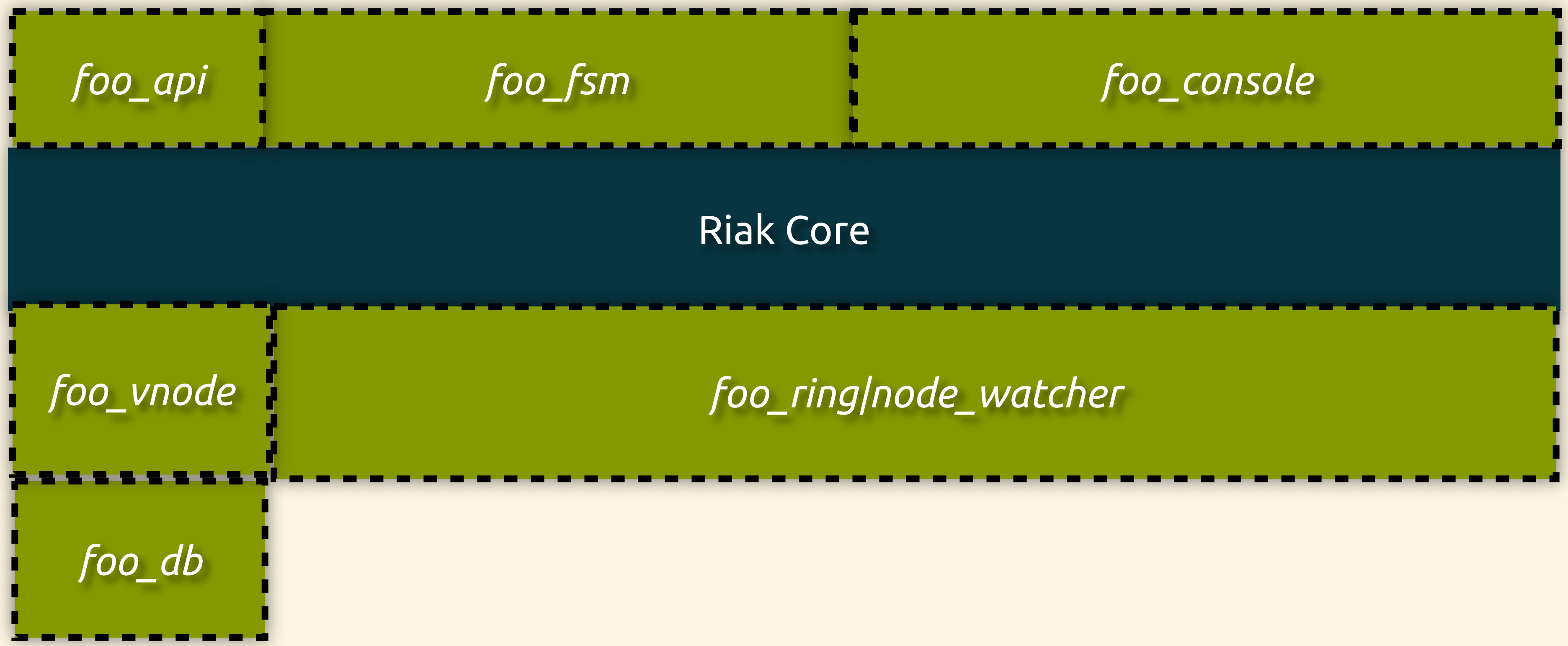
Riak Extension Points



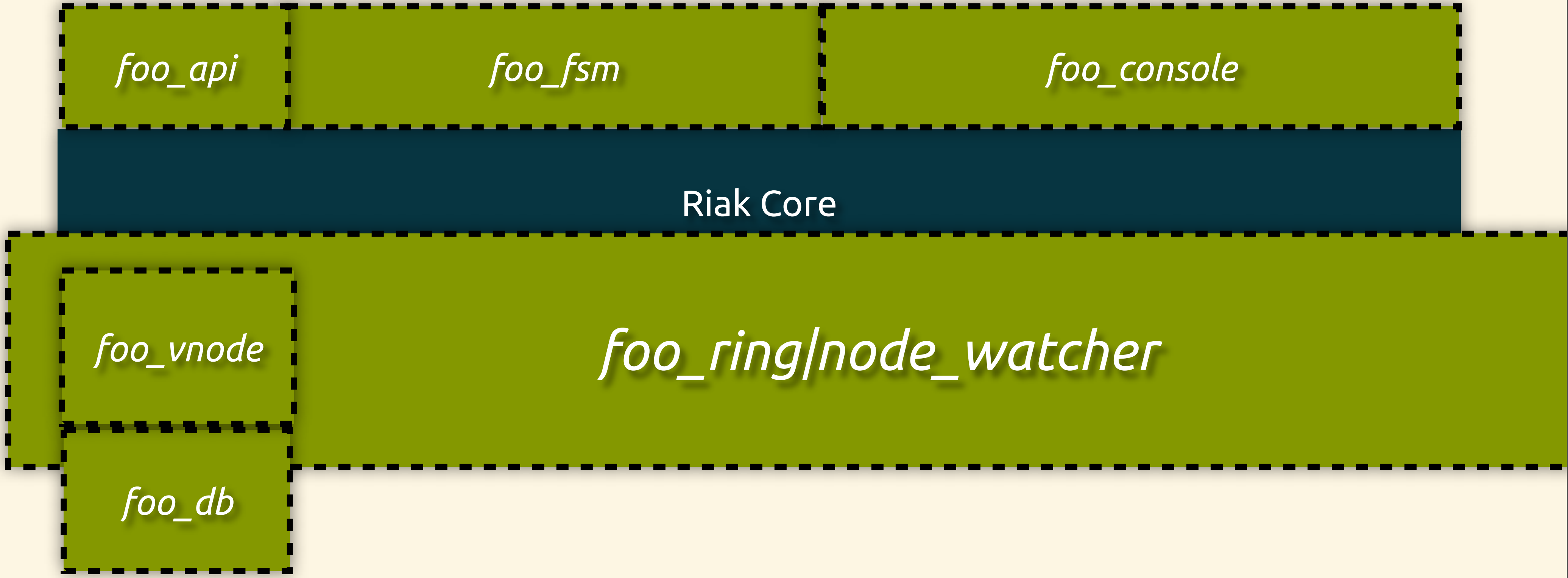
Riak Extension Points



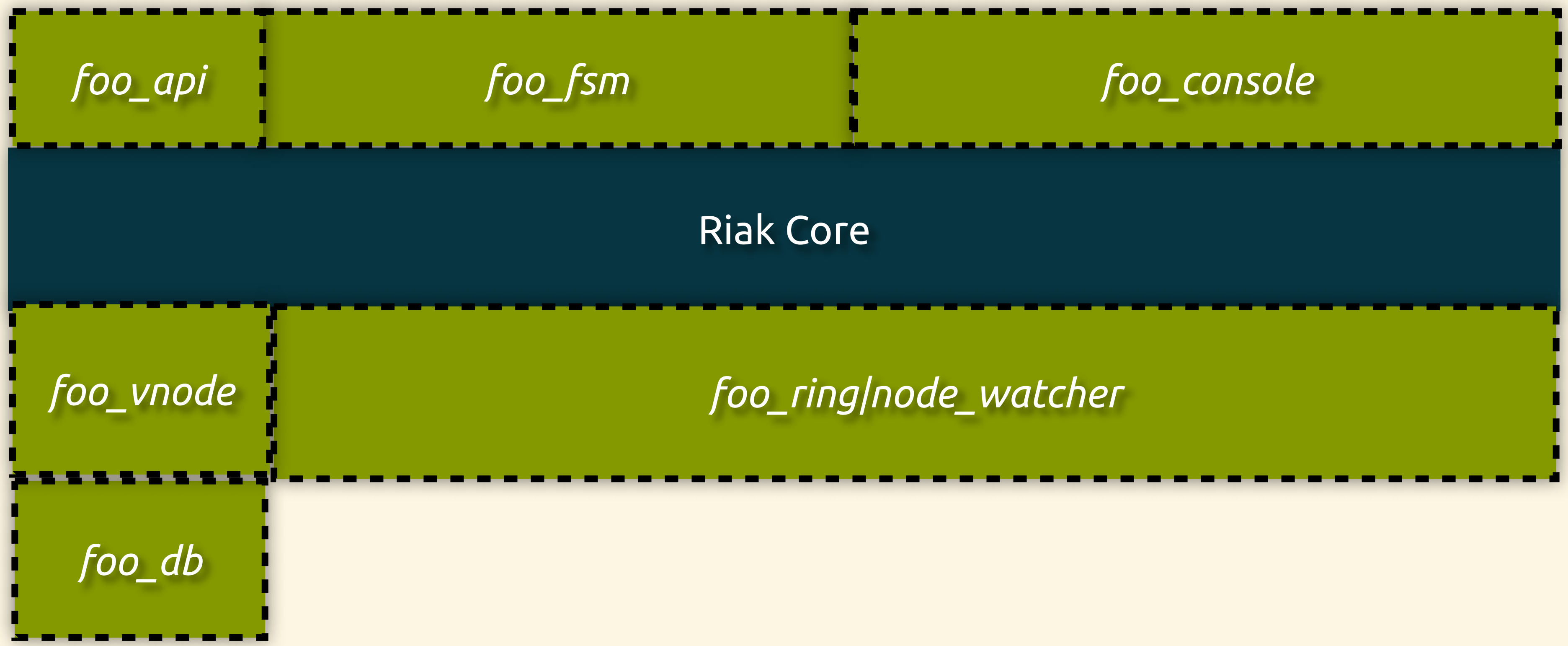
Riak Extension Points



Riak Extension Points



Riak Extension Points



Riak Core Apps

- custom vnode implementation
- client API
- FSM for coordinating with vnodes

Stateless Proxy

- implemented as client of Riak KV
- depolyed in a separate VM
- stores state in Riak KV
- proxies have no knowledge of each other
- scale independently of Riak

Clustered Proxy

- use Riak Core at proxy layer:
  - clustering
  - load balancing
  - distribution of proxy state

Cloud Services

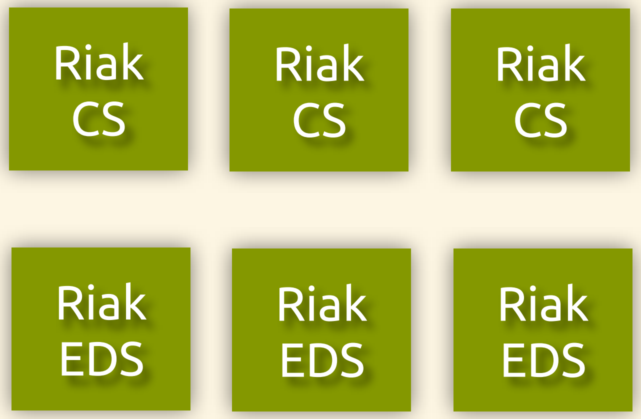
- globally distributed
- highly available
- horizontally scalable
- operationally simple

Example

riak cs

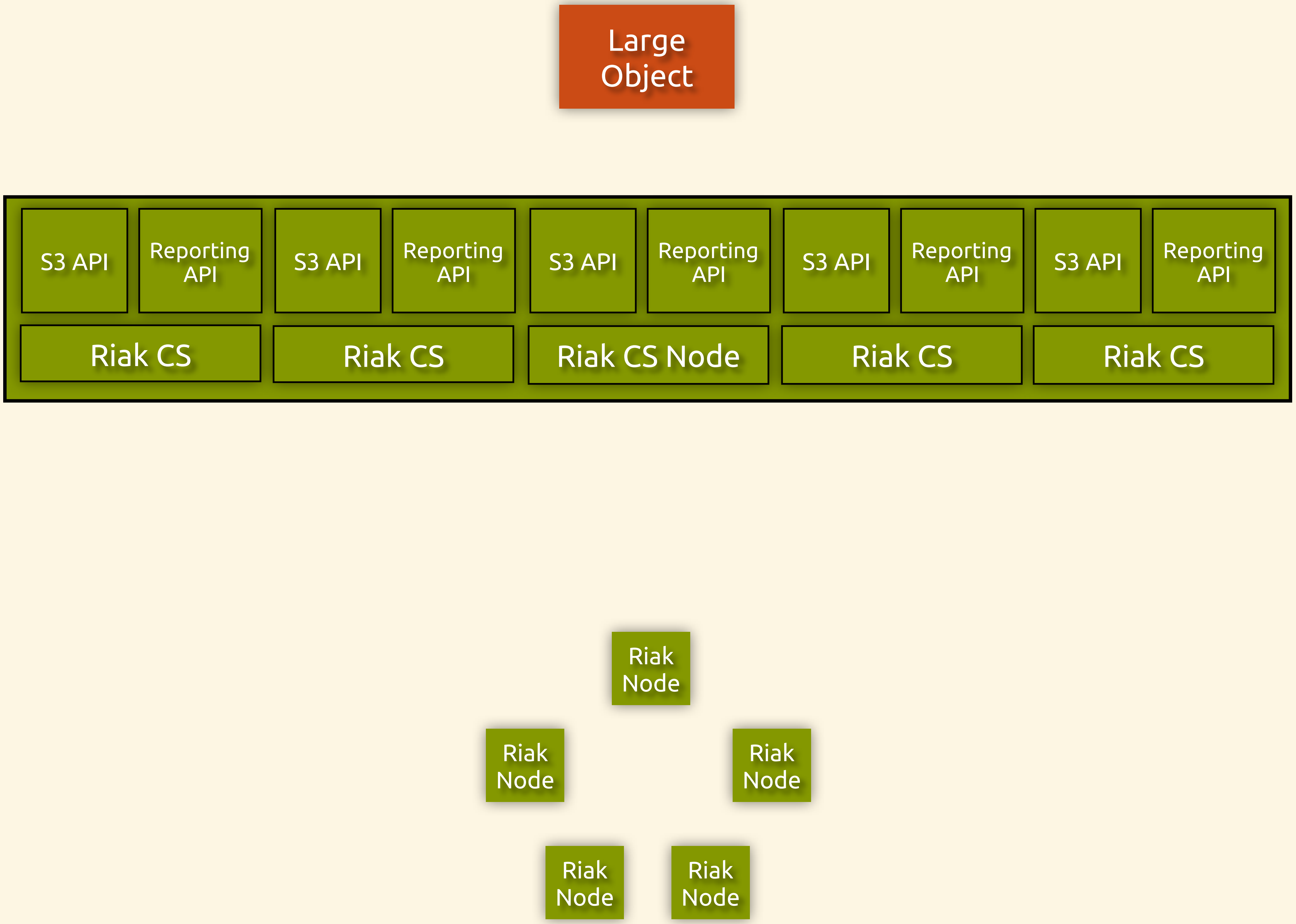
Riak CS

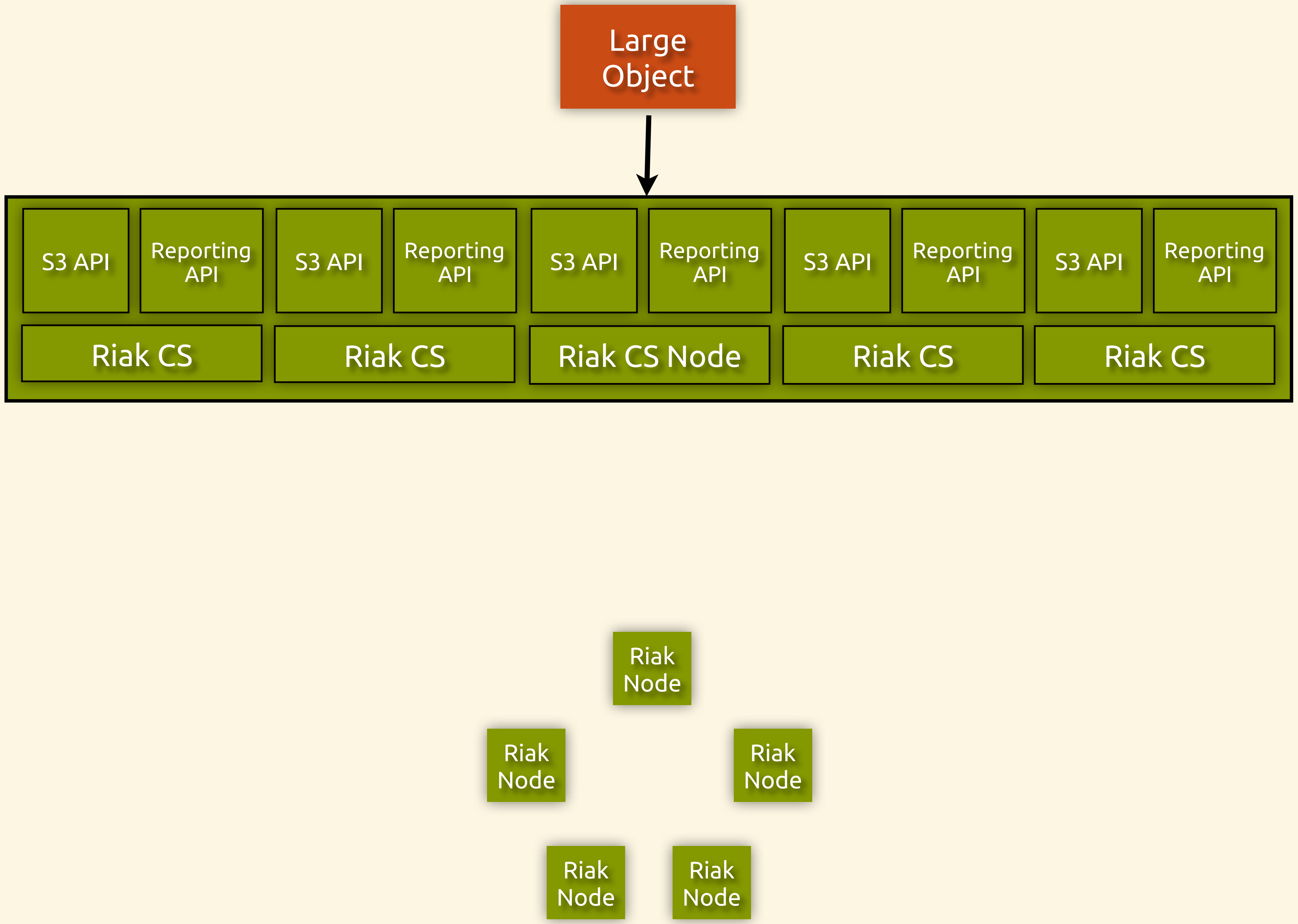
- released March 27, 2012
- S3-compatable cloud storage backed by Riak
- multi-tenancy, private + public clouds
- follows “Stateless Proxy” pattern

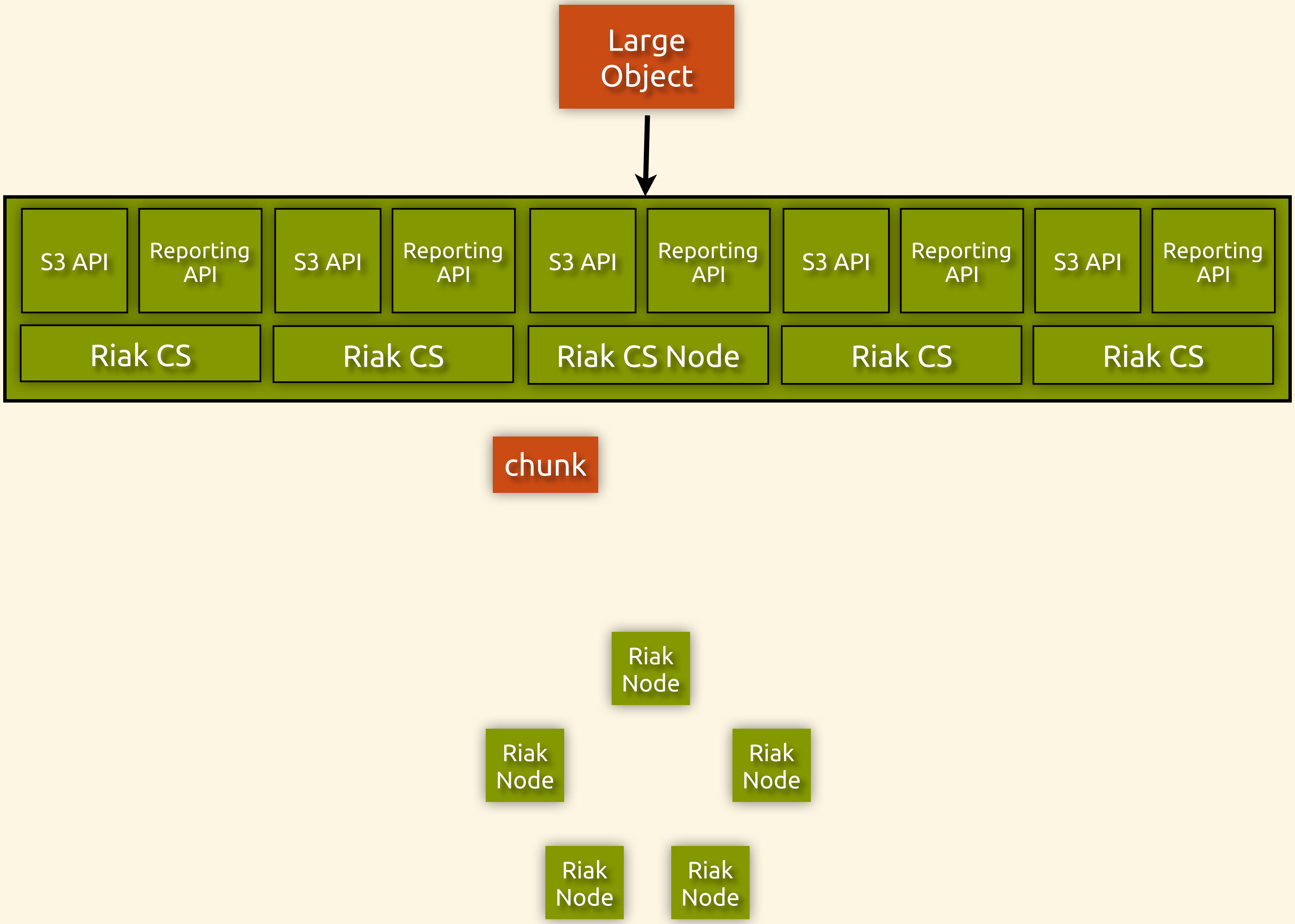


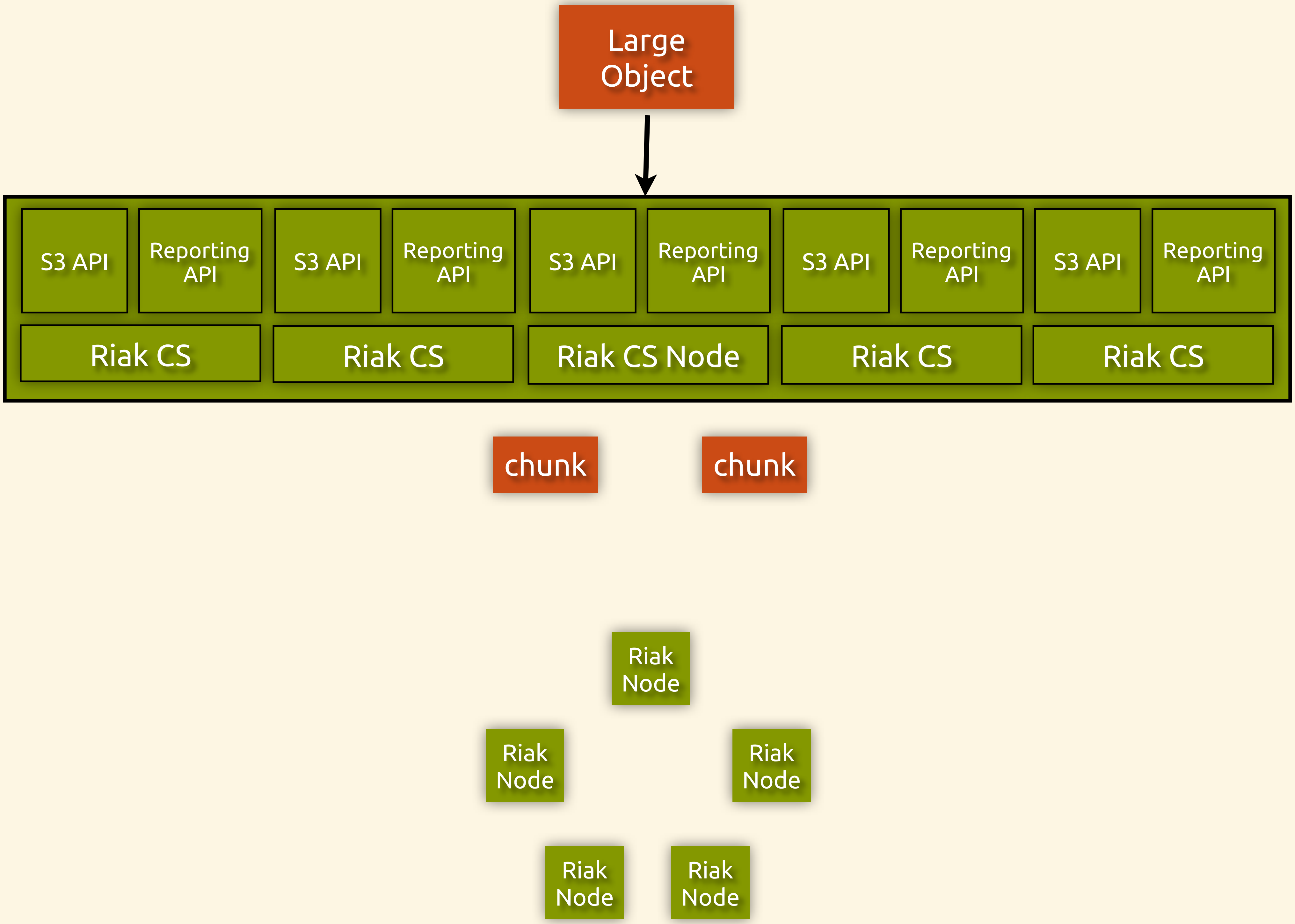
Riak CS Overview

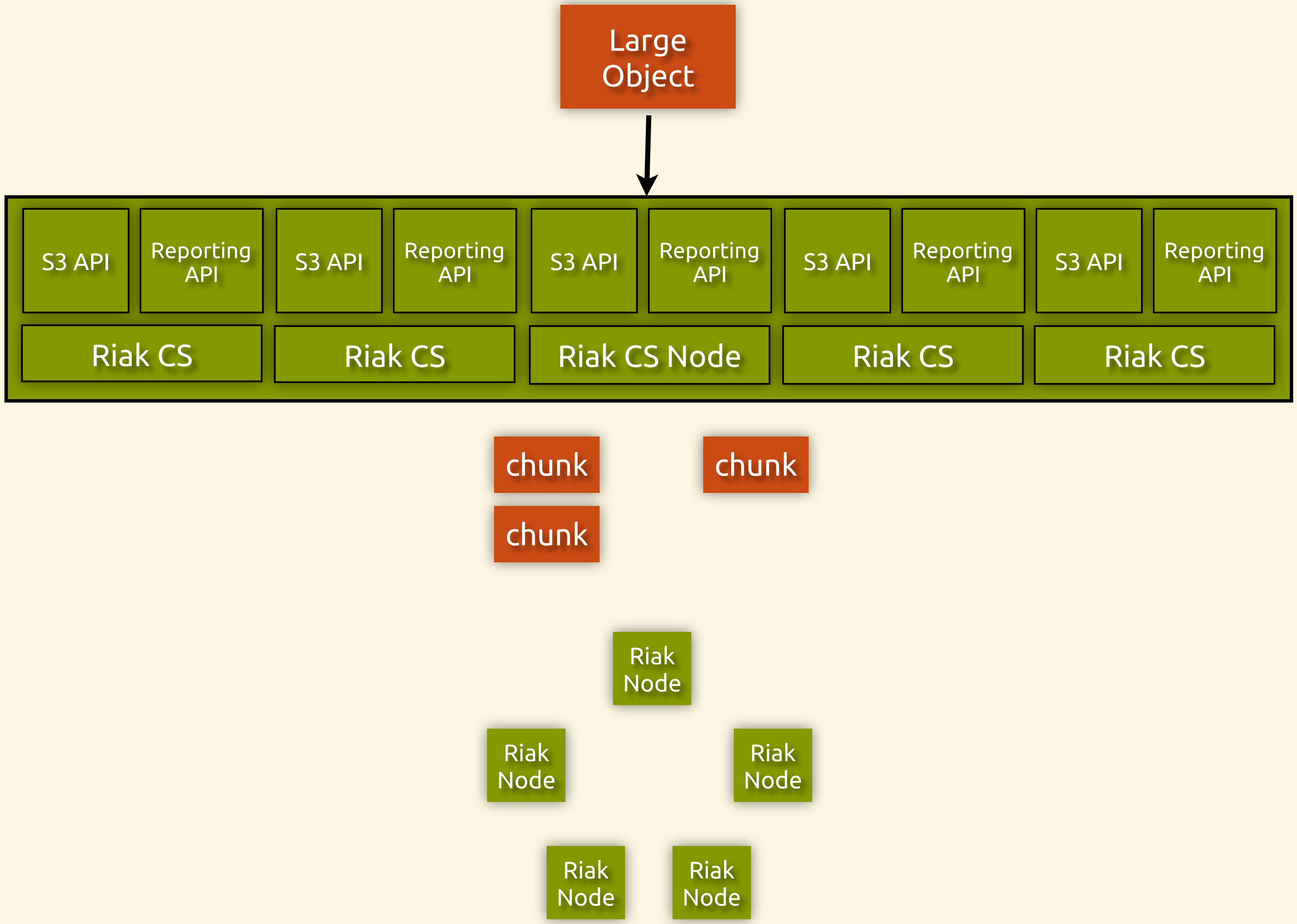
- Implements S3 API via webmachine
- Large files come in through API
- 1+ objects written:
  - manifest: file metadata
  - chunks: statically sized slices of large file

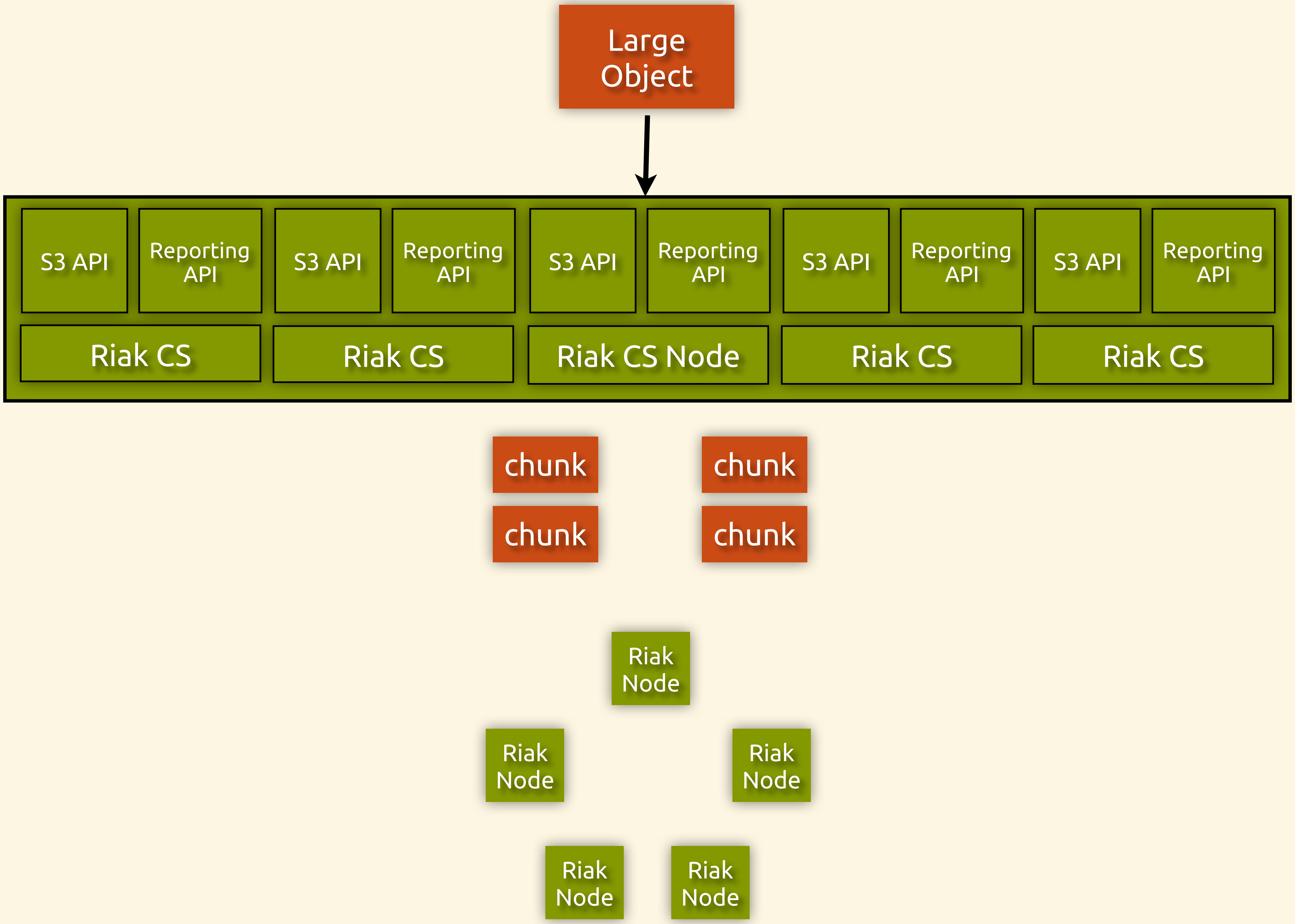

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

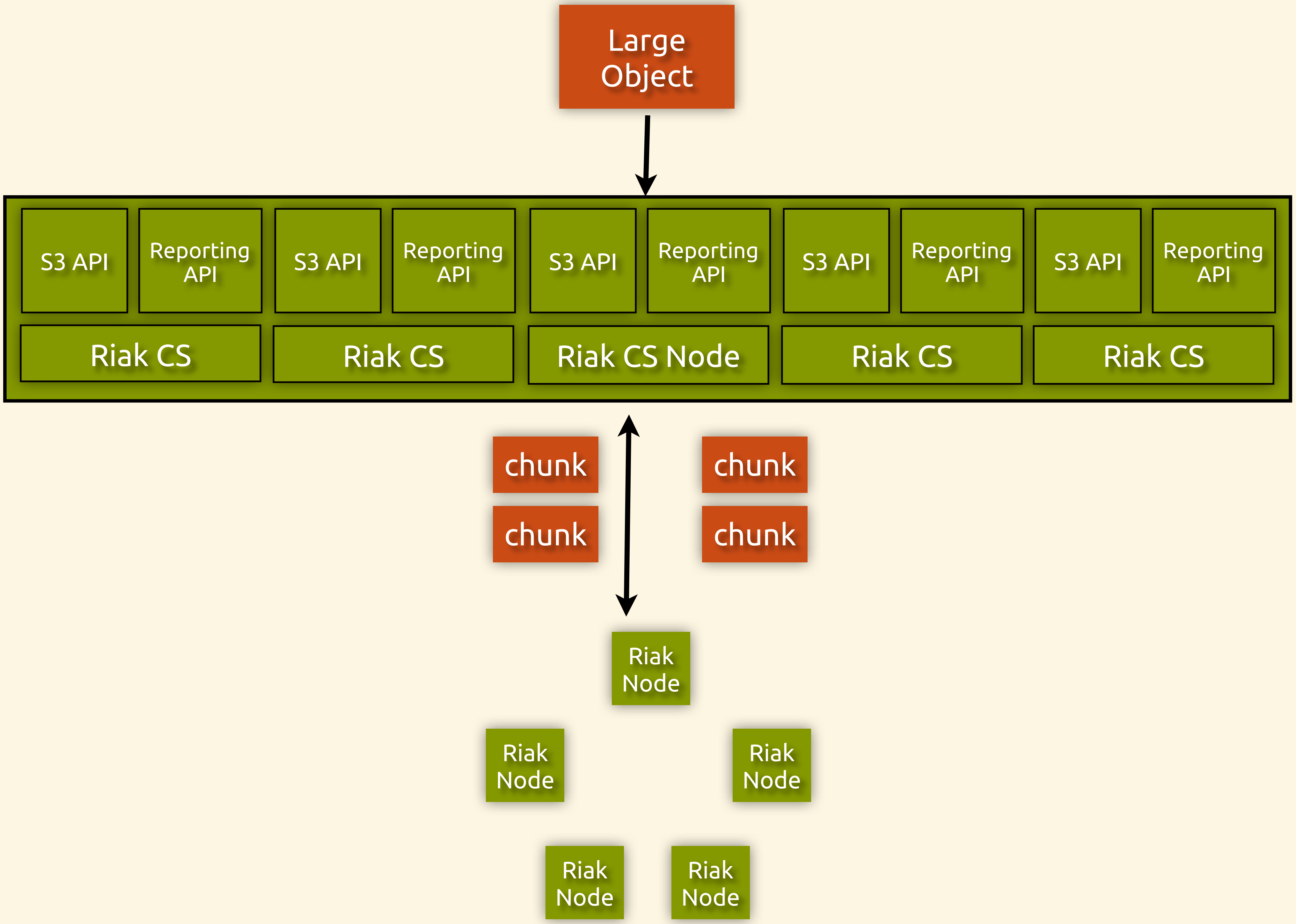

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

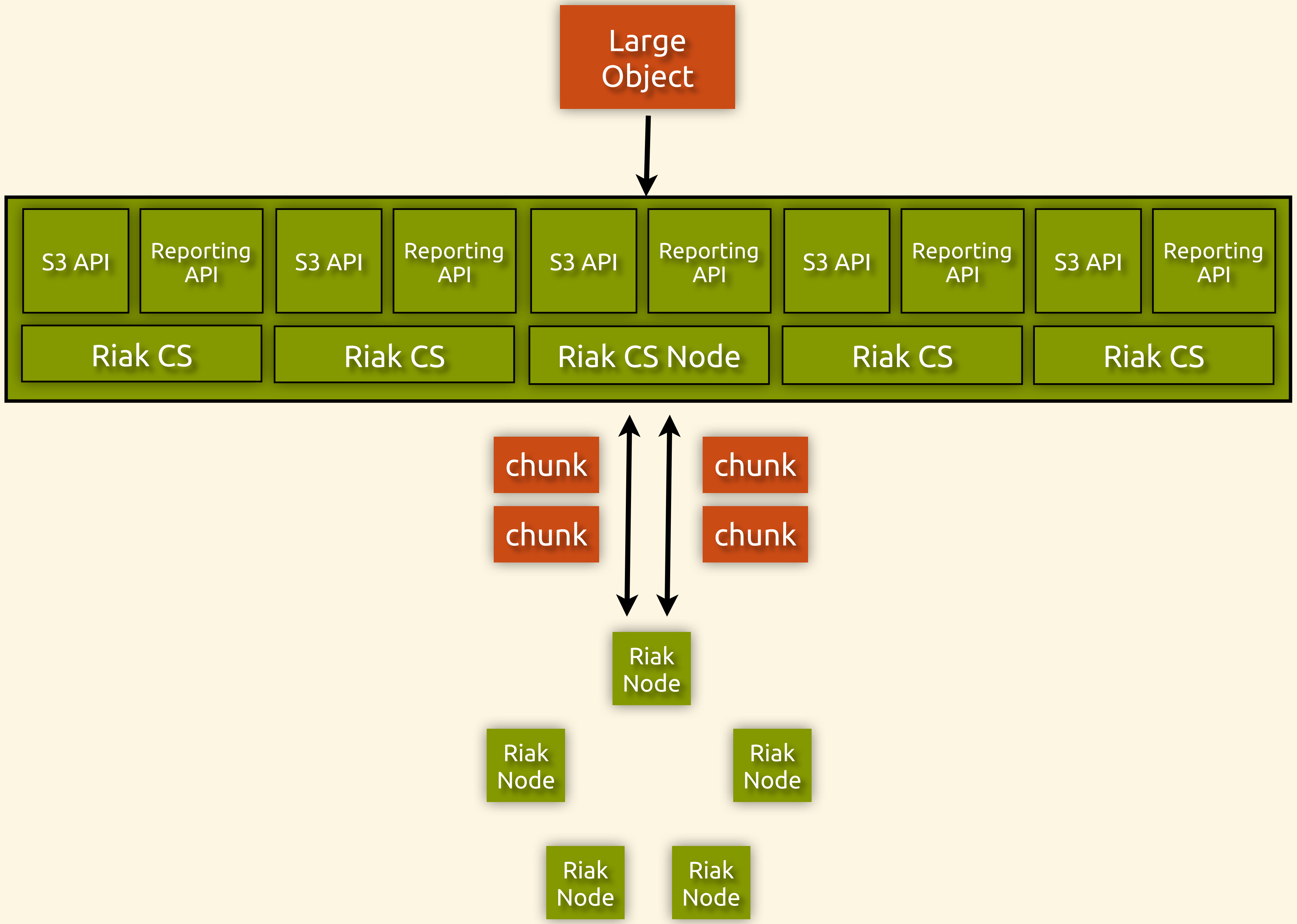

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

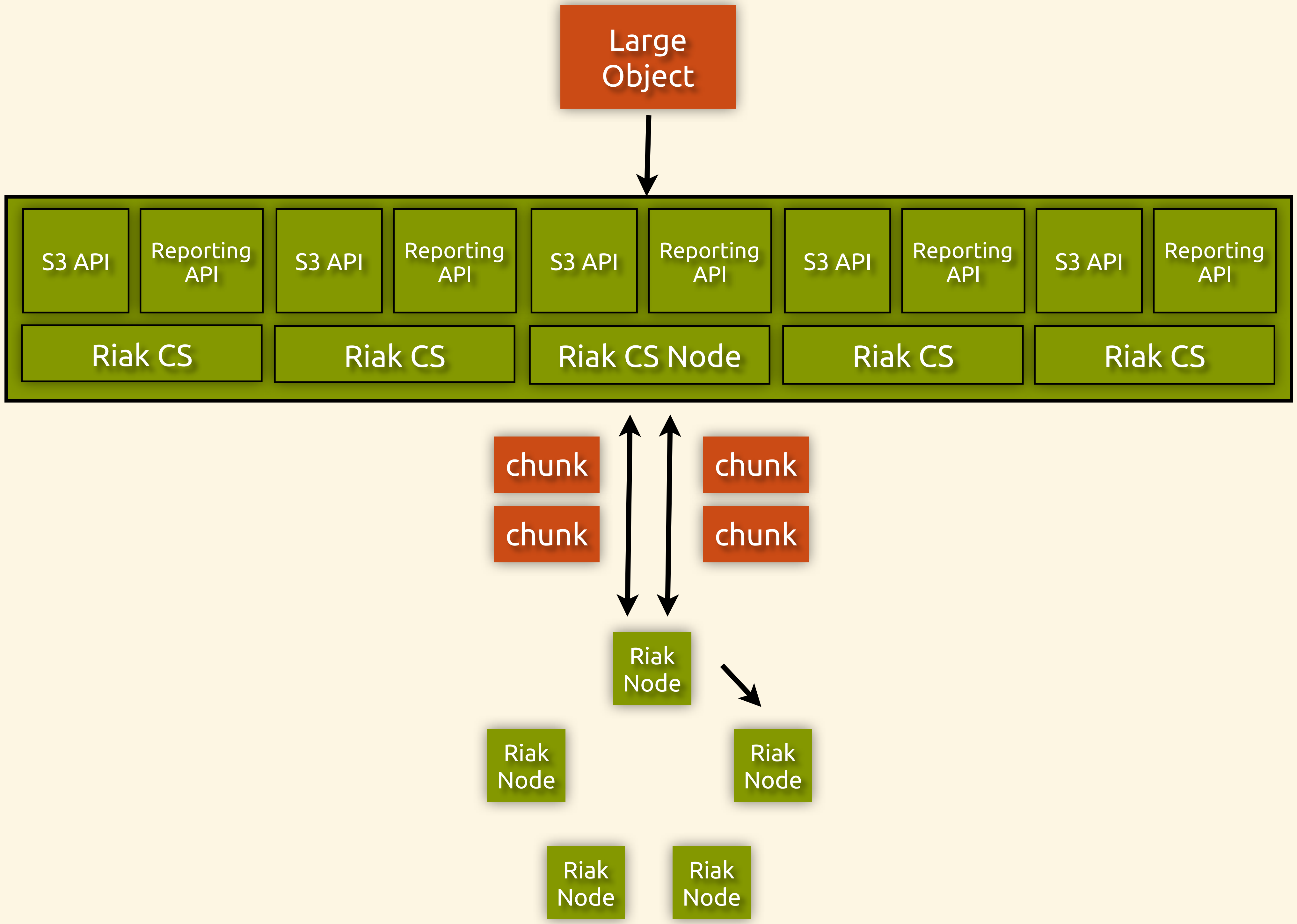

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

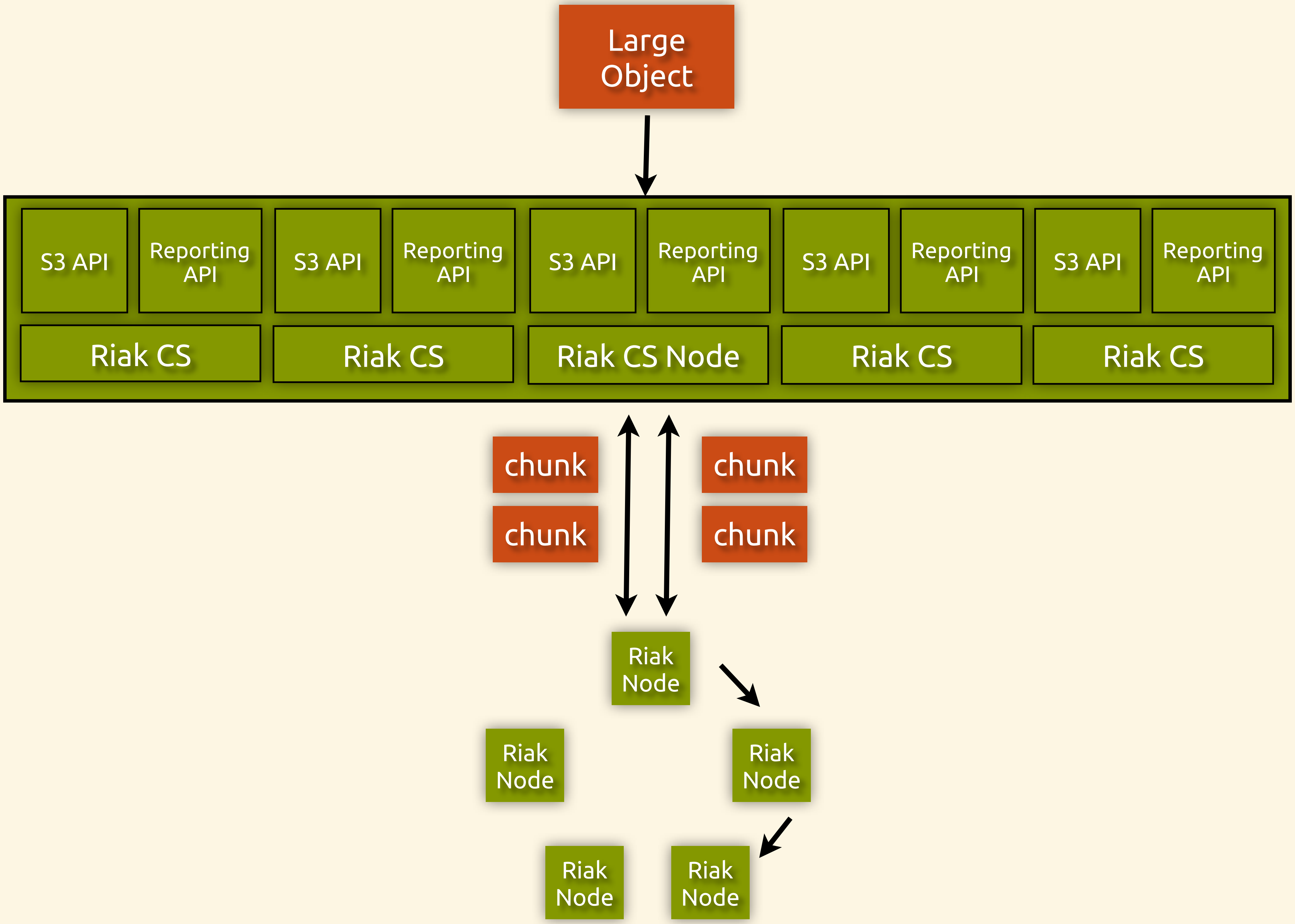

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

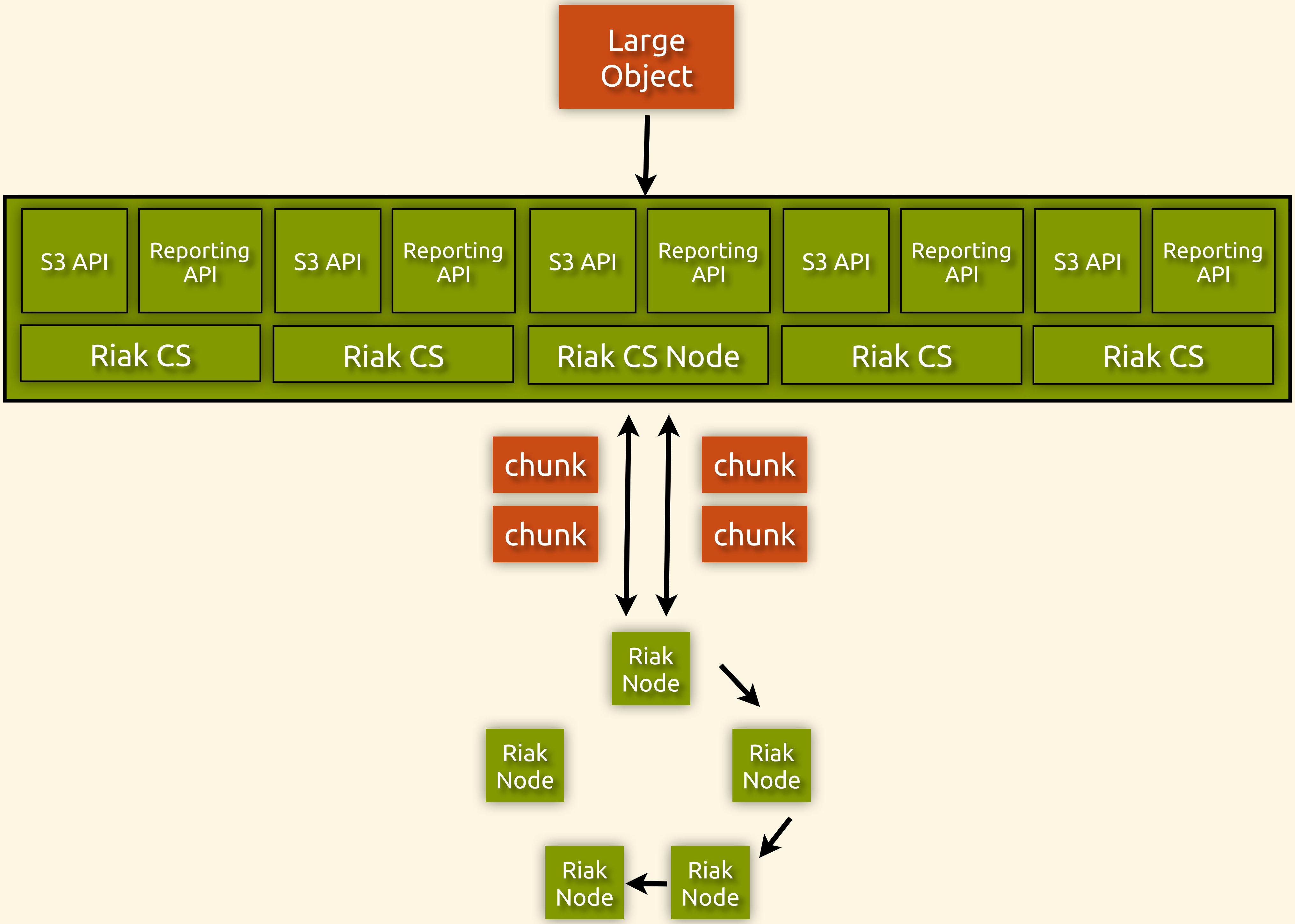

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

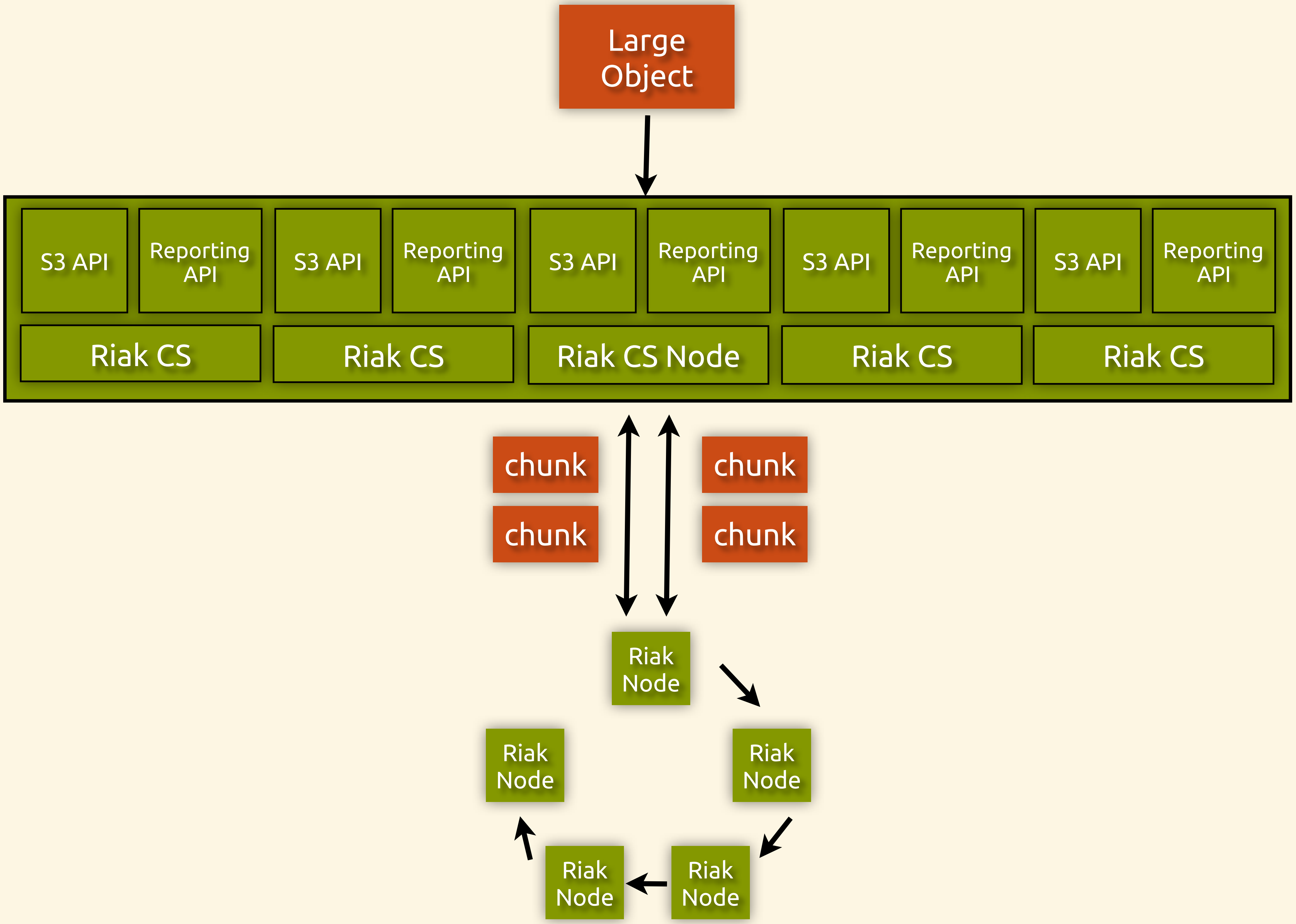

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

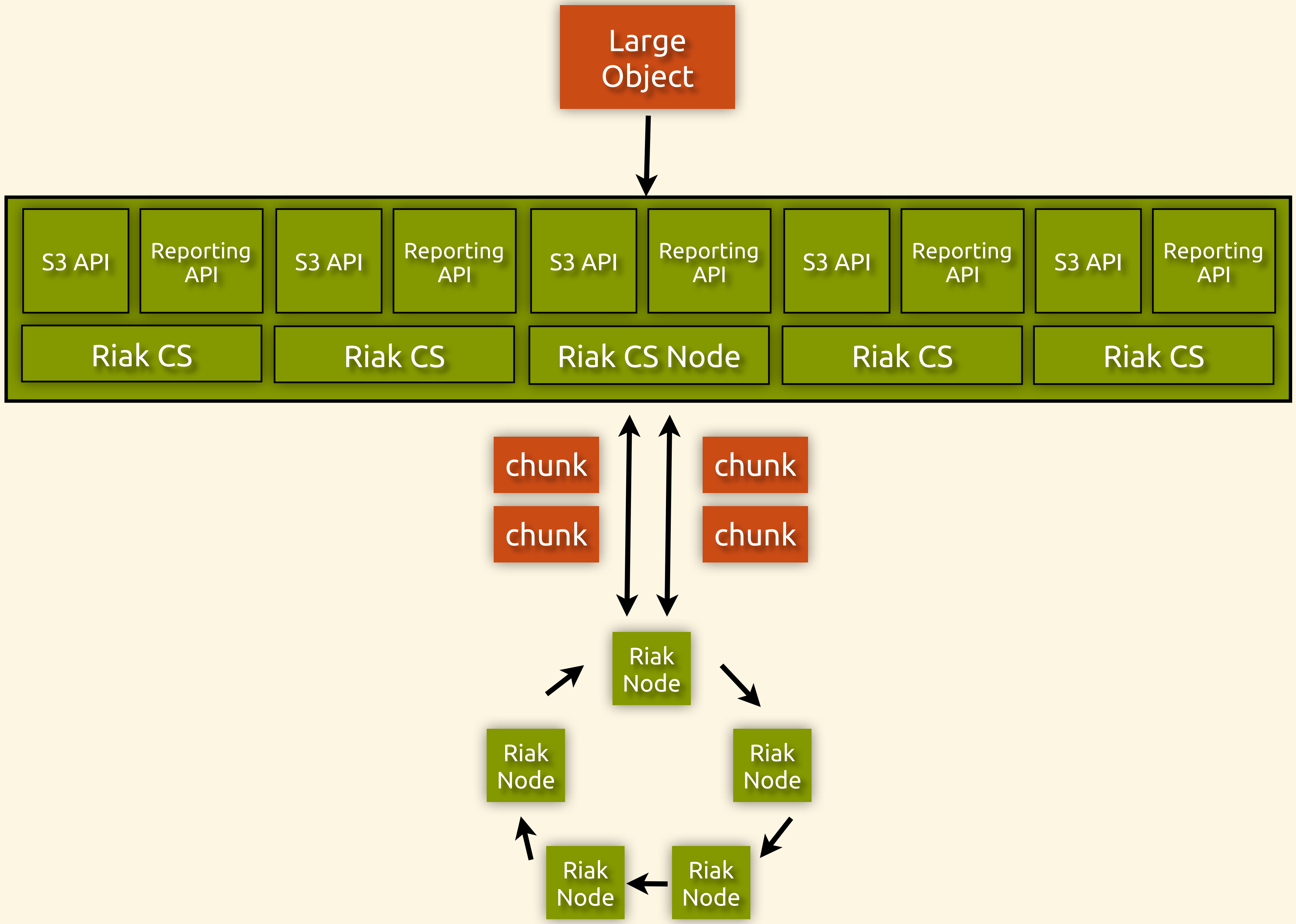

Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node


Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node


Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node


Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node


Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node


Large Object
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
S3 API
Reporting API
Riak CS
Riak CS
Riak CS Node
Riak CS
Riak CS
chunk
chunk
chunk
chunk
Riak Node
Riak Node
Riak Node
Riak Node
Riak Node

Riak CS Use Cases

- storage for cloud computing
- S3 without AWS
- cloud drive (general content storage)
- archival and preservation
- Backup-as-a-Service

Tools We Used

- Erlang
- Rebar
- Quickcheck
- Webmachine
- Other Basho open-source projects

Development Difficulties

Connection Pooling

- just as hard as caching and naming
- # incoming connections > #connection capacity of cluster
    - started with naive approach
    - outsourced to proxy software
    - wrote proper connection pool

Conflict Resolution

- implementation of conflict-handling code can be very tricky
- required for high availability
- CRDTs may help

Strong Consistency

- some S3 operations need to be atomic
- Riak doesn’t support this
- implemented a stopgap solution with less-than-ideal availability properties

Customer Environments

- everything besides Riak and Riak CS
- Software != Service
  - Planning
  - Provisioning
  - Deployment
  - Monitoring

Future Hurdles

Storage Costs

- 3x replication per datacenter = $$$
- erasure coding is a possibility
- smarter global replication
    - notion of “home cluster” N=3
    - others N = 1 || 2

Conclusions

- riak makes a perfect foundation for large scale internet services
- Basho will make more of these
- lots of work to do on the environments riak/riak cs runs in

More Info

- http://wiki.basho.com
- http://github.com/basho/
- http://lists.basho.com/mailman/listinfo/riak-users_lists.basho.com (Riak Mailing List)

Questions?