Using Envoy for data-aware traffic routing in Service Fabric

Vaclav Turecek
Principal Program Manager
Microsoft

Service Fabric overview

What is it?

- Data-aware distributed systems platform

What does it do?

- Manages containers, processes, and data on a cluster of Windows or Linux hosts

How do we use it?

- Runs Azure and most Microsoft cloud services

Can anyone use it?

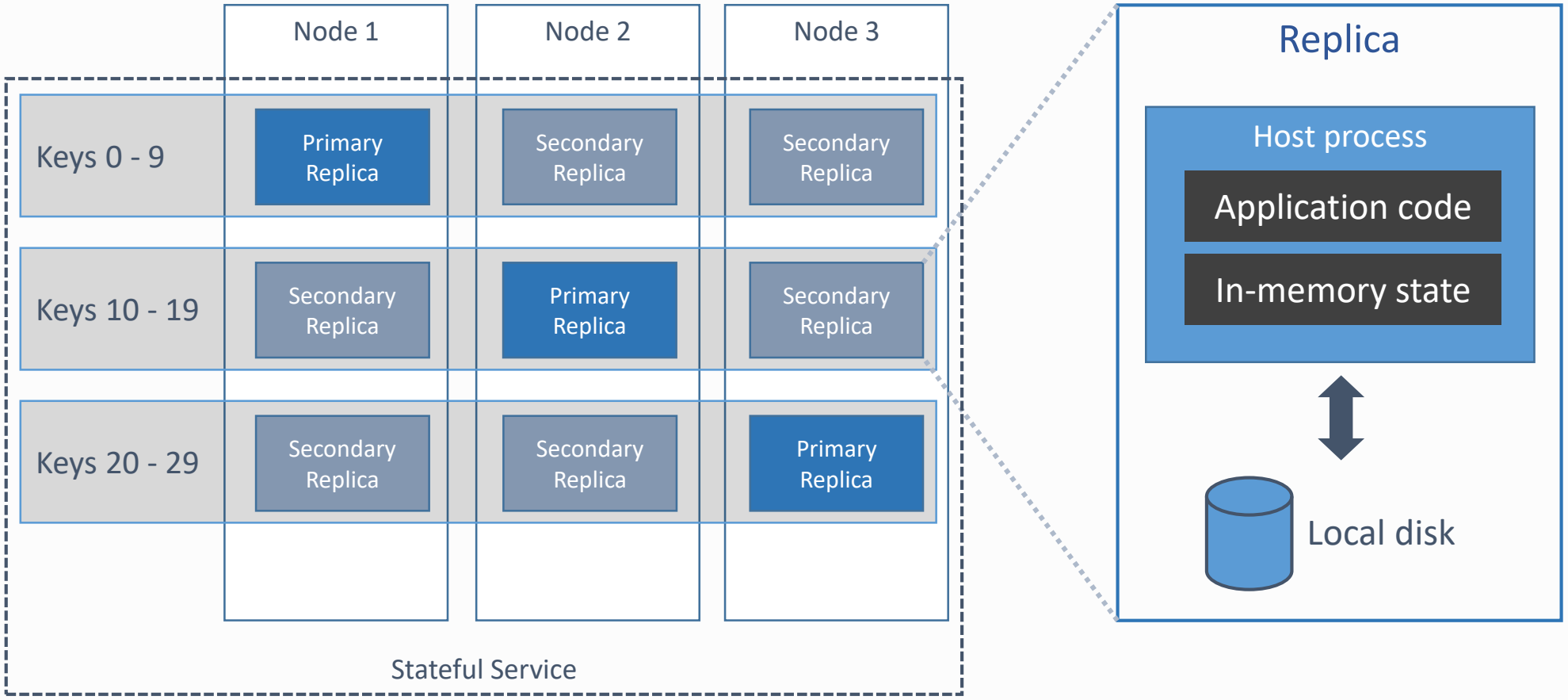
- github.com/Microsoft/service-fabric

Service Fabric overview

Designed for stateful applications with built-in distributed data primitives

- Clustering, federation, arbitration
- Leader election, consensus, reconfiguration
- Replication, quorum consistency, weighted balancing
- Partitioning (data sharding)

Stateful compute

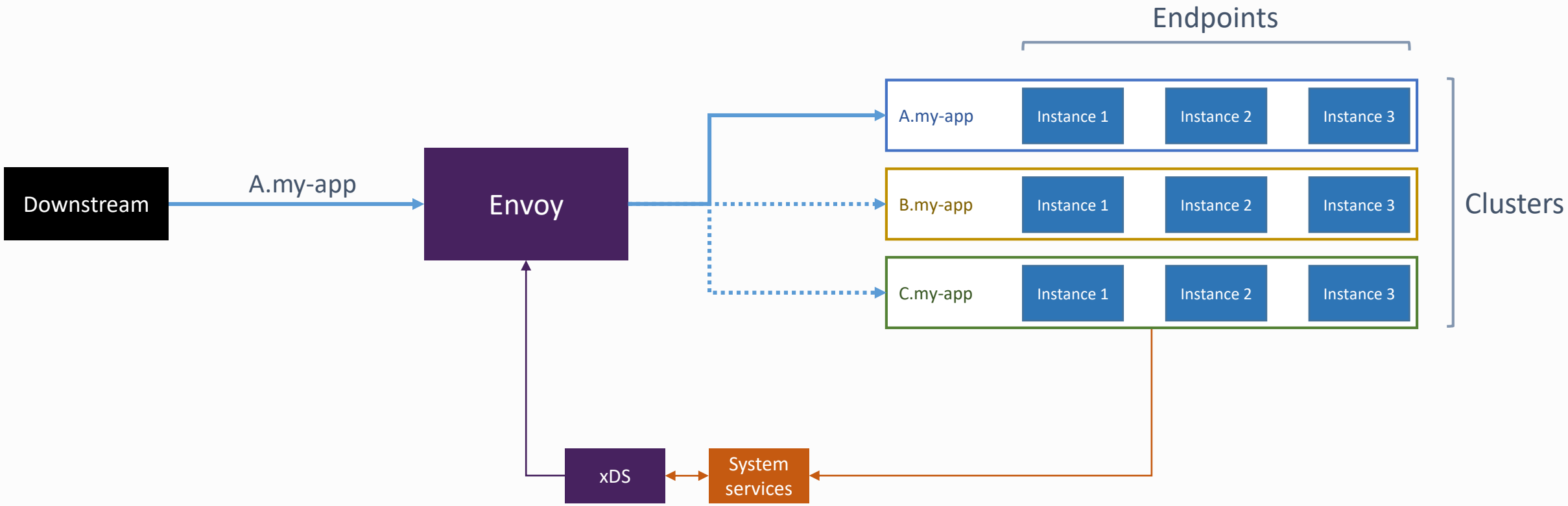


Stateful compute

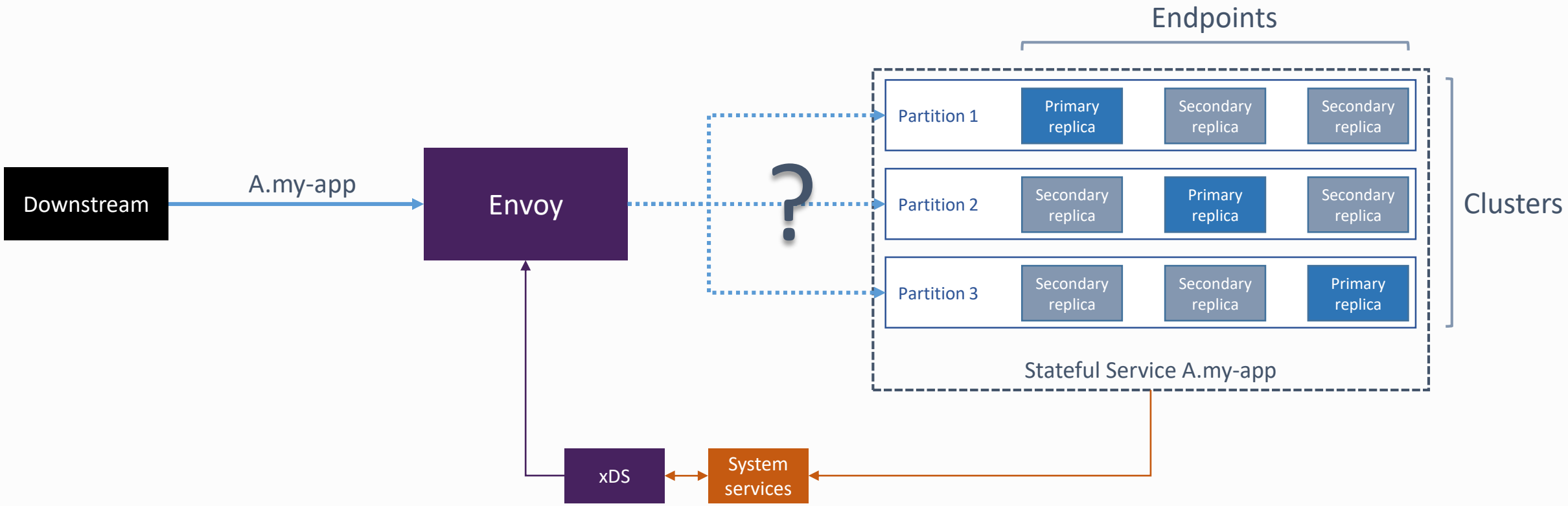
Traffic routing must be data aware too

- Service discovery and routing
- Load balancing
- Health checking
- Versioning and upgrades

Service discovery and routing



Service discovery and routing

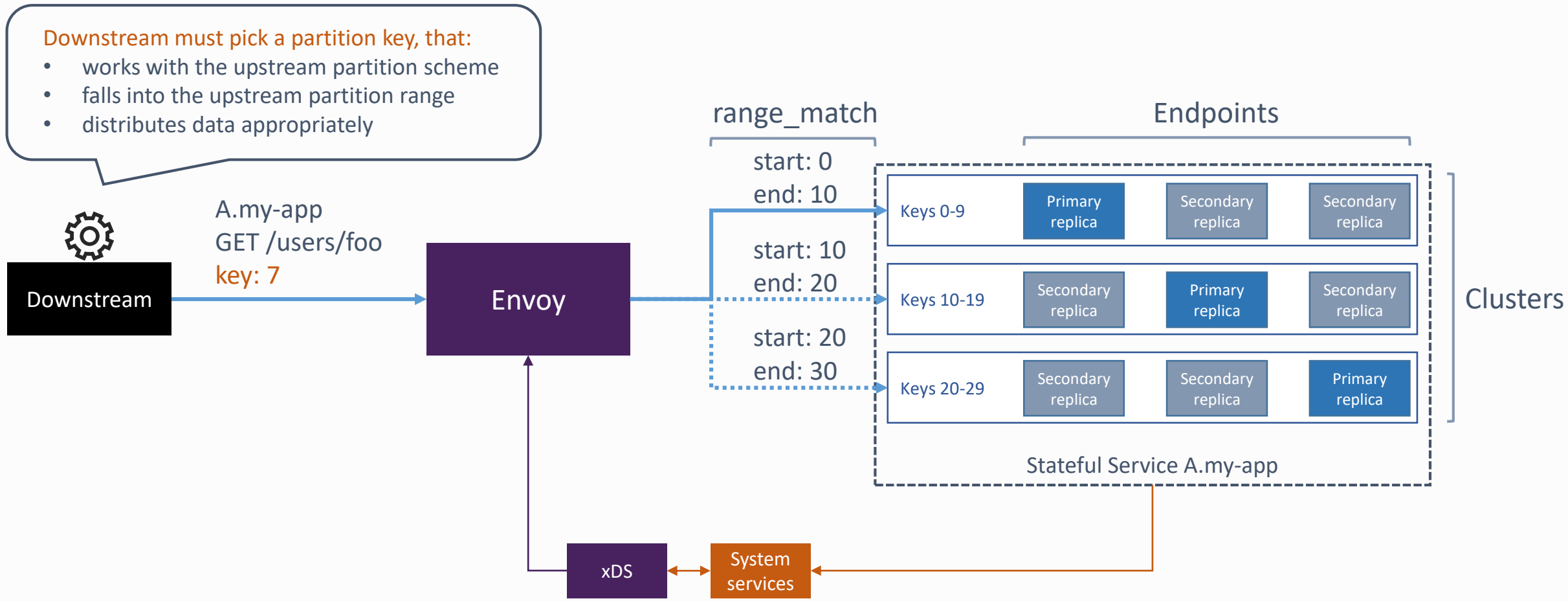


Service discovery and routing

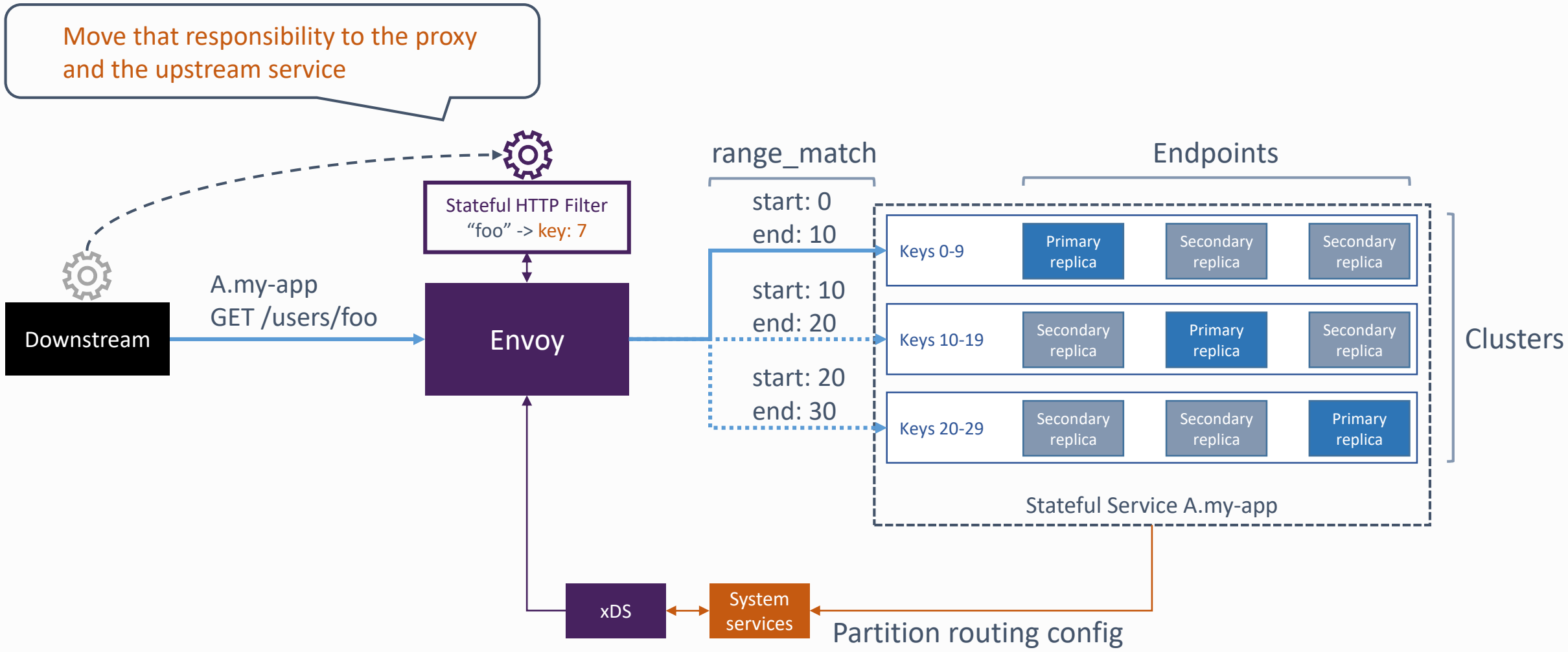
Partition resolution

1. Take some data from the request
   - Typically path, header, or query string value for HTTP
2. Transform it into a partition key
   - Determines distribution of data
3. Look up the location of the matching partition
   - Set of replica endpoints
4. Forward the request

Service discovery and routing



Service discovery and routing

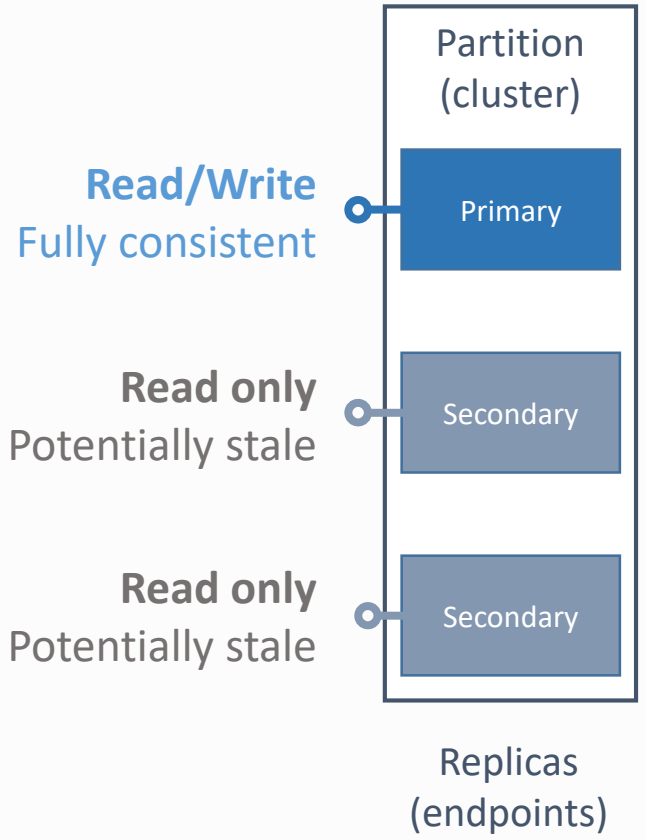


Load balancing

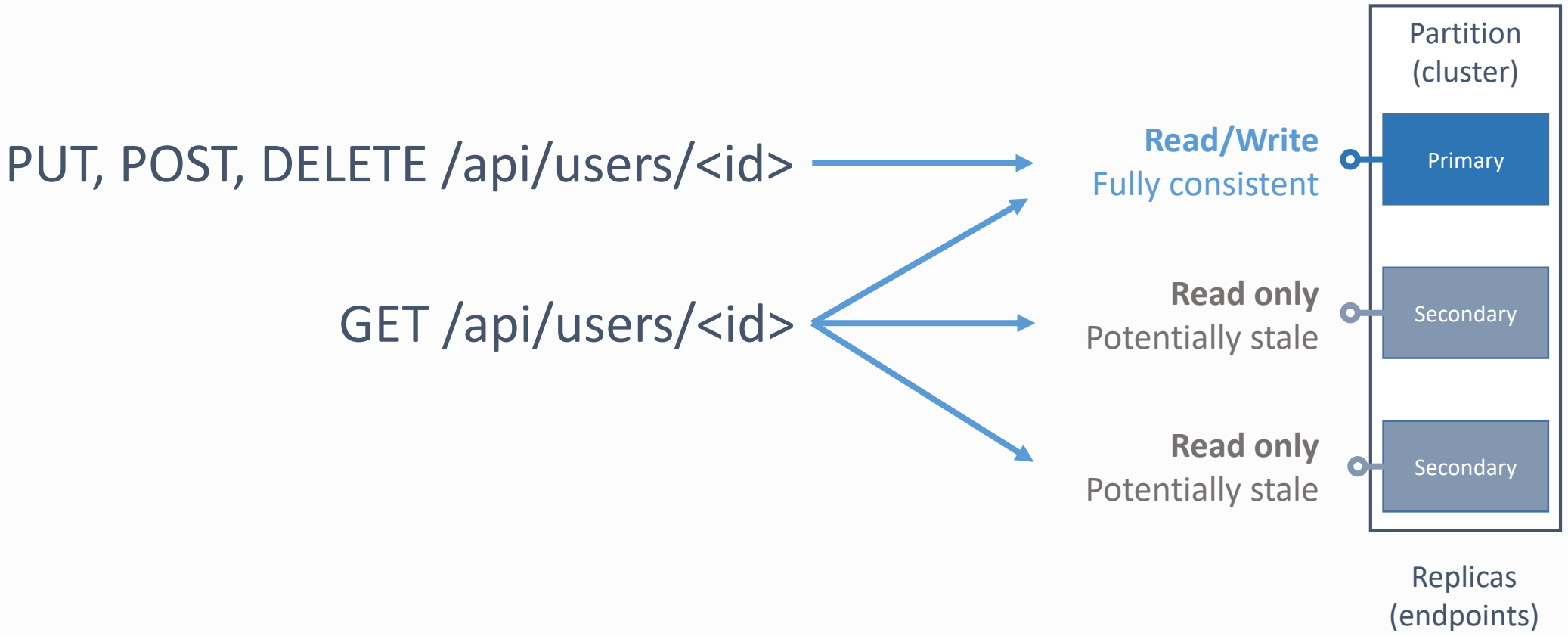
envoycon

Consistency vs. Availability

- Full consistency at cost of availability
  - Only one endpoint in cluster: the primary
- Higher read availability at cost of consistency
  - Writes go to primary only
  - Reads go to any replica when full consistency is not required



Load balancing



Load balancing

envoycon

HTTP route

```
match:
  prefix: /api/users
  headers:
  - name: ":method"
    exact_match: "POST"

route:
  cluster: partition-guid
  metadata-match:
    filter_metadata:
      envoy.lb:
        role: "primary"
```

RDS

Cluster

```
name: partition-guid
type: EDS
lb_subset_config:
  subset_selectors:
  - keys:
    - role
```

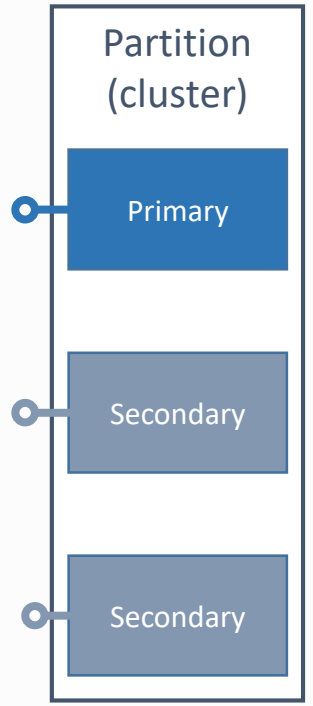
CDS

LbEndpoint

```
metadata:
  filter_metadata:
    envoy.lb:
      role: "primary"
```

```
role: "secondary"
```

```
role: "secondary"
```



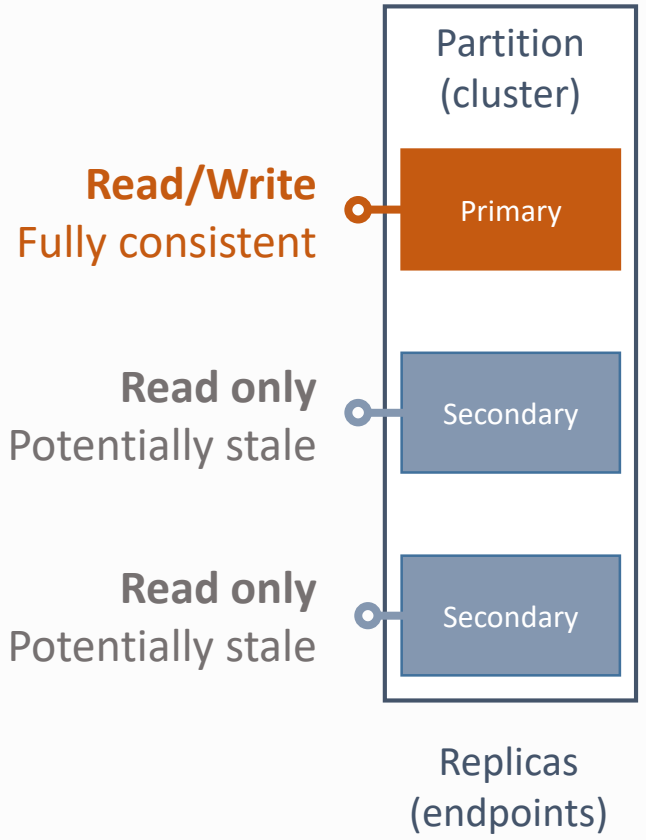
EDS

Replicas
(endpoints)

Health checking

So your primary replica is unhealthy..

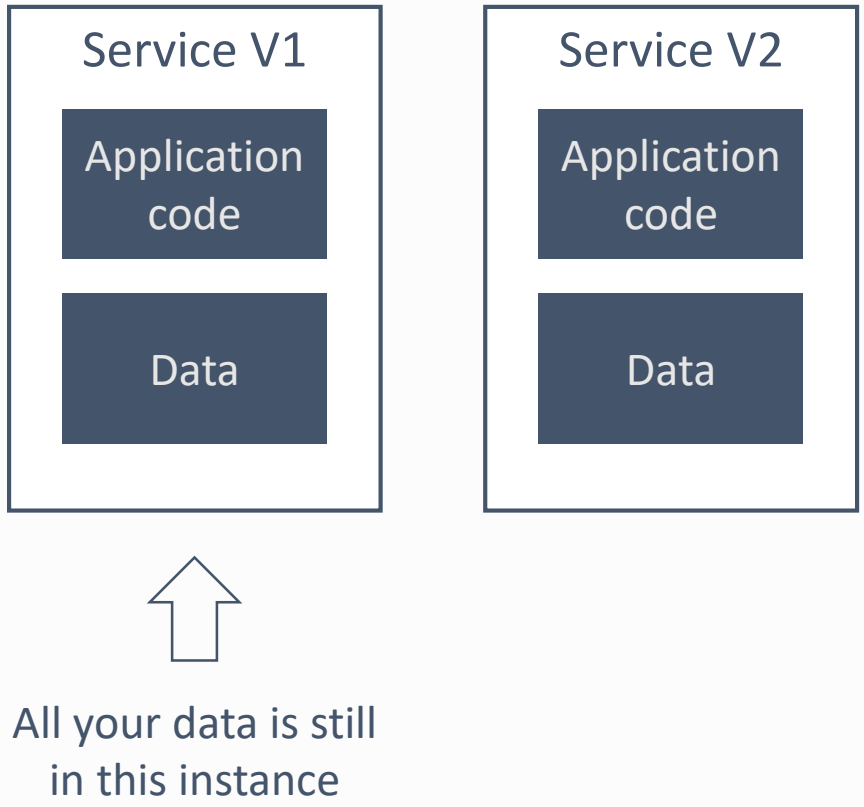
- Can’t always route to a secondary or to a different partition.
- Let system failover the primary to a different node.



Versioning and upgrades

Stateful services must be upgraded in place

- Upgrade domain walk
- Cooperative effort
- Traffic shadowing for A/B testing



Porting to Windows

- Mostly OS-agnostic
- The rest:
  - 30% API
  - 70% behavior
- Build system was a challenge

Porting to Windows

Some error codes don’t exist on Windows

```
#define ESHUTDOWN          WSAESHUTDOWN
```

…

Other error codes need to be redefined

```
auto err = WSAGetLastError();
  switch (err) {
  case WSAEINTR:
    return EINTR;
```

…

Porting to Windows

int close(int fd);

- API exists on both Linux and Windows
- Closes a socket on Linux, but not on Windows
- Still compiles and runs on both

int closesocket(IN SOCKET s);

- Must use this to close a socket on Windows

Porting to Windows

Libevent performance on Windows isn’t great

- No edge triggering on Windows
- Wasted CPU time waiting to read and write to sockets
- Ideally use I/O completion ports

Why Envoy?

- C++
- Fast
- Extendable
- Feature rich
- Excellent community