Cloud Native ClickHouse at Scale

Using the Altinity Kubernetes Operator

Robert Hodges and Altinity Engineering

Let's make some introductions

Robert Hodges

Database geek with 30+ years on DBMS systems. Day job: Altinity CEO

Altinity Engineering

Database geeks with centuries of experience in DBMS and applications

ClickHouse support and services including Altinity.Cloud
Authors of Altinity Kubernetes Operator for ClickHouse
and other open source projects

ClickHouse is a real-time analytic database

Understands SQL

Runs on bare metal to cloud

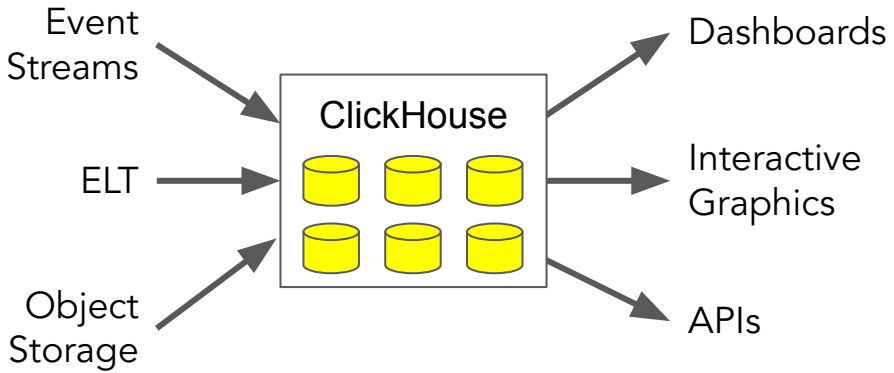
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

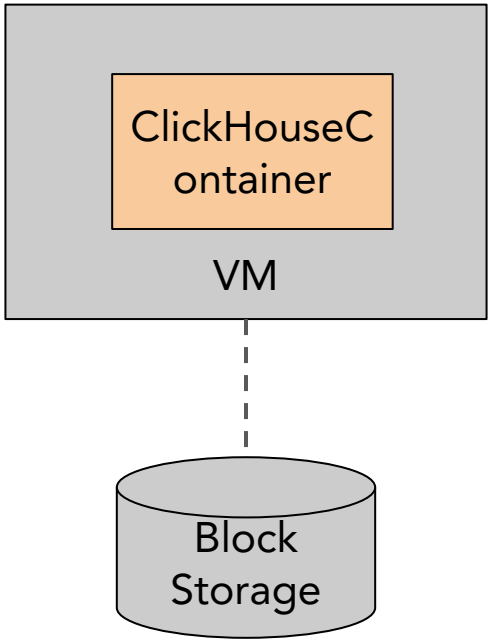
Is Open source (Apache 2.0)



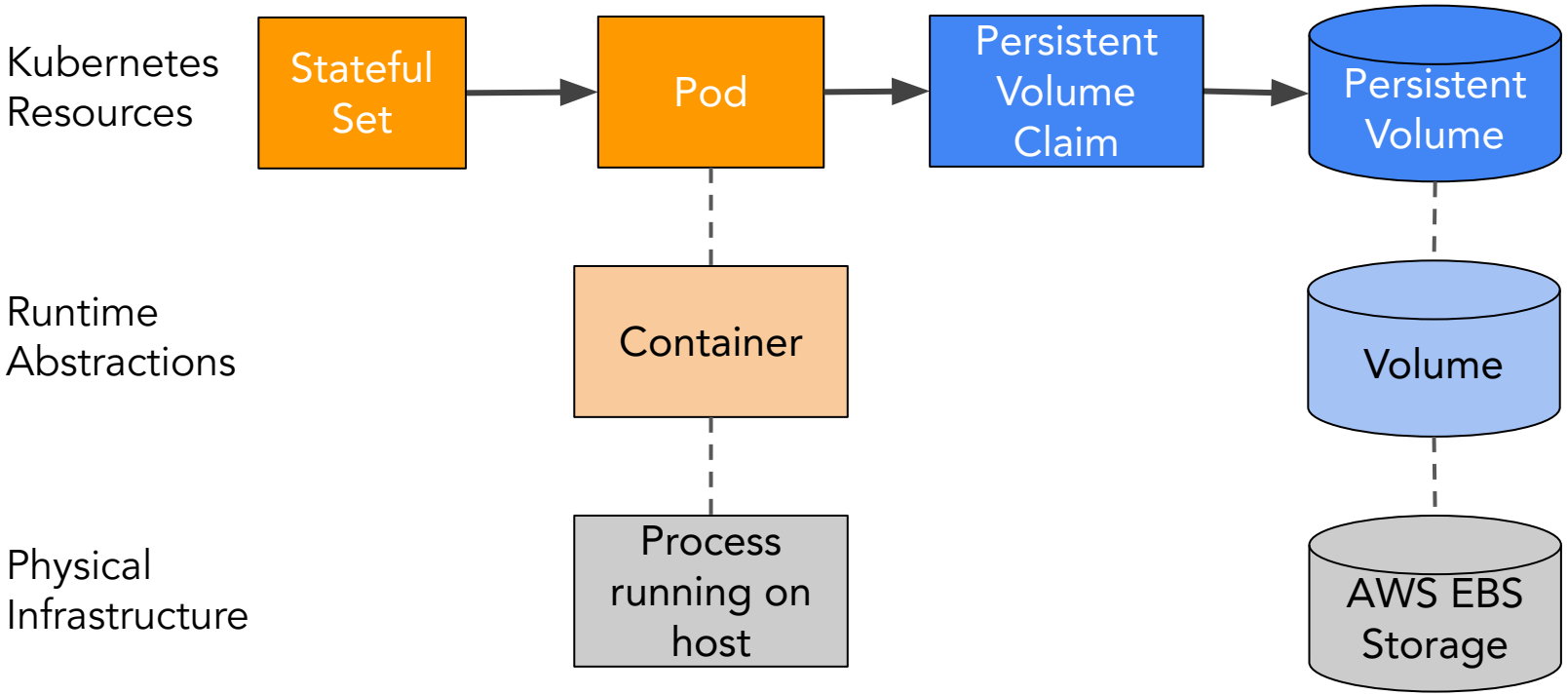
It's a popular engine for real-time analytics

Kubernetes, Operators, and ClickHouse

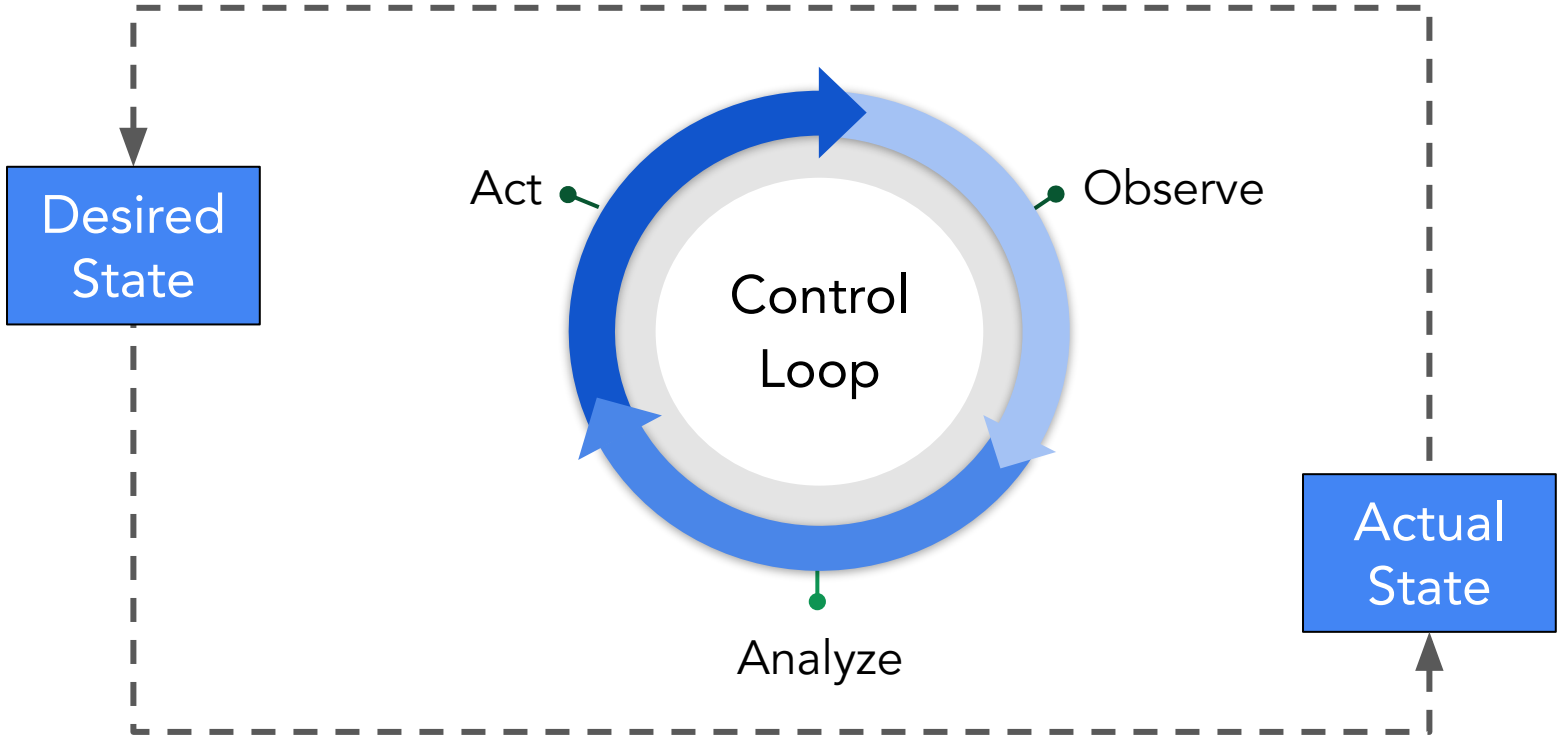
Kubernetes manages container-based applications



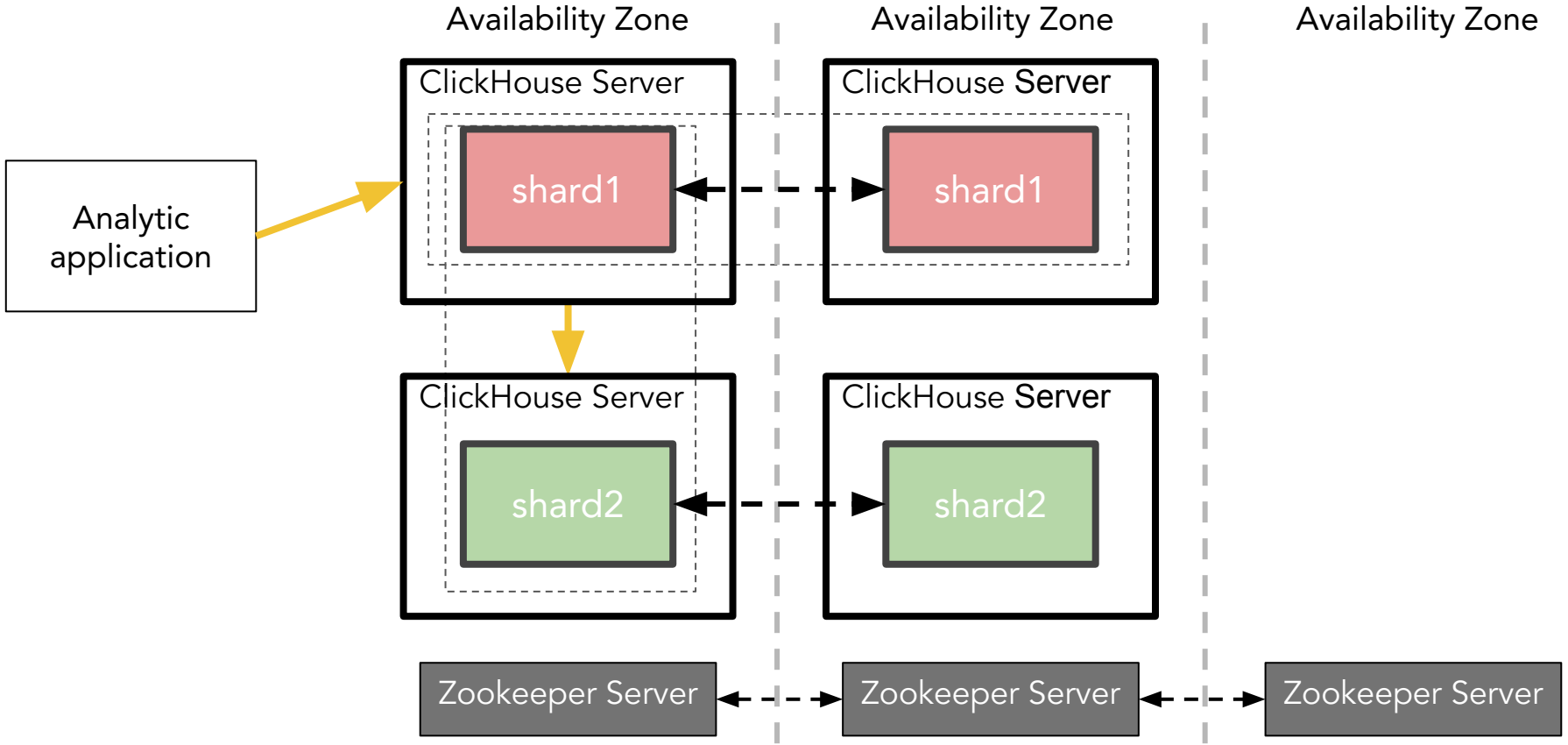
Kubernetes maps resource definitions to infrastructure



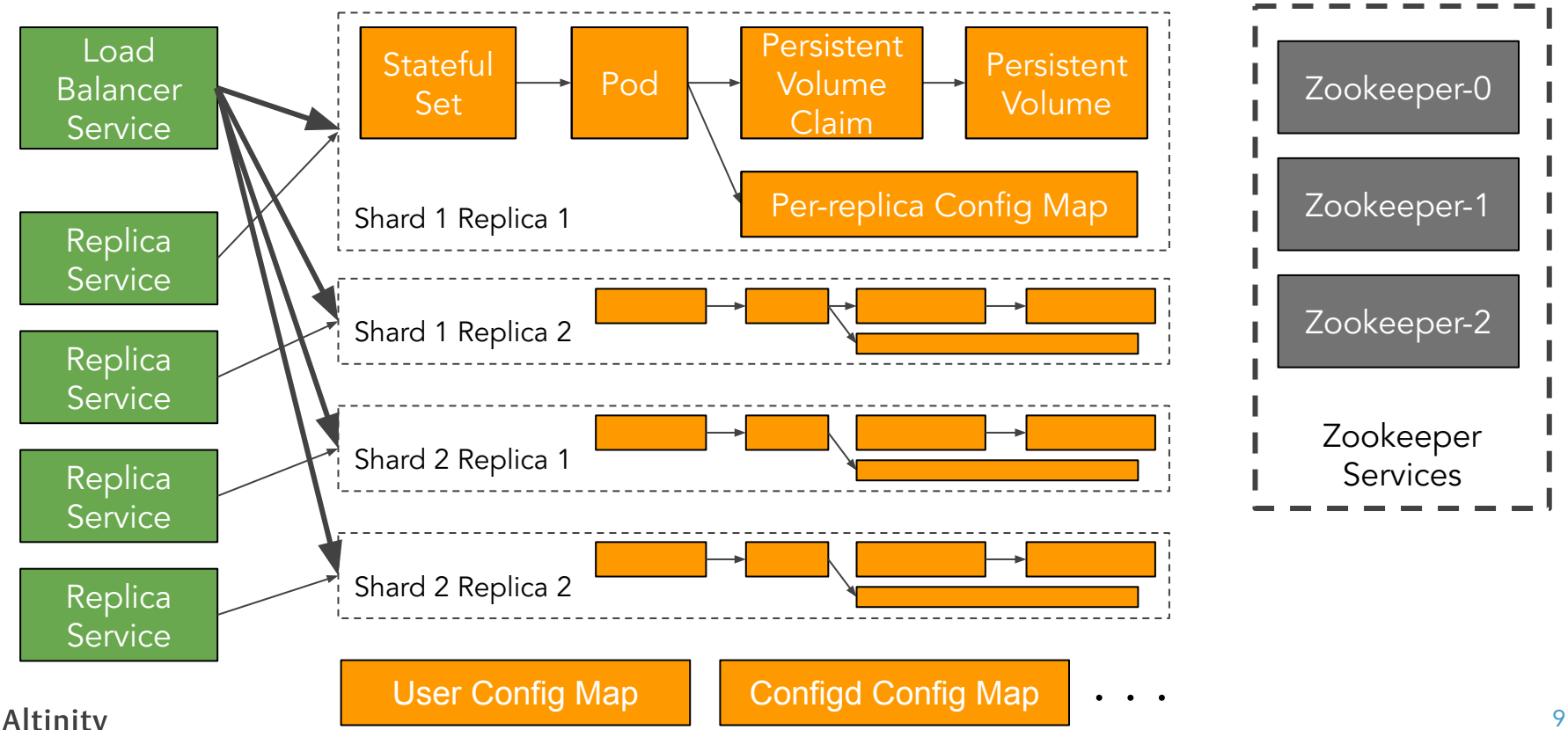
General control loop for Kubernetes resources



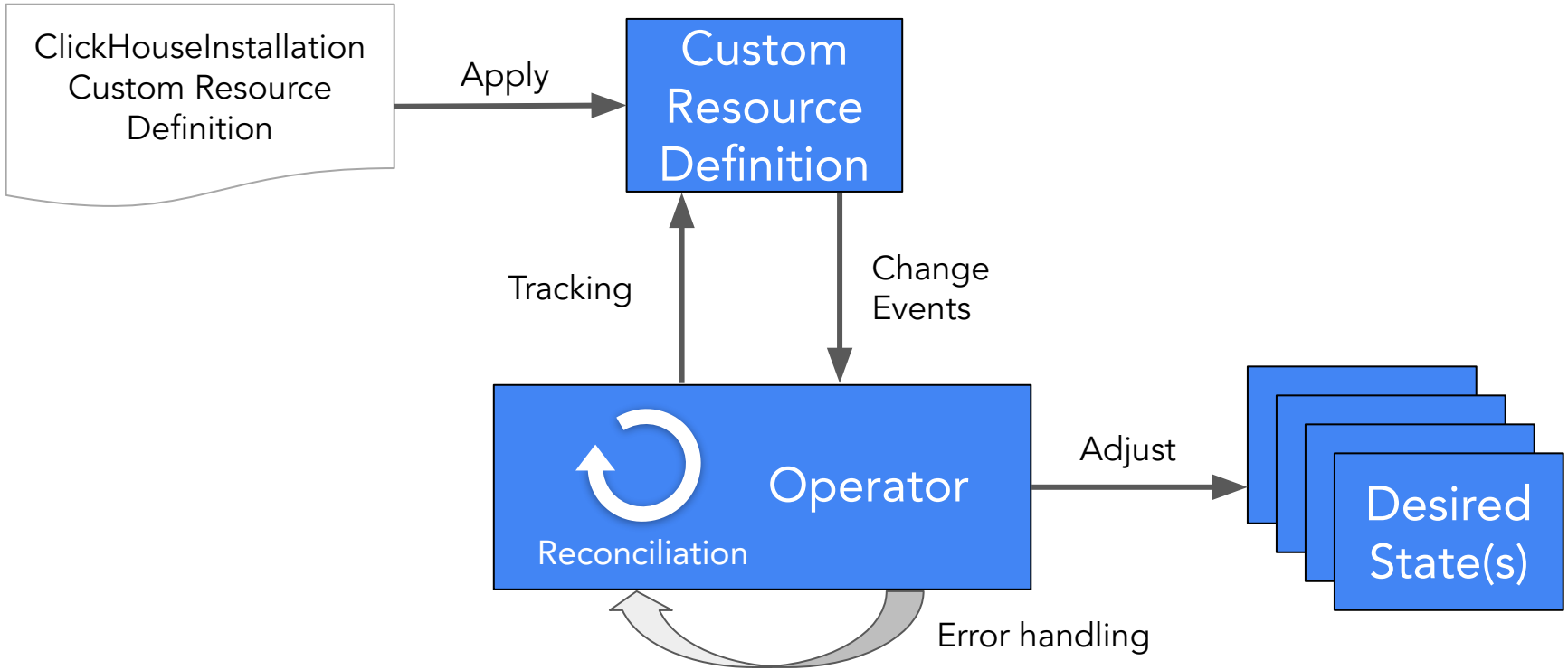
ClickHouse is usually a little more complicated!



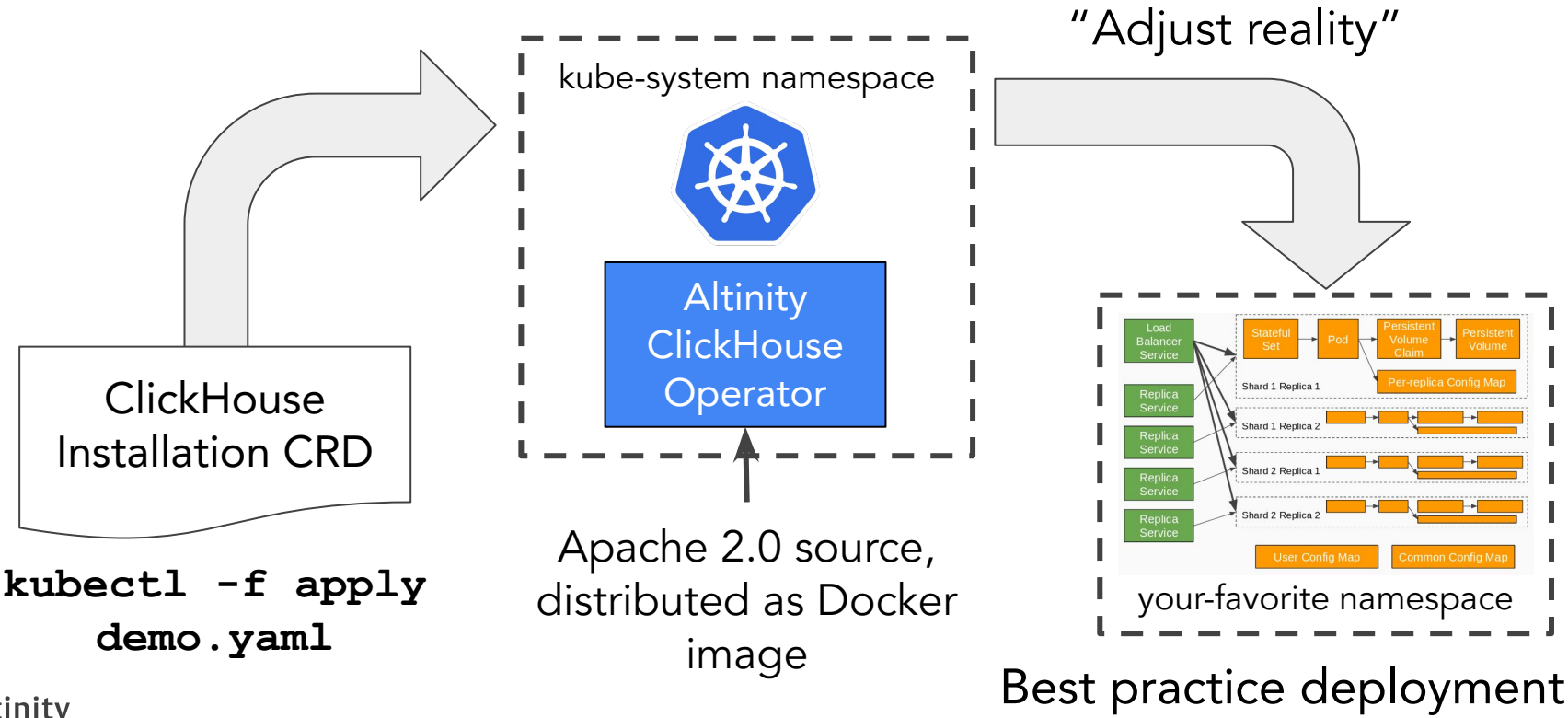
…Which means lots of Kubernetes resources



Operators implement and manage “custom resources”



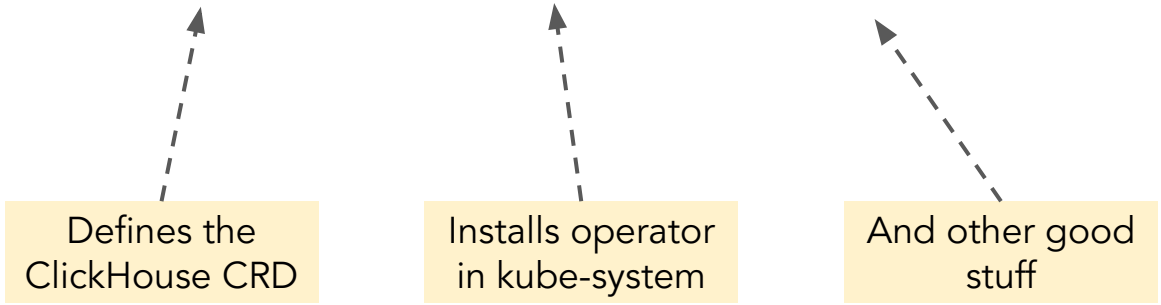
Result: Operators make databases work on Kubernetes



Setting up your first ClickHouse cluster in Kubernetes

Step 1: Install ClickHouse operator from GitHub

```
kubectl apply -f \
https://raw.githubusercontent.com/Altinity/clickhouse-operator/master/dep
loy/operator/clickhouse-operator-install-bundle.yaml
```



Step 2: Set up ZooKeeper

Get Zookeeper stateful set definition:

```
wget \
https://raw.githubusercontent.com/Altinity/clickhouse-operator/master/dep
loy/zookeeper/quick-start-persistent-volume/zookeeper-1-node.yaml
```

Install Zookeeper.

```
kubectl create ns zoo1ns
kubectl apply -f zookeeper-1-node.yaml -n zoo1ns
```

Dev only

Step 3: Define your cluster (cluster configuration)

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "demo"
spec:
  configuration:
    clusters:
      - name: "cl"
        layout:
          shardsCount: 1
          replicasCount: 2
        templates:
          podTemplate: server
          volumeClaimTemplate: storage
    zookeeper:
        nodes:
        - host: zookeeper.zoo1ns
          port: 2181
```

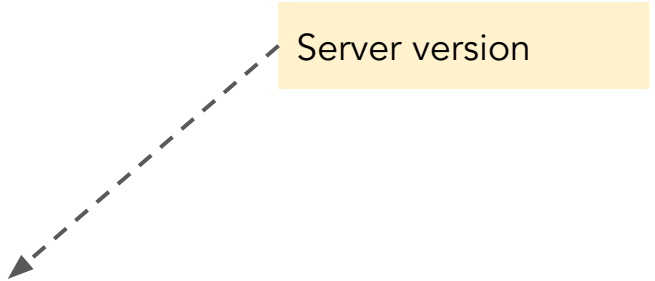
Shards and replicas

Definitions for pods and storage

Where is Zookeeper?

Step 3: Define your cluster (pod definition)

```
templates:
  podTemplates:
    - name: server
      spec:
        containers:
        - name: clickhouse
          image: altinity/clickhouse-server:22.3.15.34.altinitystable
```



Step 3: Define your cluster (pod definition)

```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC if installation is dropped.
    reclaimPolicy: Retain
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

Protect storage from deletion

Storage size

DEMO TIME!

Accessing our creations

Safety first...Check storage!

```
$ kubectl get pvc; kubectl get pv
```

Access ClickHouse

```
$ kubectl exec -it pod/chi-demo-cl-0-0-0 -- clickhouse-client
```

Forward port to external network.

```
$ kubectl port-forward service/clickhouse-demo 8123 > /dev/null &
$ curl http://localhost:8123
Ok.
```

Adding a user

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
. . .
spec:
  configuration:
    clusters: . . .
    zookeepers: . . .
    users:
      root/password_sha256_hex: 2bb80……7a25b
      root/networks/ip:
        - ::1
        - 127.0.0.1
      root/quota: default
      root/access_management: 1
```

User definition

Enable RBAC

Scaling up to production on Kubernetes

Choosing a Kubernetes distribution

Amazon EKS

Google Kubernetes Engine

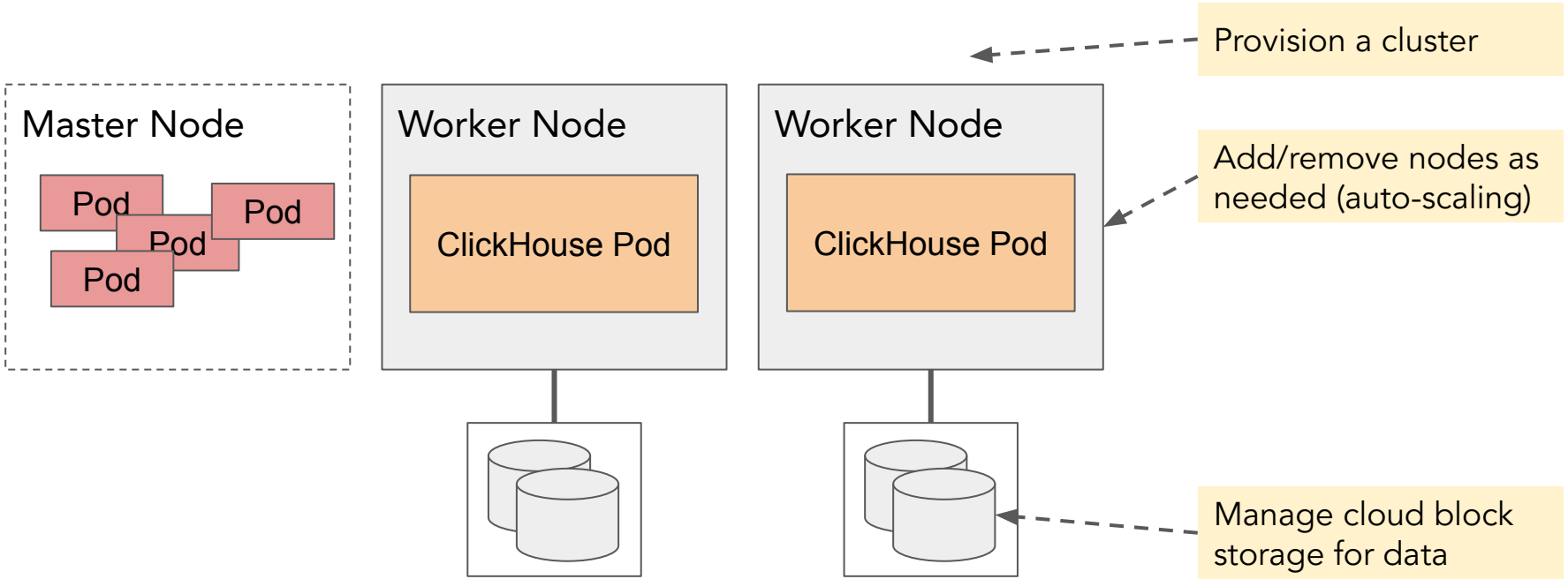
DigitalOcean

minikube*

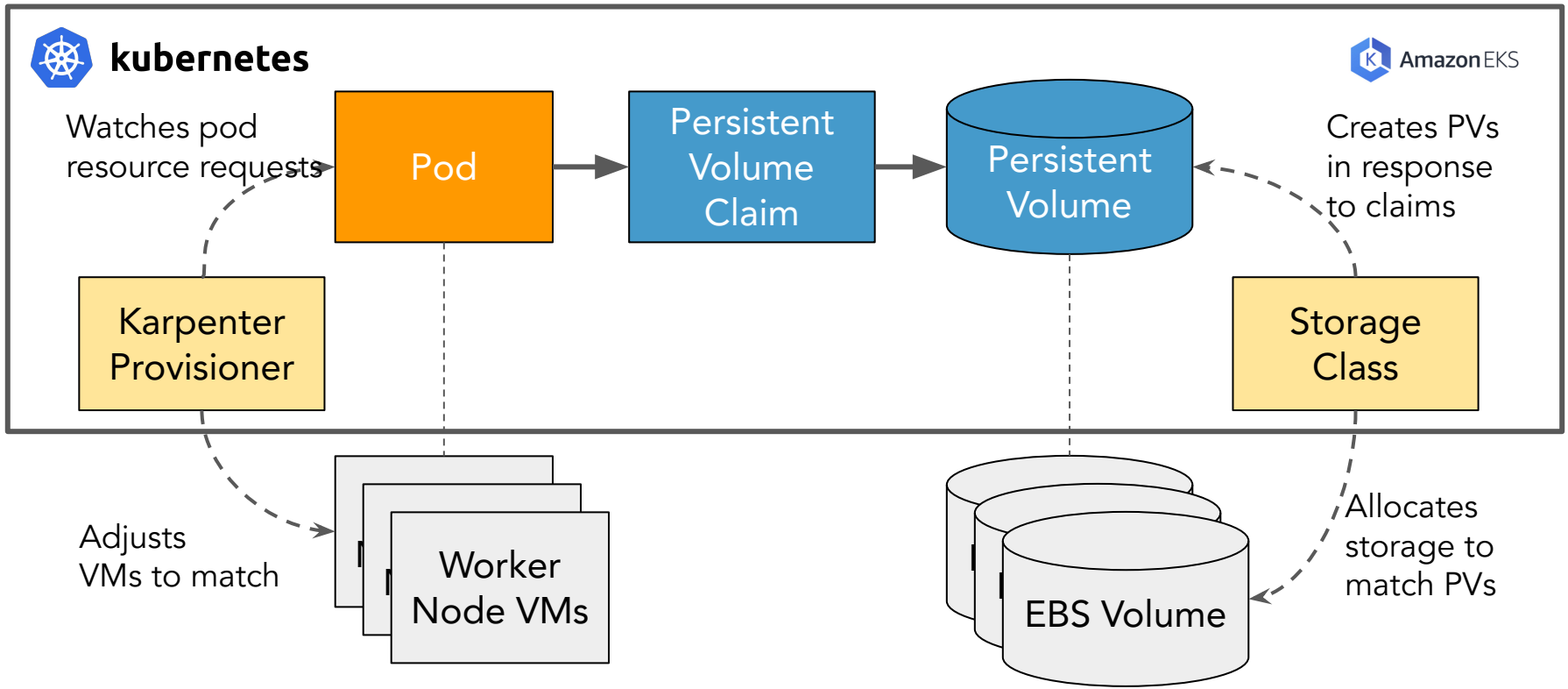
AZURE KUBERNETES SERVICE

* Used for demo/test only

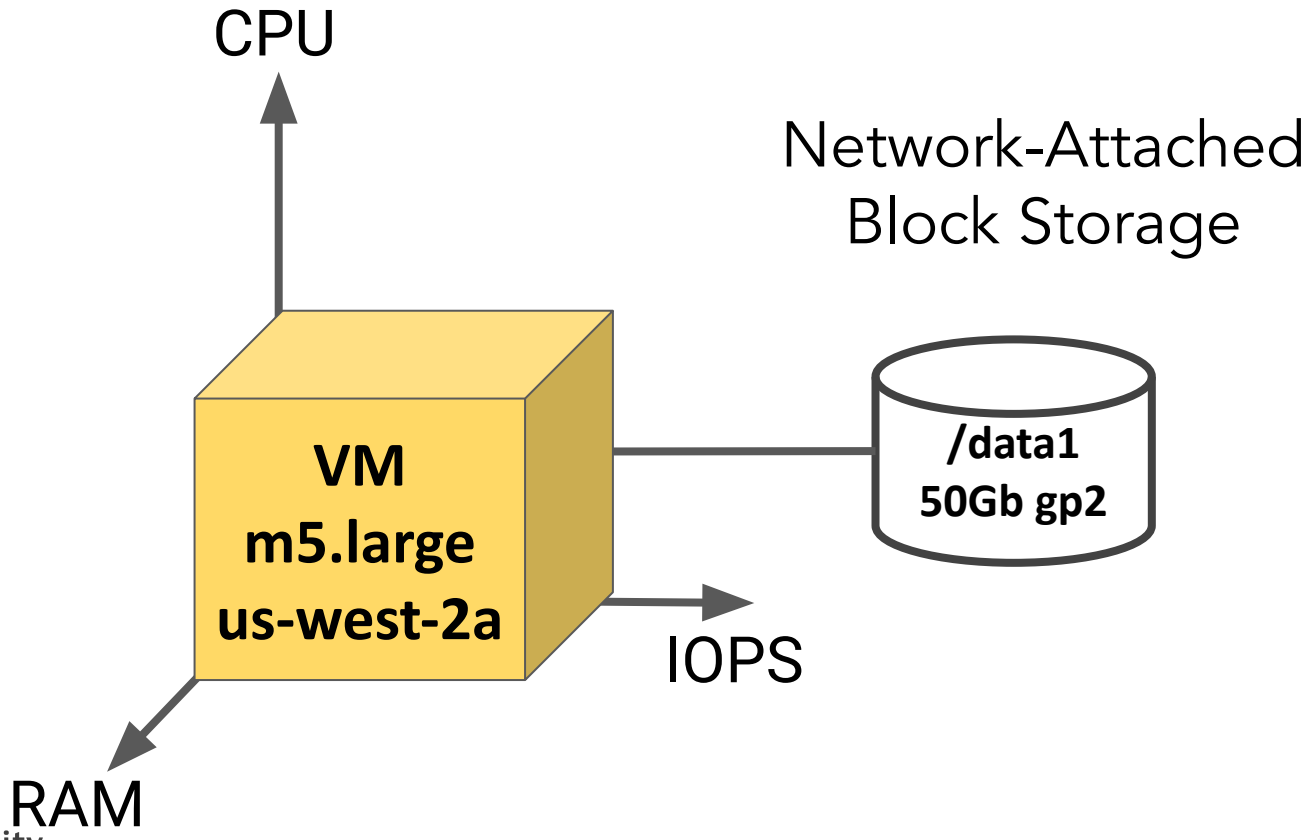
Your Kubernetes setup needs to get a few things right



…Provided you have the right magic configured

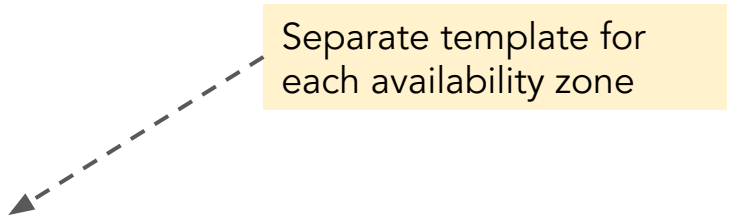


We can control VM type, zone, and storage type



Use pod templates to place replicas in different zones

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"metadata:
  name: "prod"
spec:
  configuration:
    clusters:
      - name: "ch"
        layout:
          replicas:
          - templates:
              podTemplate: clickhouse-zone-2a
          - templates:
              podTemplate: clickhouse-zone-2b
          shardsCount: 1
        templates:
          volumeClaimTemplate: storage
```



Node selectors and instance types force pods to nodes

```
podTemplates:
 - name: clickhouse-zone-2a
   spec:
     containers:
     - name: clickhouse
       image: altinity/clickhouse-server:22.3.15.34.altinitystable
       resources:
         limits:
           cpu: 1800m
           memory: 7Gi
         requests:
           cpu: "1"
           memory: 6452Mi
     nodeSelector:
       node.kubernetes.io/instance-type: m5.large
   zone:
     key: topology.kubernetes.io/zone
     values:
     - us-west-2a
```

Reserves resources on VM

Requires a node with m5.large VM type

Requires a node in zone us-west-2a

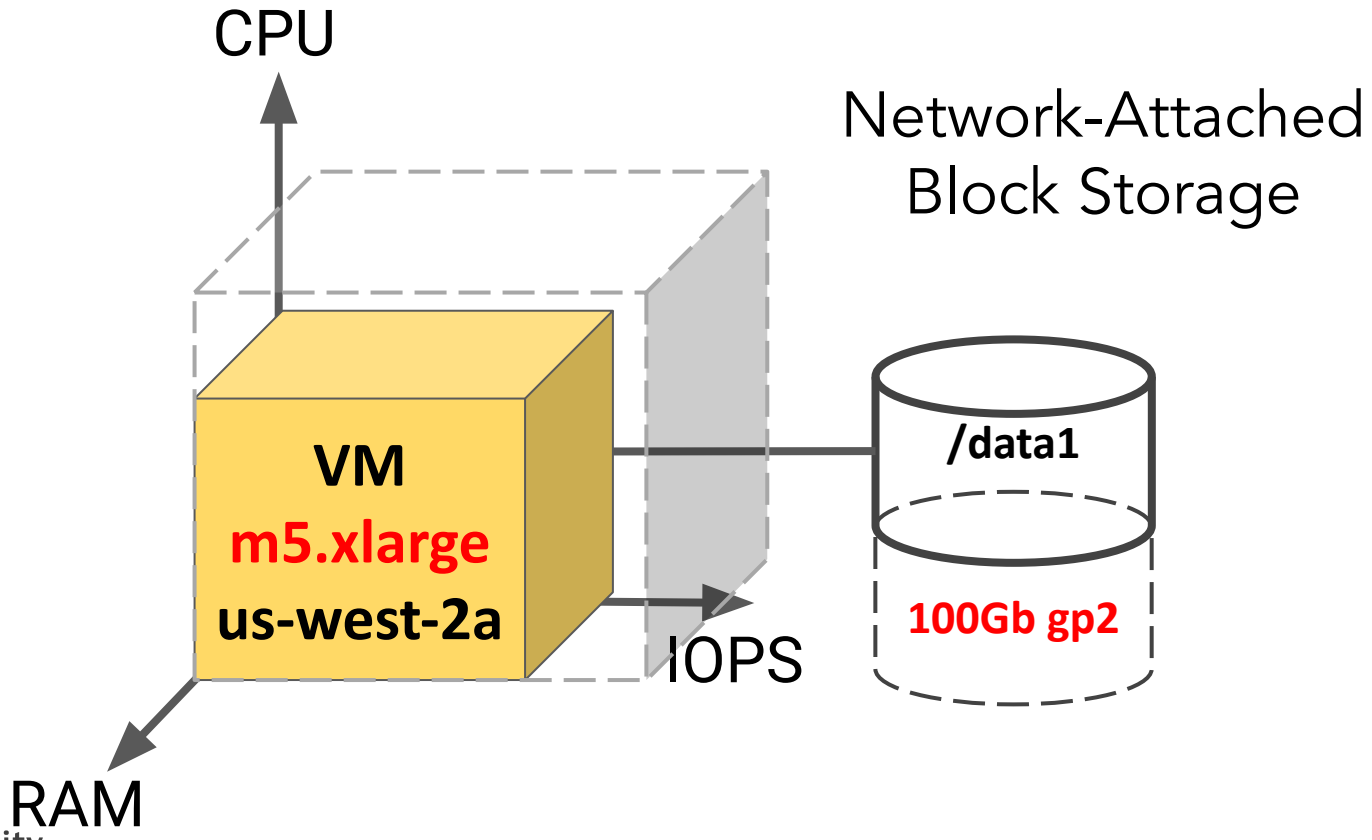
Volume claim templates allocate storage for pods

```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC if installation is dropped.
    reclaimPolicy: Retain
    spec:
      storageClassName: gp2
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

Set up storage classes for the storage types that you want

Amount of storage requested

Let's scale up vertically!



Scale pods using nodeSelector and resources

```
podTemplates:
 - name: clickhouse-zone-2a
   spec:
     containers:
     - name: clickhouse
       image: altinity/clickhouse-server:22.3.15.34.altinitystable
       resources:
         limits:
           cpu: 3600m
           memory: 15Gi
         requests:
           cpu: "2"
           memory: 13Gi
     nodeSelector:
       node.kubernetes.io/instance-type: m5.xlarge
   zone:
     key: topology.kubernetes.io/zone
     values:
     - us-west-2a
```

Reserves resources on VM; max the values to reserve it for ClickHouse

Requires an m5.xlarge now

Storage templates define persistent volume claims (PVCs)

```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC if installation is dropped.
    reclaimPolicy: Retain
    spec:
      storageClassName: gp2
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 100Gi
```

Will use default storage class if this is omitted

You can increase storage but cannot decrease it

Future versions of the operator will extend storage without requiring a restart

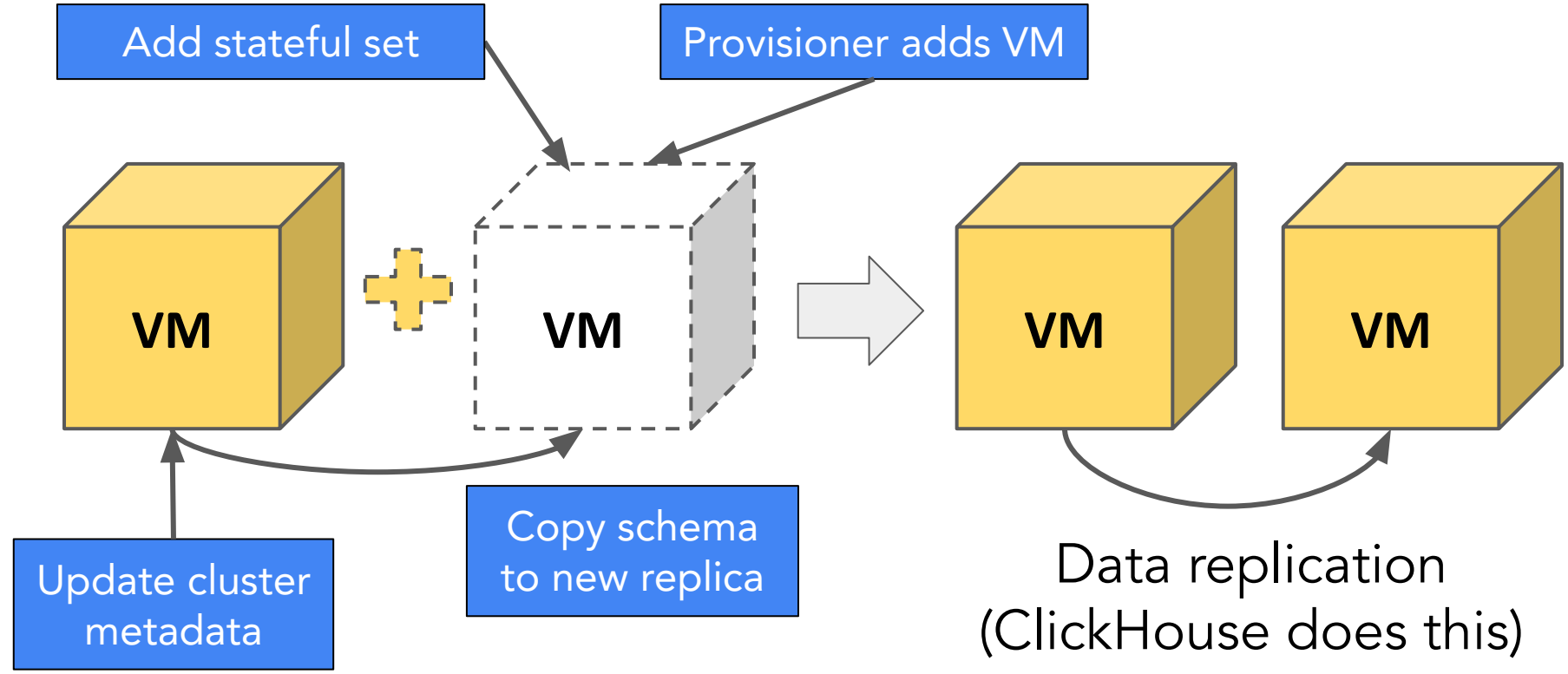
Know your storage class(es)!

```
$ kubectl get storageclass gp2 -o yaml
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  storageclass.kubernetes.io/is-default-class: "true"
  name: gp2
parameters:
  fsType: ext4
  type: gp2
provisioner: ebs.csi.aws.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Make sure your storage class supports volume expansion

This storage class will be picked if you don't specify anything else

How does the operator manage adding replicas?



Safety tips for happy Kubernetes operation

Tip 1: Never run this command while there are active ClickHouse clusters. It deletes the ClickHouseInstallation CRD definition. Kubernetes will then delete your clusters.

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

Tip 2: Use the "reclaimPolicy: Retain" setting to keep storage from being deleted if you accidentally delete a cluster

Tip 3: Move data off shards before deactivating them. The Altinity Operator will not do it automatically.

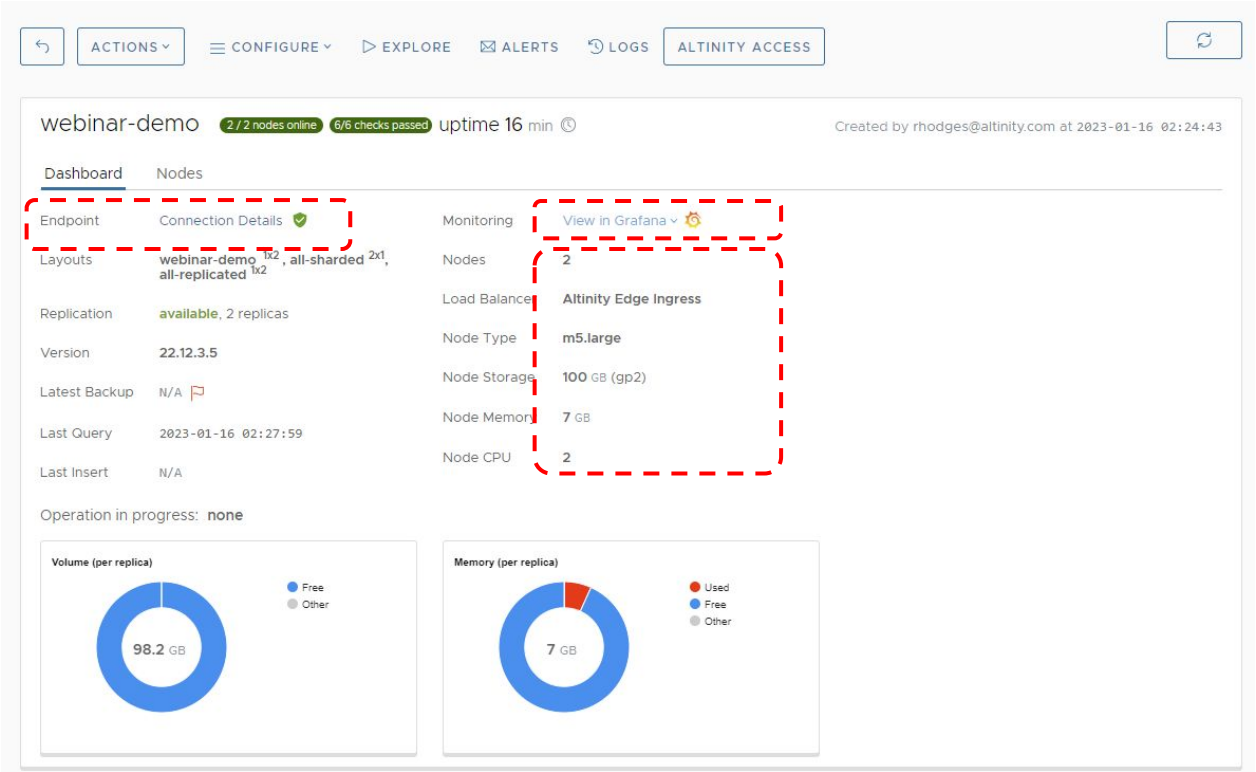
What's next?

More things to learn about

- External network access
  - Check out service deployment examples in https://github.com/Altinity/clickhouse-operator
  - Altinity Operator can configure external and internal load balancers
- Backup
  - We use clickhouse-backup running as a sidecar to ClickHouse
- ZooKeeper 3 node ensemble
  - See setup recommendations on GitHub
- Security
  - Check out the write-up in the Altinity Operator hardening guide.
- Monitoring and alerting

Typical Day 2 monitoring from Altinity.Cloud

Looking for an easier way? Check out Altinity.Cloud.



Final thoughts

How to get started with ClickHouse on Kubernetes

- The Altinity Kubernetes Operator manages ClickHouse clusters
- Try it out on Minikube or other dev versions of ClickHouse
- Most people use managed Kubernetes services for production systems
  - But OpenShift, Rancher, KOPS are OK too…
- Map ClickHouse nodes to VMs using a provisioner or node groups
- Check the docs for advanced topics

More information!

- Altinity Kubernetes Operator for ClickHouse on GitHub
  - https://github.com/Altinity/clickhouse-operator
- Altinity documention (https://docs.altinity.com)
- Altinity blog (https://altinity.com/blog)
- Kubernetes docs (https://kubernetes.io/docs/home/)
- EKS and GKE documentation
  - Including eksctl
- Karpenter documentation (https://karpenter.sh/)

Help make the operator better!!!

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

Tell your friends!

Log issues!

Send us pull requests!

Thank you!
Questions?

https://altinity.com
rhodges at altinity.com
alz at altinity.com

Altinity.Cloud

Altinity Kubernetes Operator for ClickHouse

We're hiring!