Altinity

Democratizing Analytics

Cloud native data warehouses on Kubernetes

Robert Hodges and Vlad Klimenko

Presenting the Presenters

Robert Hodges
CEO Altinity

30+ years on DBMS including 20 different DBMS types. Working on Kubernetes since 2018

Vlad Klimenko
Senior Software Engineer, Altinity

15+ years of database and application experience. Main designer of ClickHouse Kubernetes operator.

Introducing ClickHouse

ClickHouse is an open source data warehouse

Understands SQL

Runs on bare metal to cloud

Shared nothing architecture

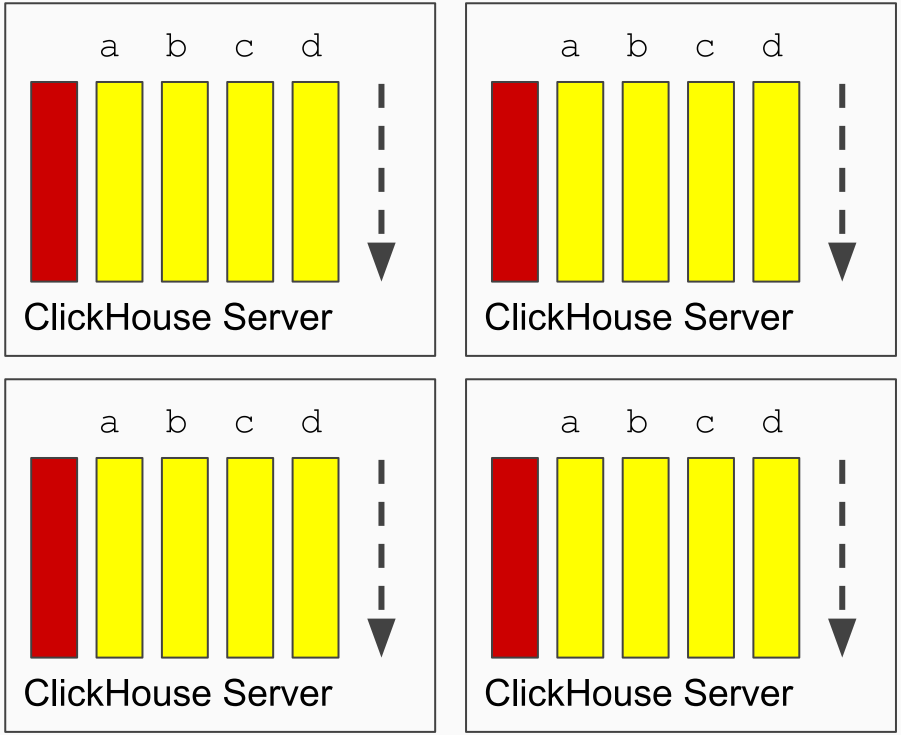
Stores data in columns

Parallel and vectorized execution

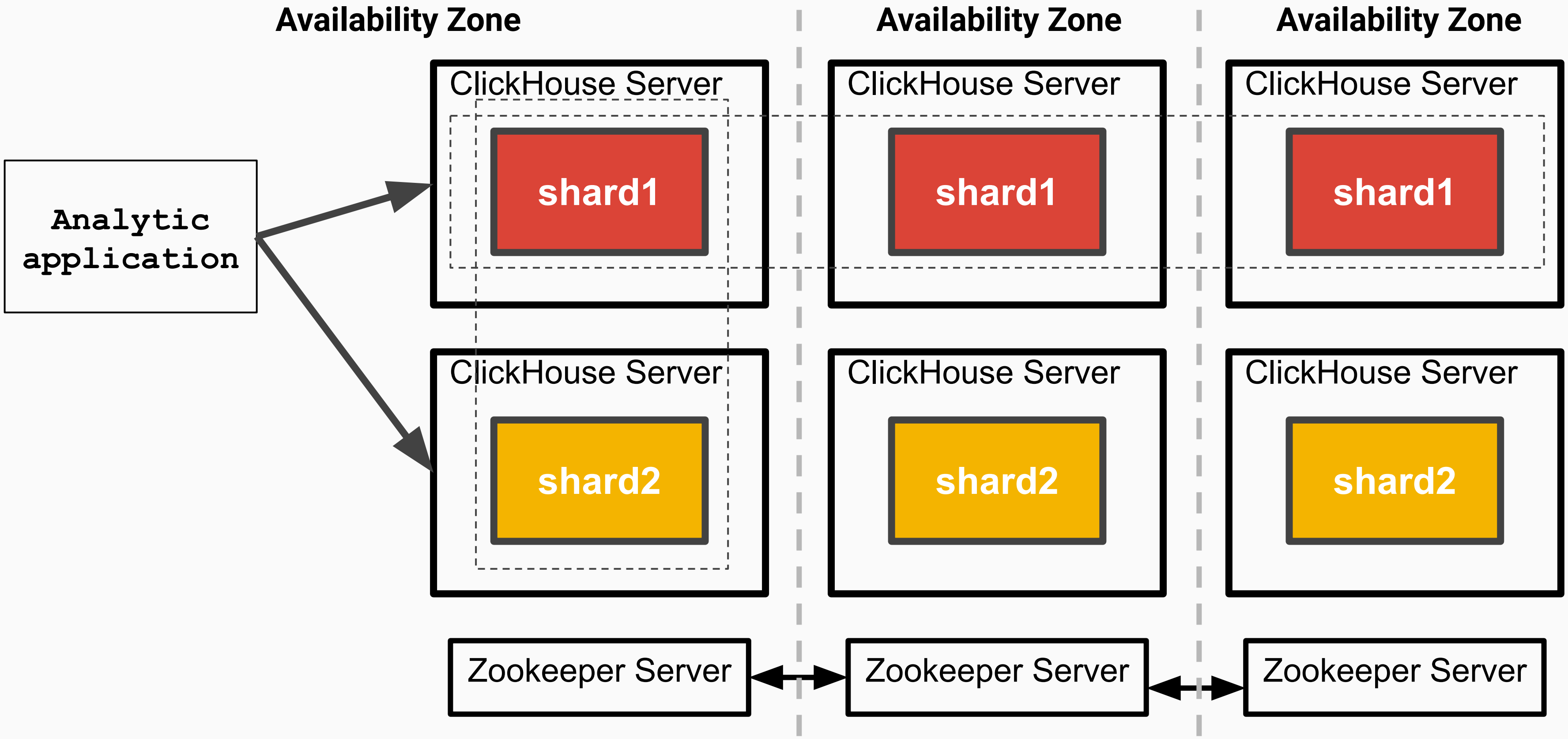
Scales to many petabytes

Is Open source (Apache 2.0)

And it's <u>really</u> fast!



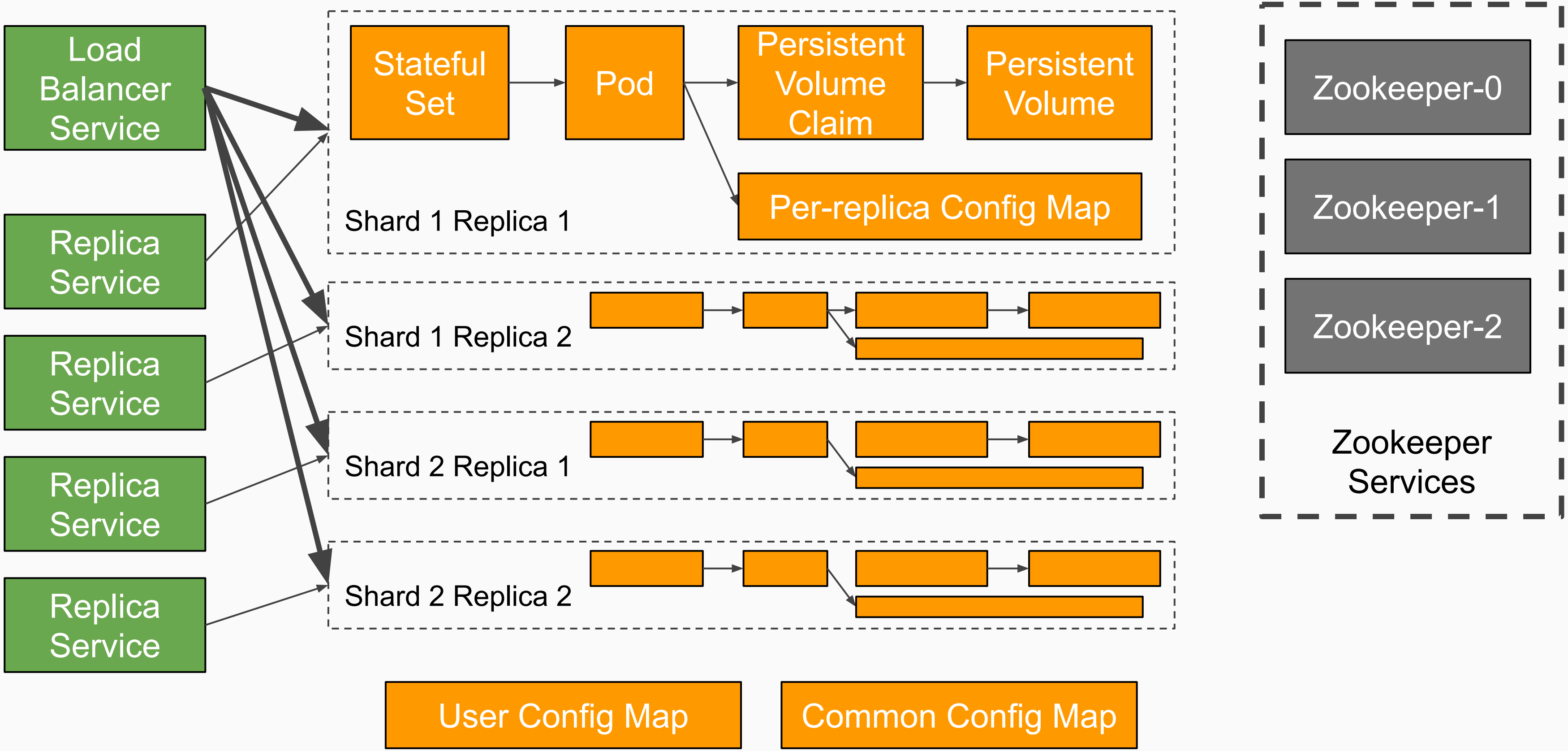
ClickHouse is also a distributed application



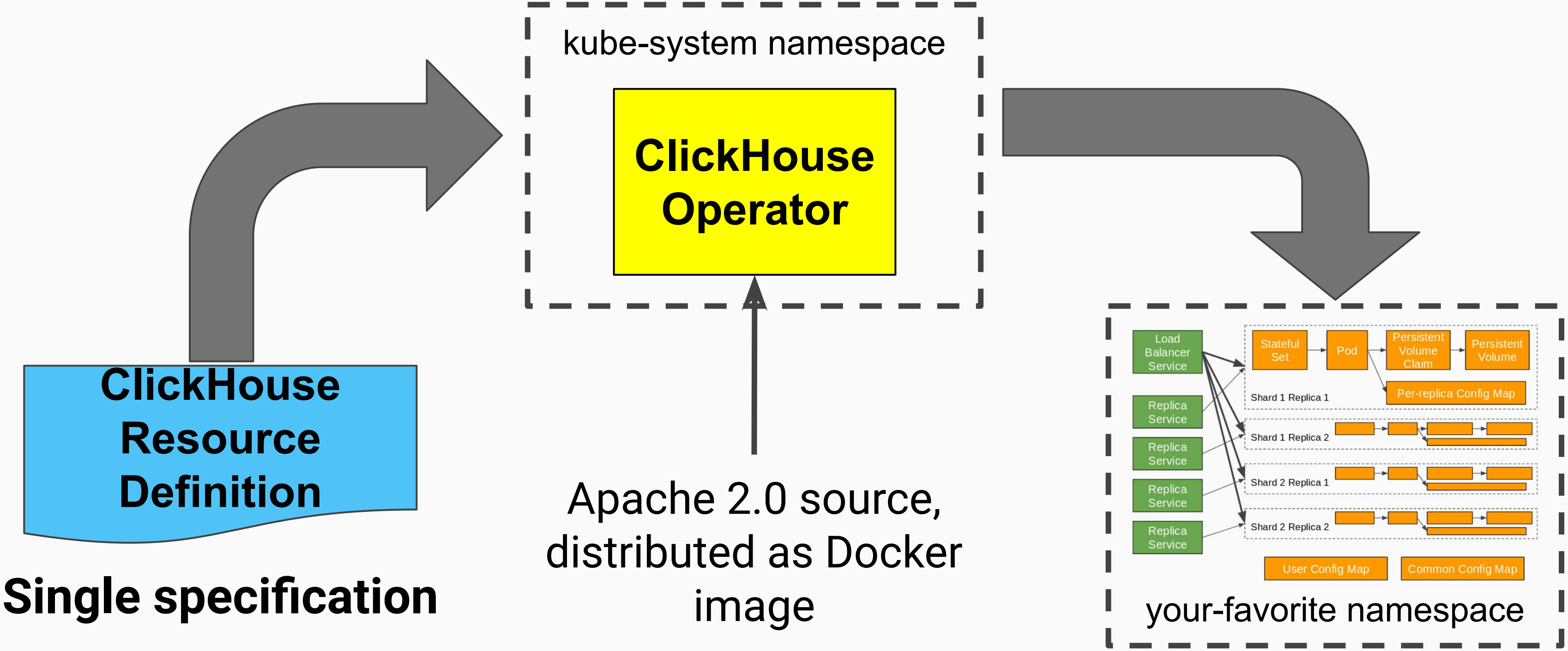
Altinity

Going Cloud Native -- the User View

ClickHouse on Kubernetes is complex!



Operators encapsulate complex deployments



Single specification

Apache 2.0 source, distributed as Docker image

Best practice deployment

ClickHouse operators are easy to install

Get operator custom resource definition:

```
wget \
https://raw.githubusercontent.com/Altinity/clickhouse-operator/master/deploy/operator/clickhouse-operator-install.yaml
```

Install the operator:

```
kubectl apply -f clickhouse-operator-install.yaml
```

Remove the operator:

```
kubectl delete -f clickhouse-operator-install.yaml
```

You need at least one Zookeeper ensemble

Simplest way is to use helm:

```
kubectl create namespace zk

helm install --namespace zk --name zookeeper \
incubator/zookeeper
```

(There’s also an operator for Zookeeper now)

Setting up a data warehouse--the basics

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "cncf"
spec:
  configuration:
    clusters:
    - name: replicated
      layout:
        shardsCount: 2
        replicasCount: 1
  zookeeper:
    nodes:
     - host: zookeeper.zk
```

Name used to identify all resources

Definition of cluster

Location of service we depend on

Adding users and changing configuration

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "cncf"
spec:
  configuration:
    users:
      demo/password: demo
      demo/profile: default
      demo/quota: default
      demo/networks/ip: "::/0"
    clusters:
    - name: replicated
```

Changes take a few minutes to propagate

Templates make it easy to define defaults

```
defaults:
  templates:
    volumeClaimTemplate: persistent
    podTemplate: clickhouse:19.6
templates:
  volumeClaimTemplates:
    - name: persistent
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
```

Name of template

Storage misconfigurations lead to insidious errors

Scale up/down by modifying layout

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "cncf"
spec:
  configuration:
    clusters:
    - name: replicated
      layout:
        shardsCount: 3
        replicasCount: 3
```

Increase shards to add write capacity

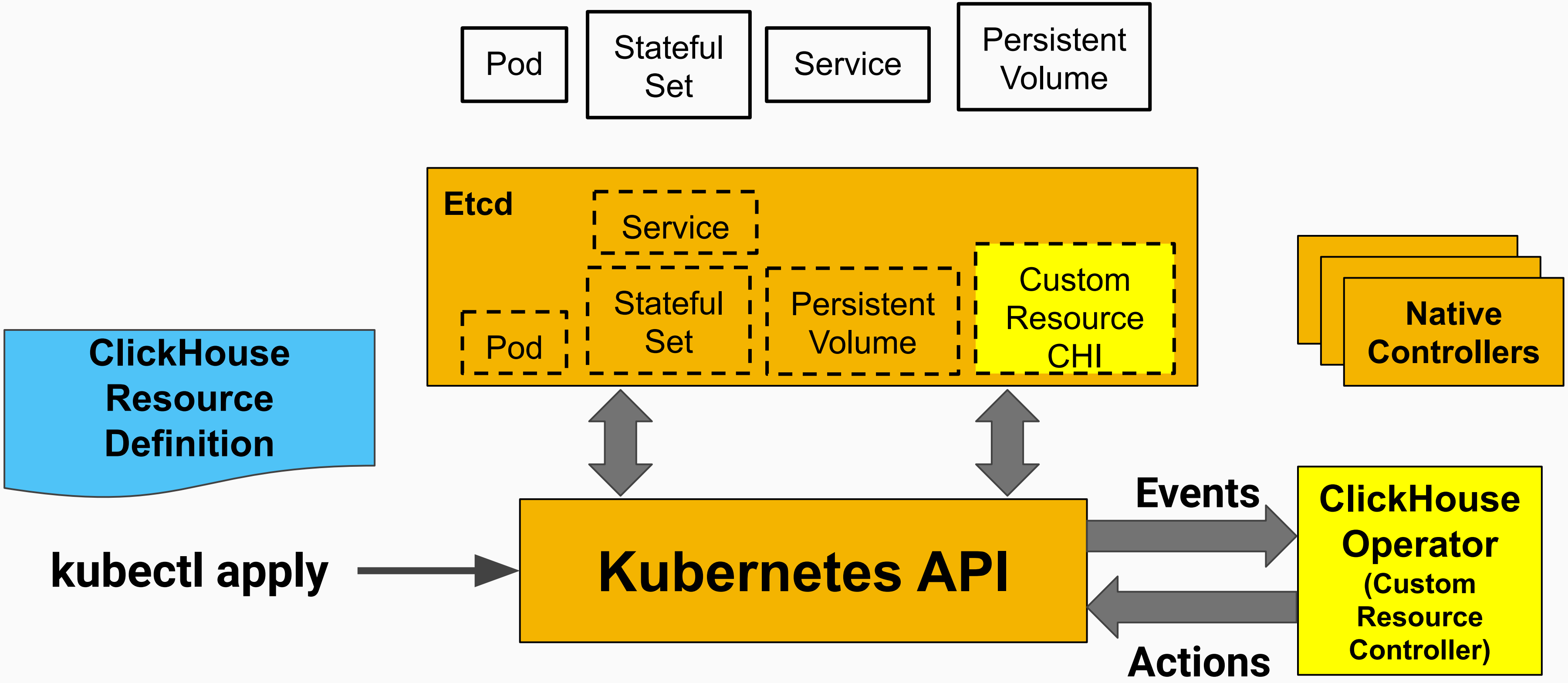
Raise replicas to add read capacity

Demo Time!

ClickHouse
on
Kubernetes
in Action

Going Cloud Native -- Inside the Operator

Operators and event processing



Pod

Stateful Set

Service

Persistent Volume

Etcd

Service

Pod

Stateful Set

Persistent Volume

Custom Resource CHI

Native Controllers

ClickHouse Resource Definition

kubectl apply

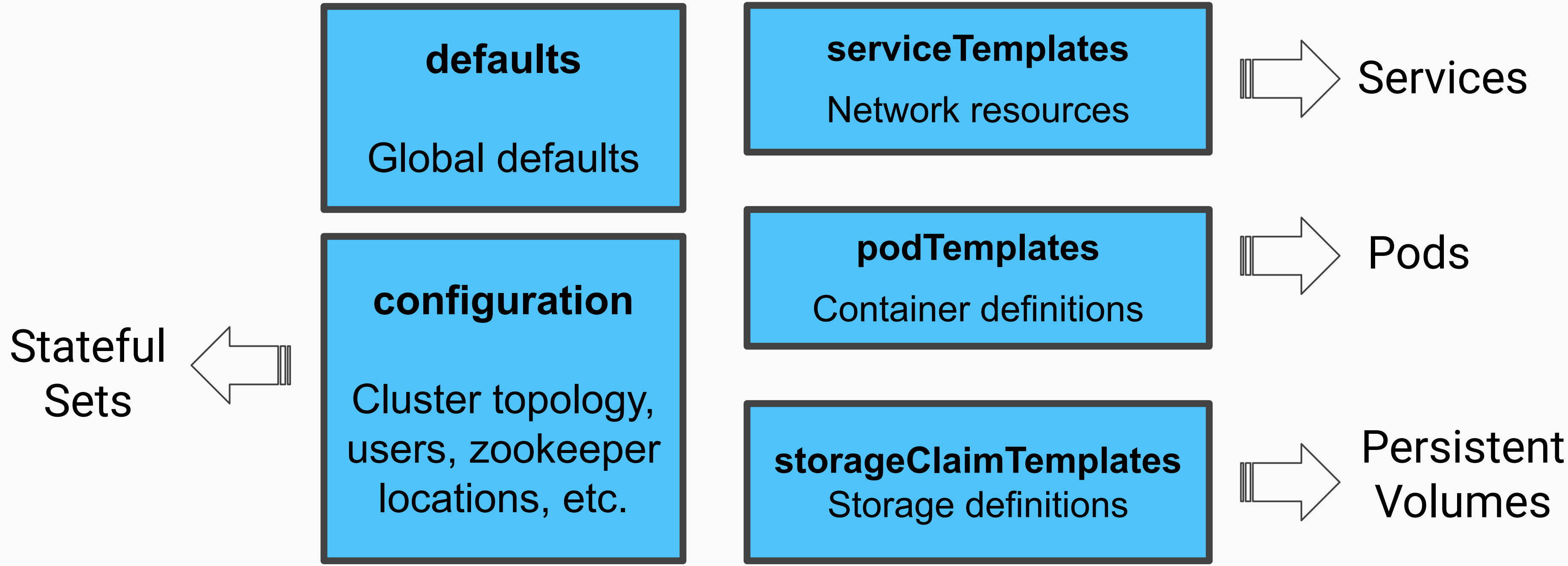
Kubernetes API

Events

Actions

ClickHouse Operator (Custom Resource Controller)

Clickhouse custom resource definition



defaults

Global defaults

configuration

Cluster topology, users, zookeeper locations, etc.

Stateful Sets

serviceTemplates

Network resources

Services

podTemplates

Container definitions

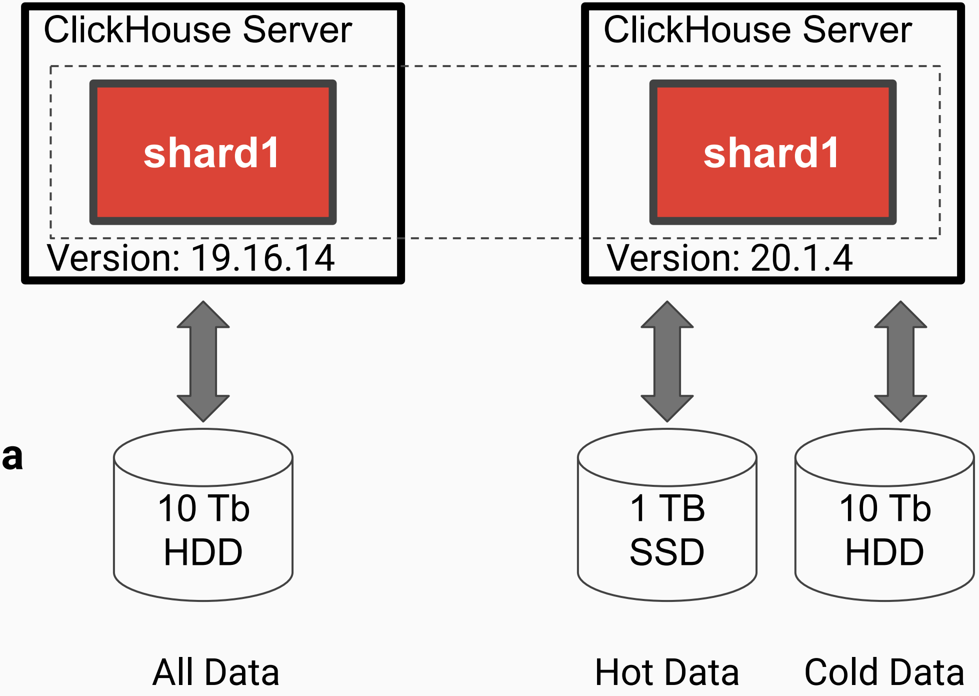
Pods

storageClaimTemplates

Storage definitions

Persistent Volumes

Node variation is common in data warehouses



Canary Replica on new version with tiered storage

<u>May run for weeks!</u>

Production Replica on stable version with single HDD

Stateful sets don’t quite match our node model

🙂

Pods have identical configuration and non-varying storage

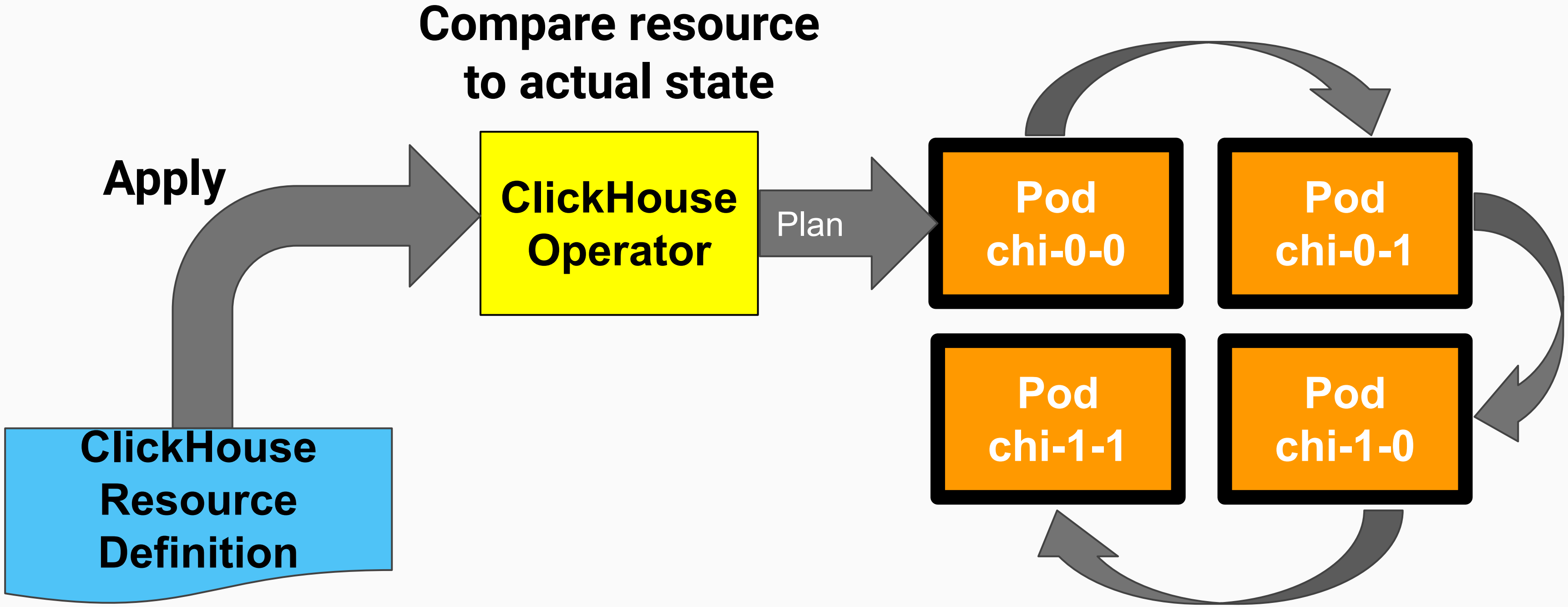
Stateful Set

🙁

Pods have varying capacity, affinity, version

Need to alter storage to add/change capacity

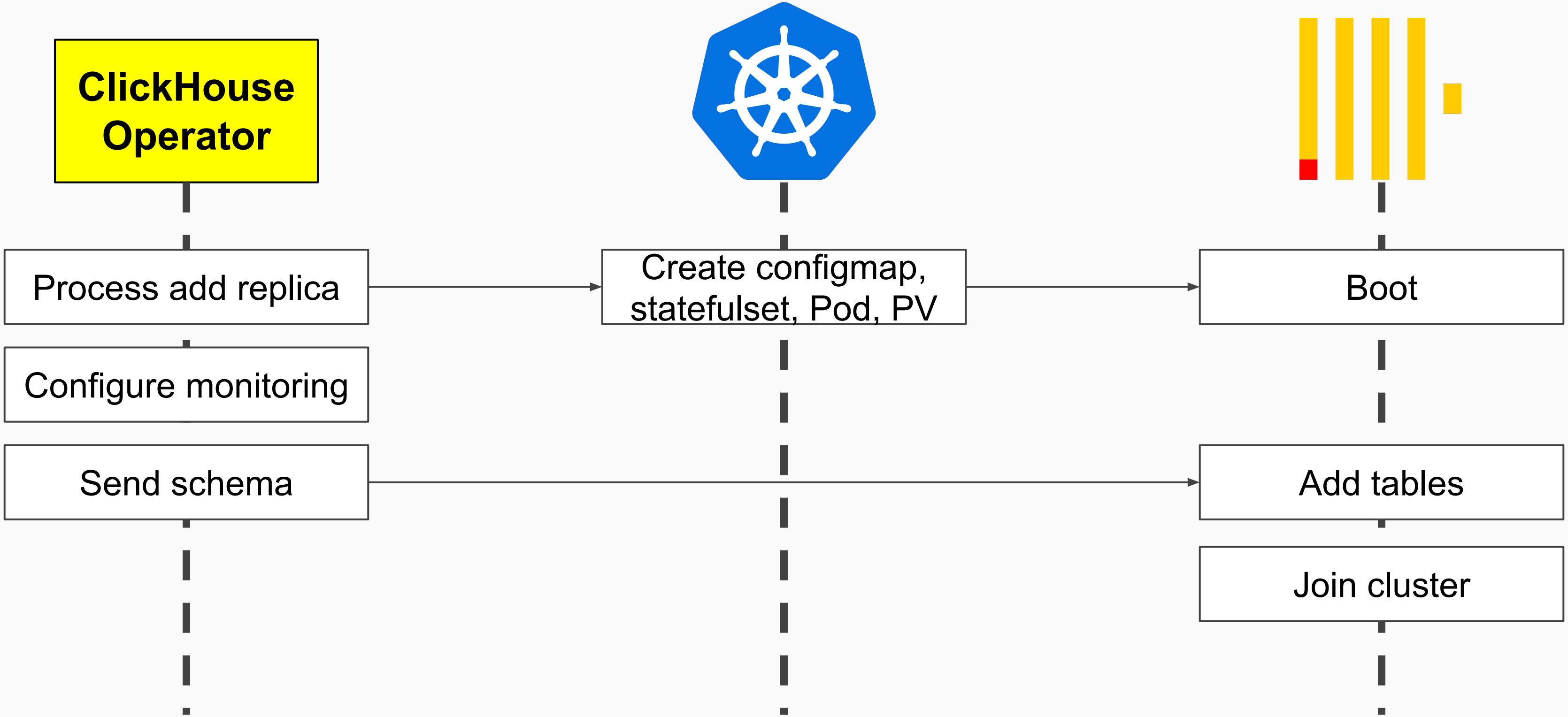
Analyzing state to perform upgrades



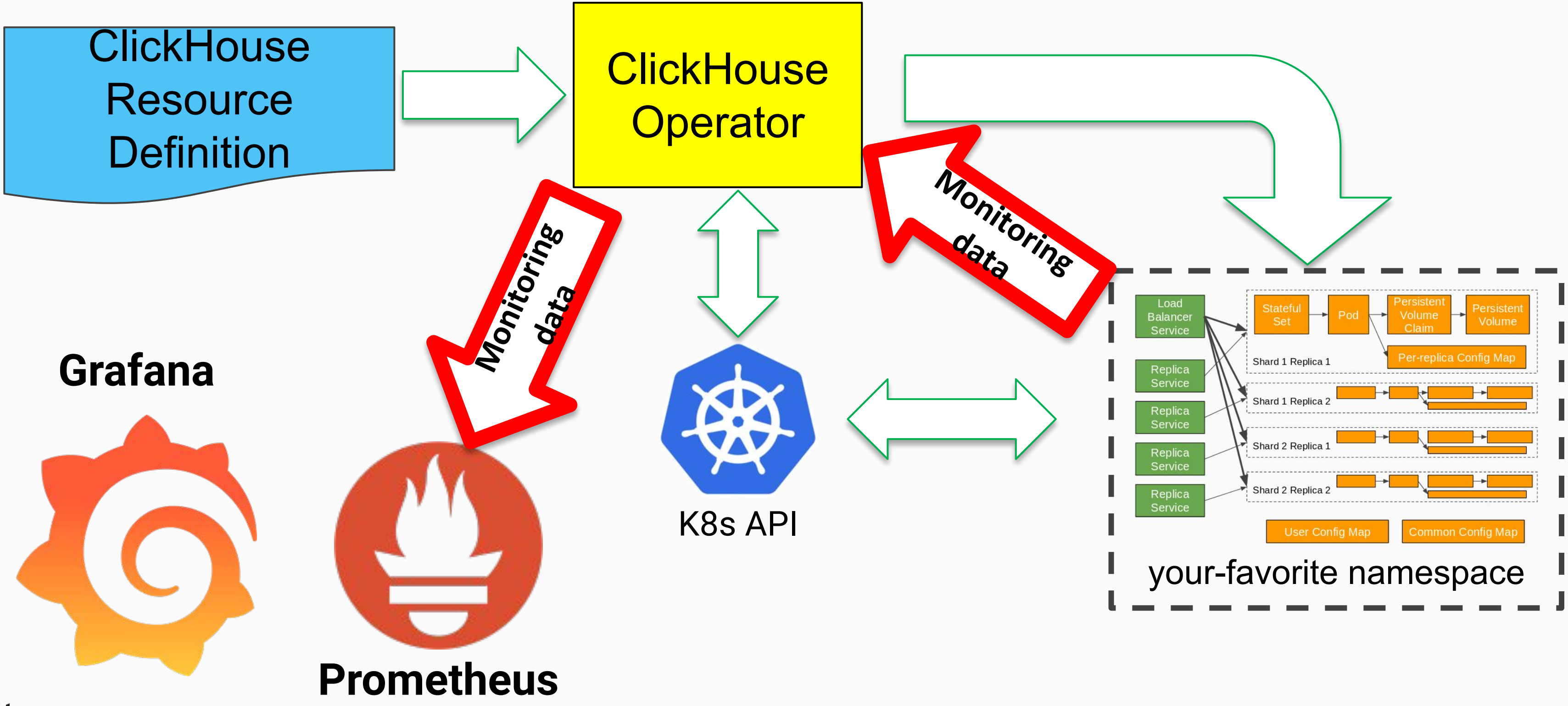
Update resource definition

Upgrade pods sequentially

Division of responsibilities -- adding a replica



Operator = Deployment + Operation + **Monitoring**



ClickHouse Resource Definition

ClickHouse Operator

Monitoring data

Monitoring data

Grafana

Prometheus

K8s API

Load Balancer Service

Stateful Set

Pod

Persistent Volume Claim

Persistent Volume

Per-replica Config Map

Shard 1 Replica 1

Replica Service

Shard 1 Replica 2

Replica Service

Shard 2 Replica 1

Replica Service

Shard 2 Replica 2

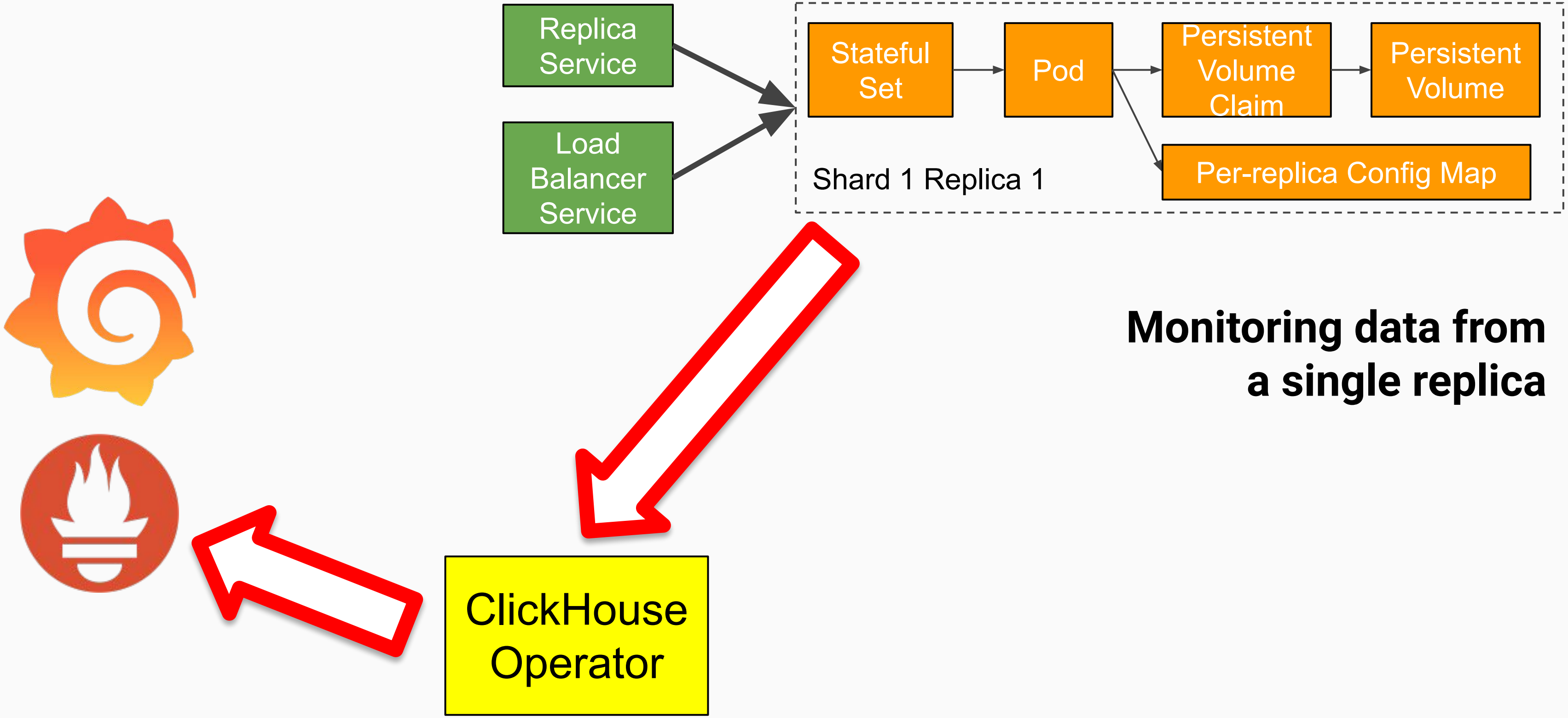
Replica Service

User Config Map

Common Config Map

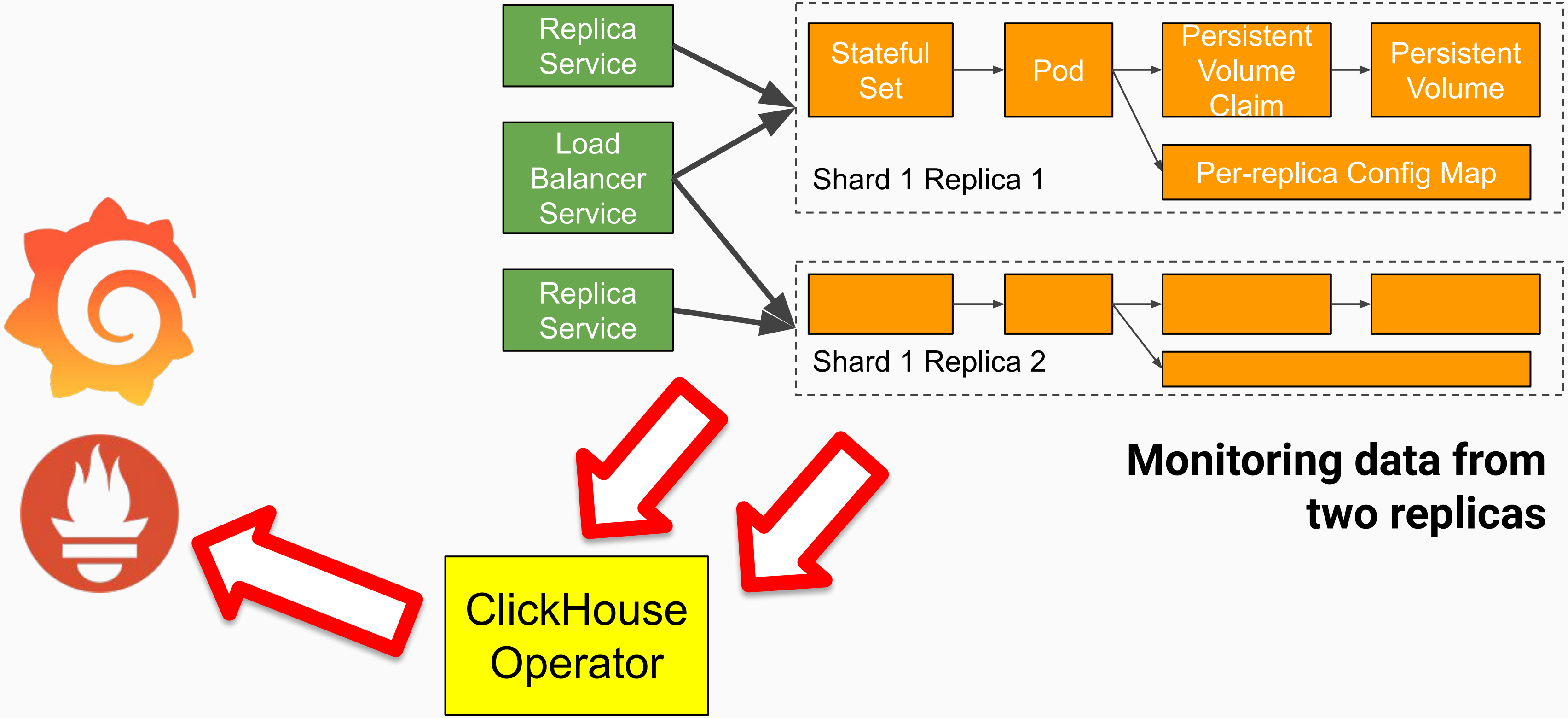
your-favorite namespace

System dynamism complicates monitoring



Monitoring data from a single replica

System dynamism complicates monitoring



Replica Service

Load Balancer Service

Replica Service

Stateful Set

Pod

Persistent Volume Claim

Persistent Volume

Per-replica Config Map

Shard 1 Replica 1

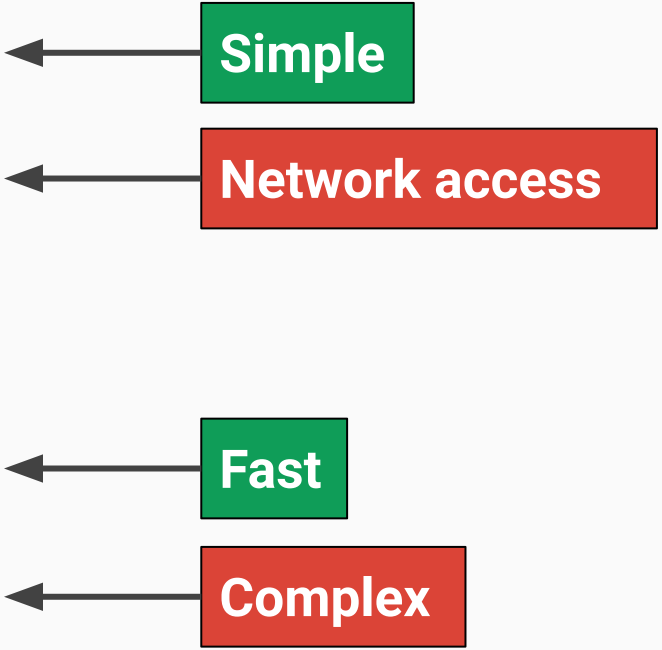
Shard 1 Replica 2

ClickHouse Operator

Monitoring data from two replicas

Speaking of storage, we have options

- Cloud storage:
  - AWS
  - GKE
  - Other cloud providers
- Local storage
  - emptyDir
  - hostPath
  - local



Simple

Network access

Fast

Complex

Use storageClassName to bind storage

Use kubectl to find available storage classes:

```
kubectl describe StorageClass
```

Bind to default storage:

```
spec:
  storageClassName: default
```

Bind to gp2 type

```
spec:
  storageClassName: gp2
```

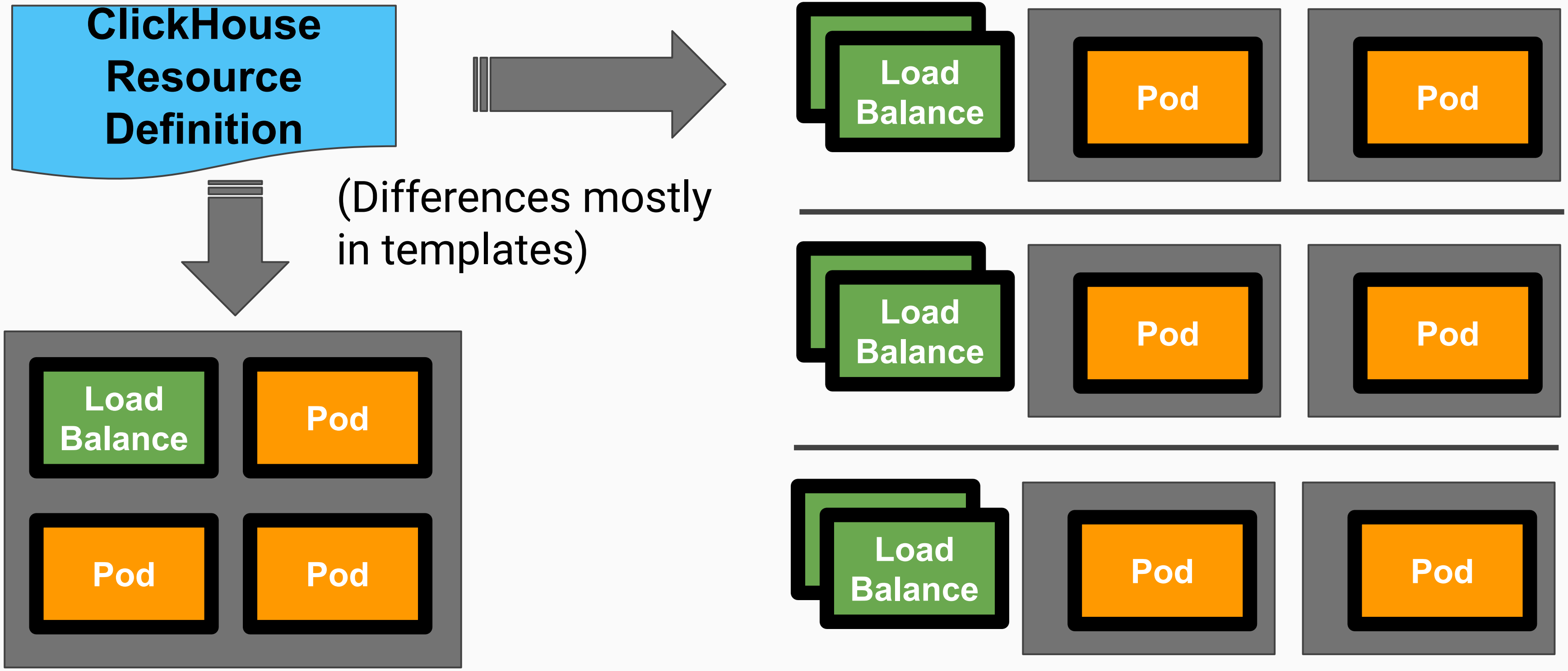
Explicit data placement is simplest for users

```
clusters:
  - name: zoned
     layout:
     shardsCount: 3
     replicas:
        - templates:
            podTemplate: clickhouse-in-zone-us-west-2a
        - templates:
            podTemplate: clickhouse-in-zone-us-west-2b
```

Replicas use different pod templates

Templates encapsulate affinity to specific zone

Templates also enable portability



ClickHouse Resource Definition

(Differences mostly in templates)

Load Balance

Pod

Pod

Load Balance

Pod

Pod

Load Balance

Pod

Pod

Load Balance

Pod

Pod

Pod

Minikube

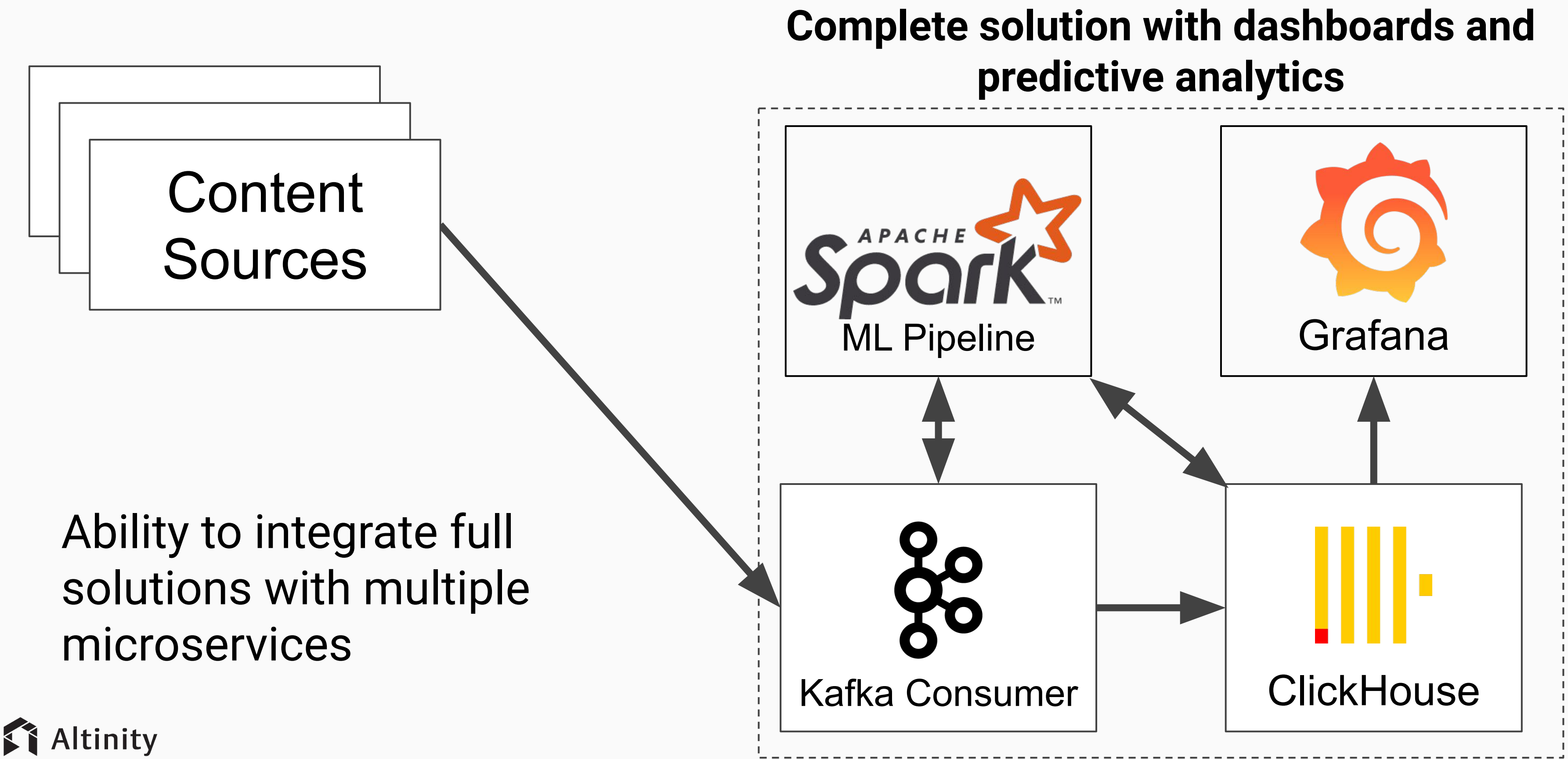
Multi-AZ Deployment

ClickHouse operator roadmap

- Backup and restore
- Reclaim storage in new clusters
- Services
  - Per node access
  - External vs. internal endpoints
- Security
  - Certificate management across nodes
  - Encrypted storage

Using Kubernetes to Build Analytic Solutions

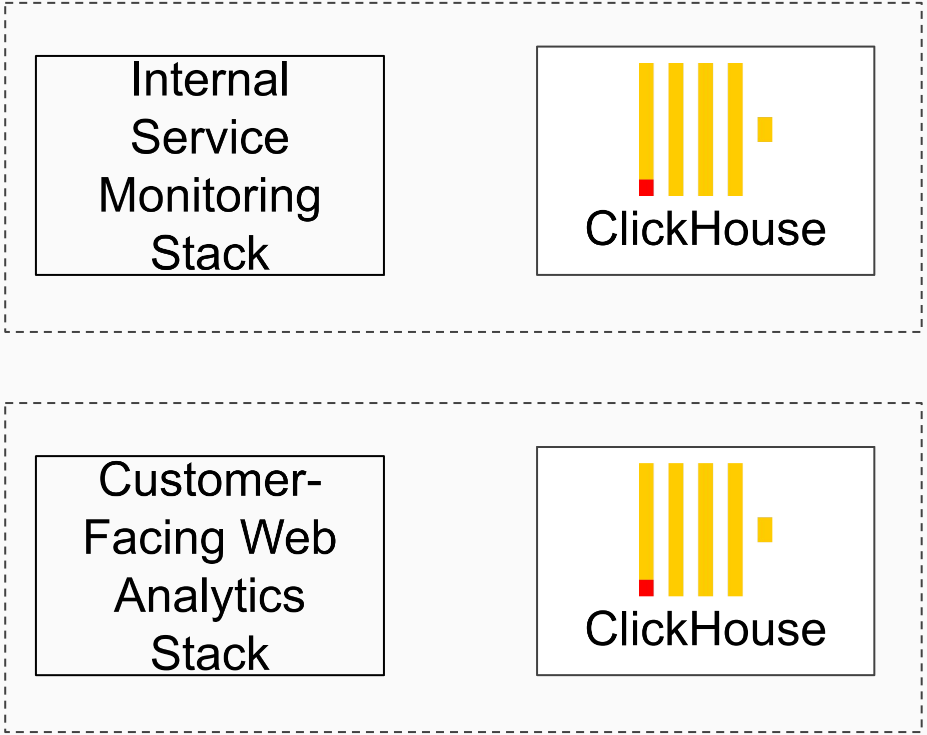
Kubernetes benefit #1



Ability to integrate full solutions with multiple microservices

Kubernetes benefit #2

Independently scaled analytics backend for different services


Internal Service Monitoring Stack
ClickHouse
Customer-Facing Web Analytics Stack
ClickHouse

Separate sharding, replication, storage, etc.

Biggest long-range opportunity

Kubernetes democratizes data warehouse access

Operators enable effective data management

ClickHouse Kubernetes operator enables any application to add high-performance analytics

Thank you!

Questions?

Presenters:

rhodges@altinity.com
vladislav@altinity.com

ClickHouse Operator:

https://github.com/Altinity/clickhouse-operator

ClickHouse:

https://github.com/yandex/ClickHouse

Altinity:

https://www.altinity.com