Adventures in Data

Leaning on Kubernetes Storage to Run Hundreds of Real-Time Analytic Databases

Robert Hodges, Altinity
KubeCon 2023

Let's make some introductions

Robert Hodges

Database geek with 30+ years on DBMS. Kubernaut since 2018. 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

Introducing Analytic Databases

ClickHouse is a SQL Data Warehouse

Understands SQL

Runs on bare metal to cloud

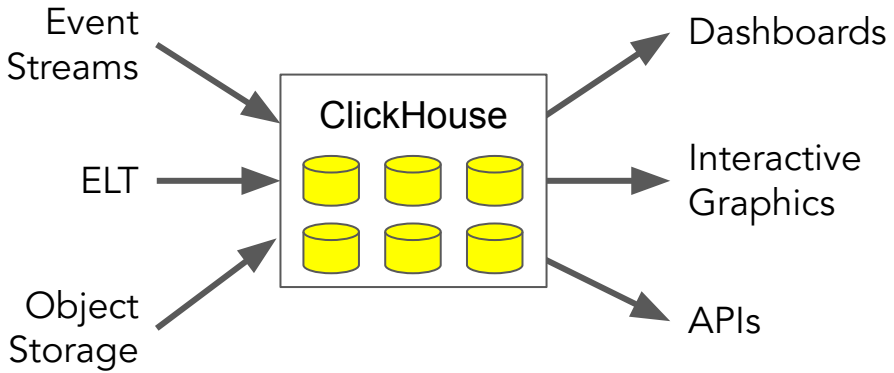
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

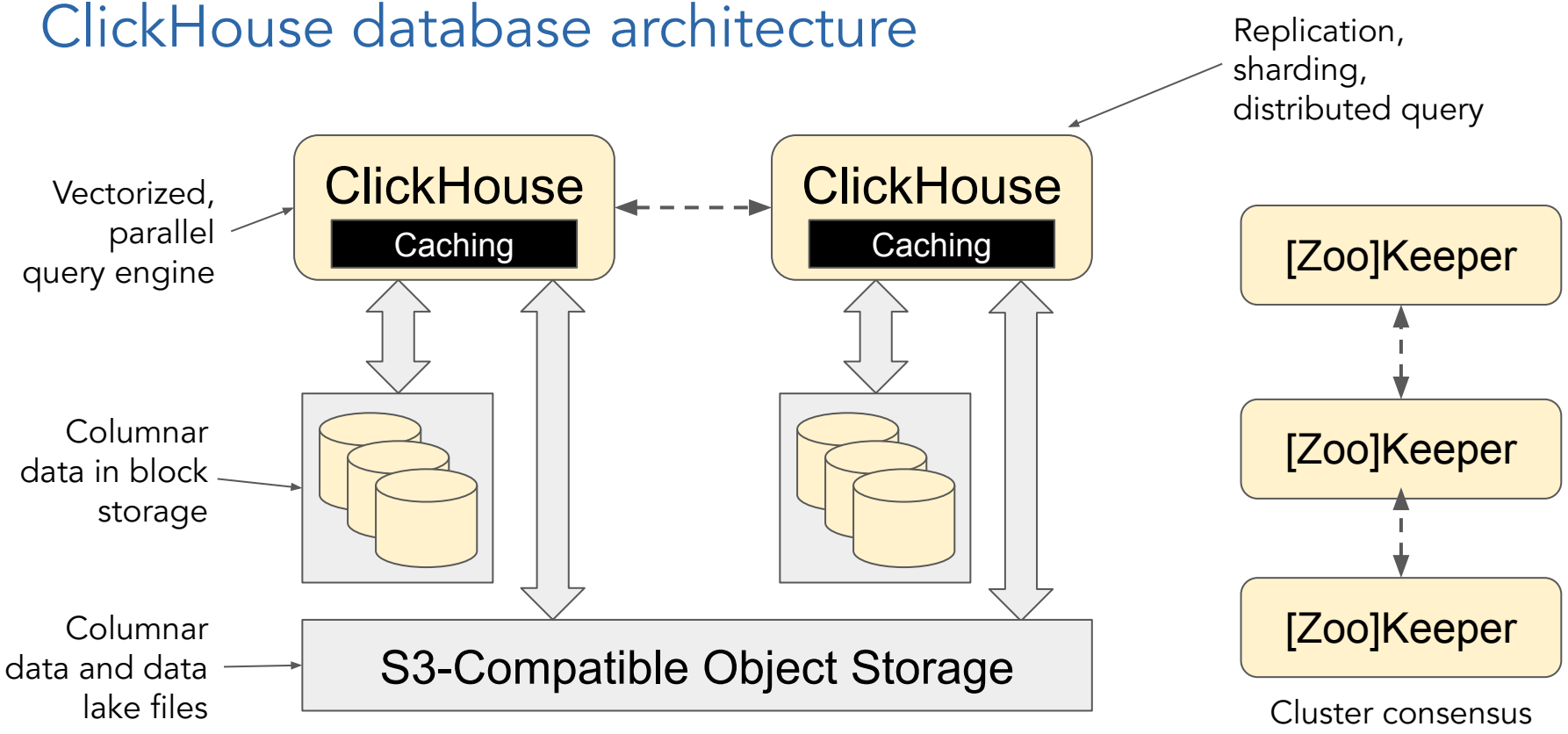
Scales to many petabytes

Is Open source (Apache 2.0)



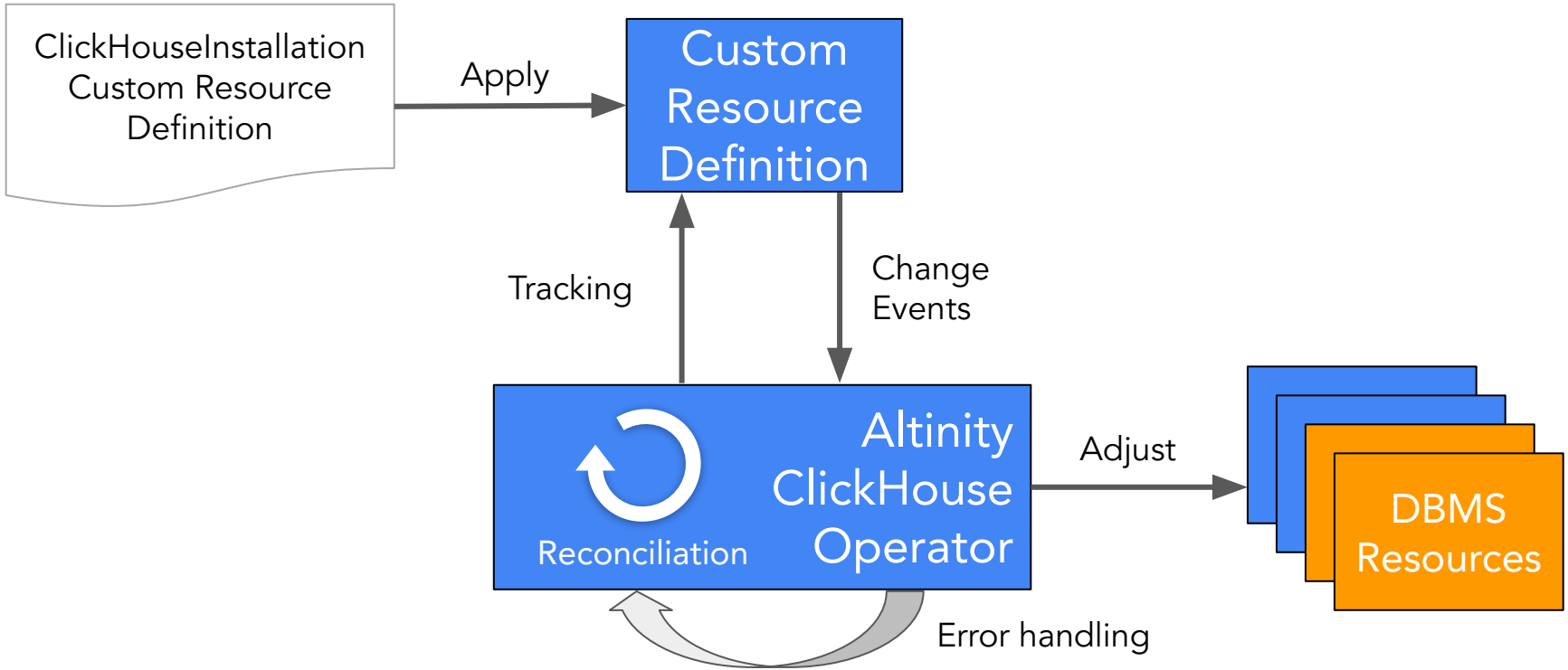
It's a popular engine for real-time analytics

ClickHouse database architecture

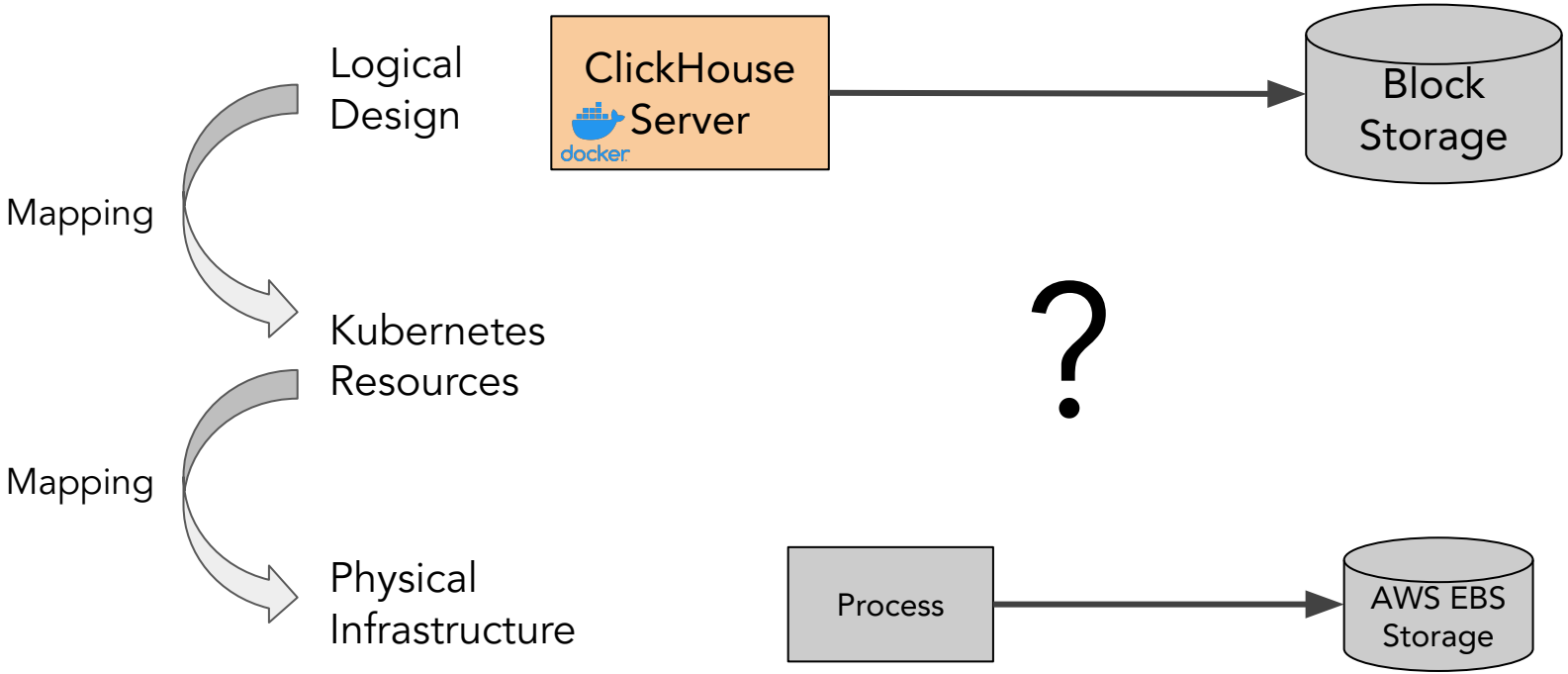


Mapping the database to Kubernetes

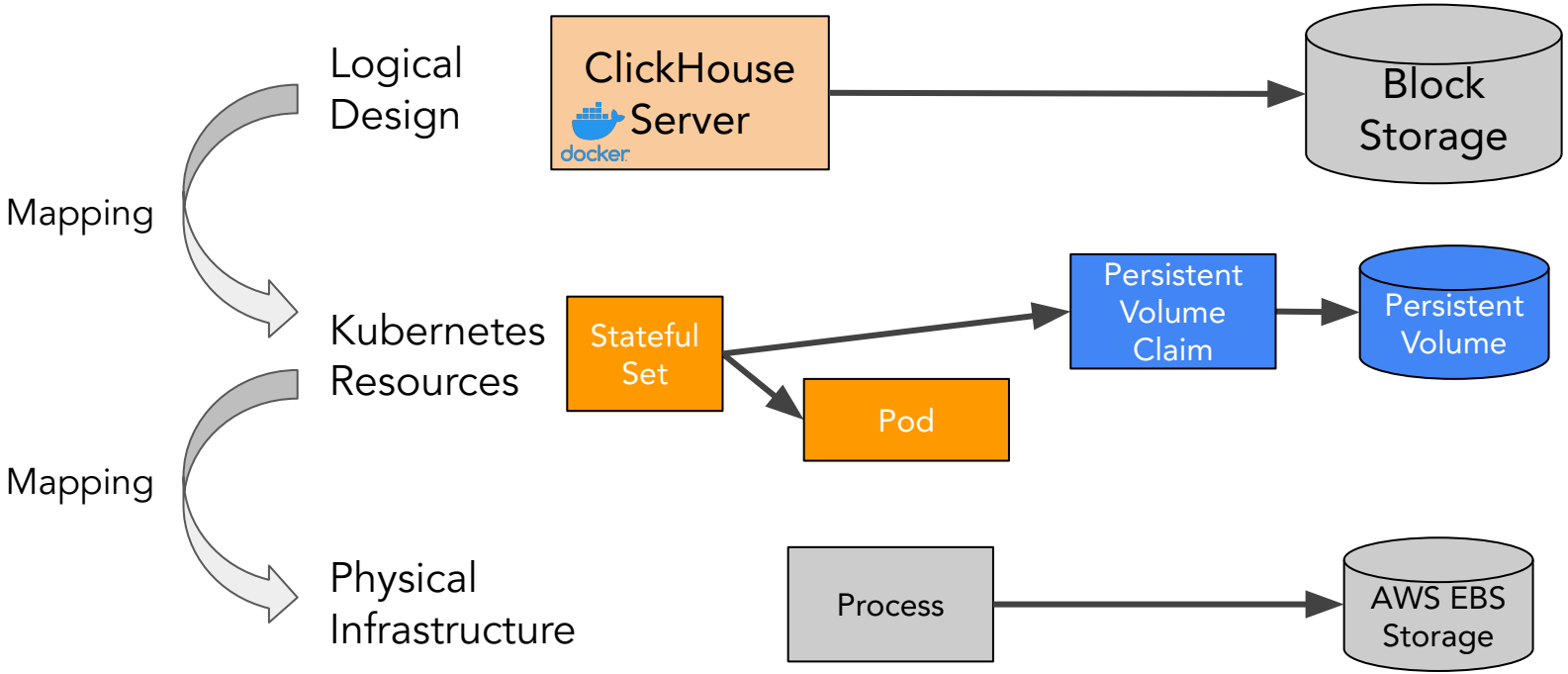
We started by writing an operator



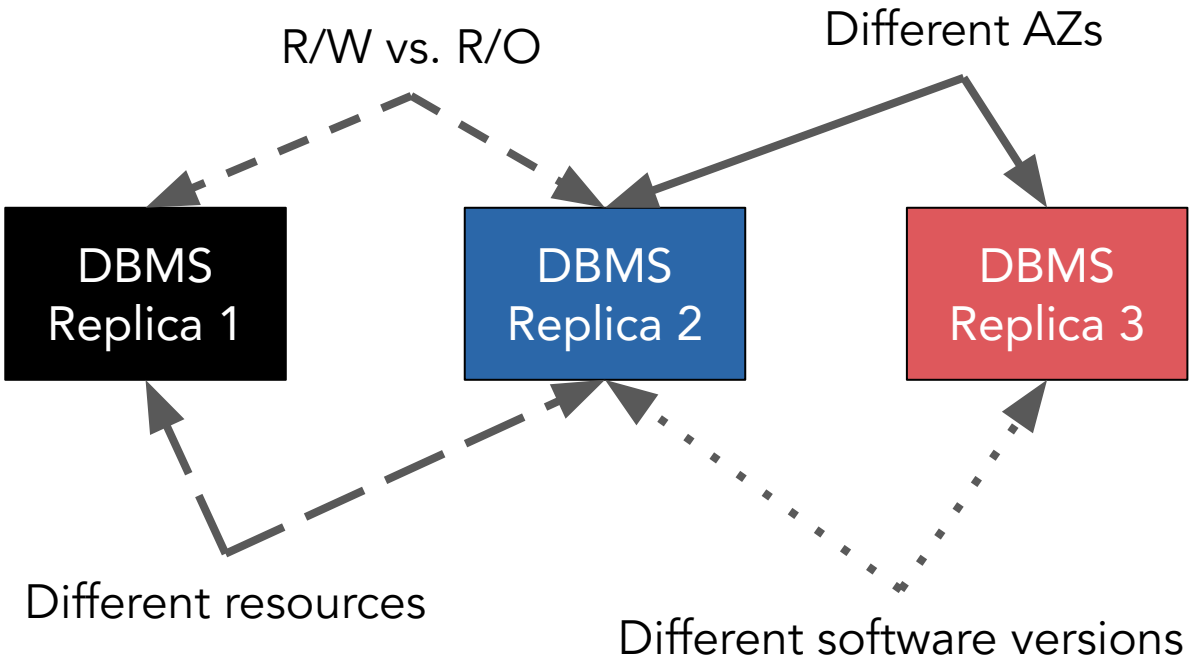
Question: how to represent ClickHouse in K8s resources



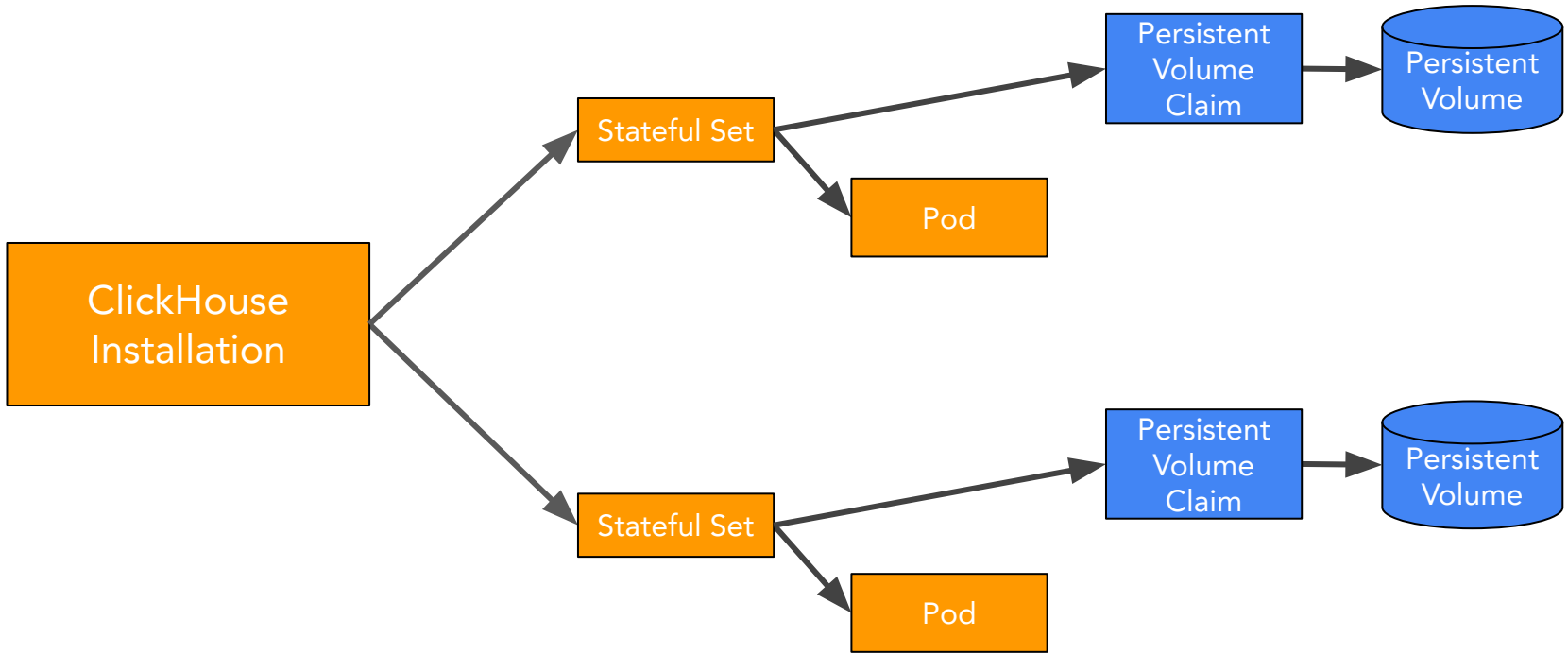
Stateful sets are a useful abstraction for simple services



Problem: Database replicas are asymmetric



We use a stateful set per server to map resources



ClickHouse CRDs echo Stateful Set template syntax

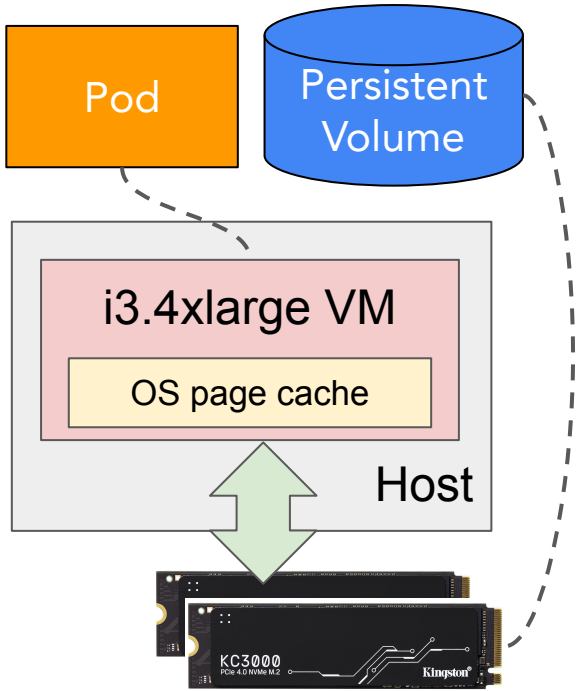
```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"metadata:
  name: "prod"
spec:
  configuration:
    clusters:
      - name: "ch"
        layout:
          replicas:
          - templates:
              podTemplate: clickhouse-zone-2a
          - templates:
              podTemplate: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

Different templates to divide pods by zone

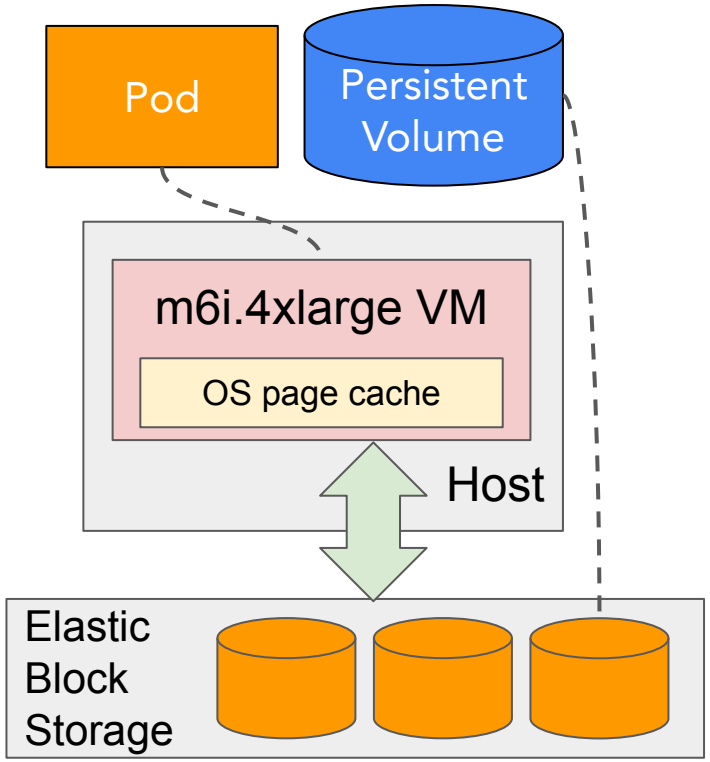
All pods have the same storage spec

Surprising facts about storage performance

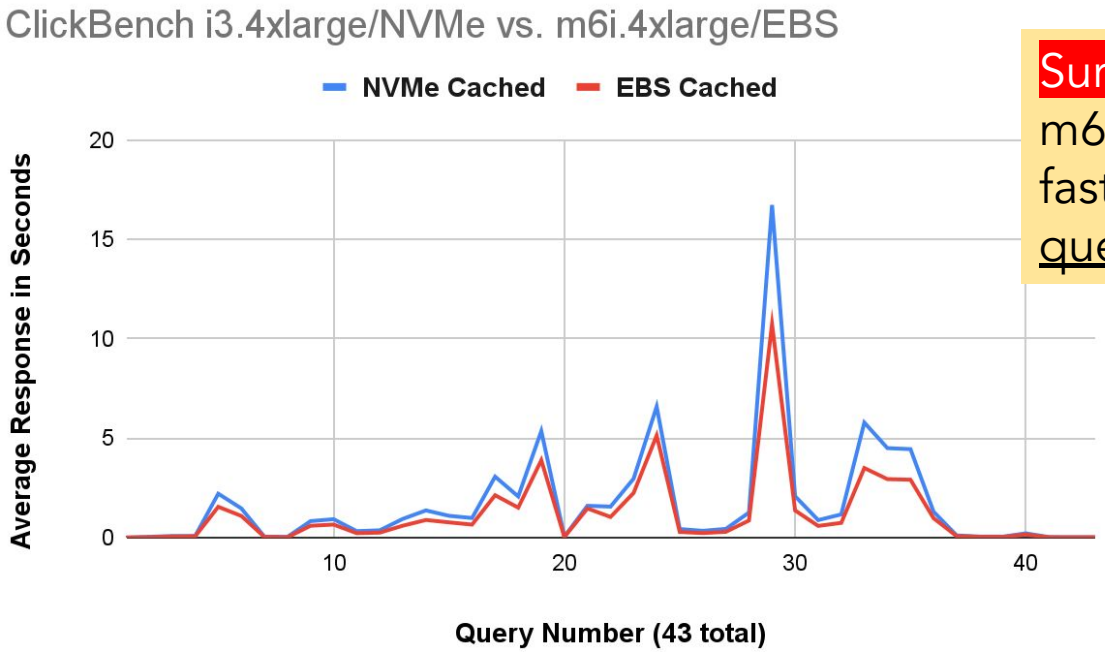
Comparing NVMe SSD vs Cloud Block Storage



OR

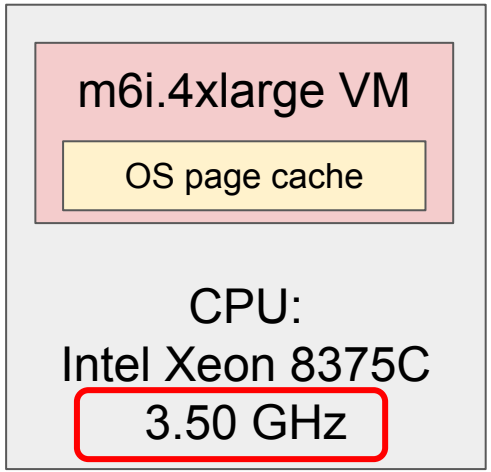


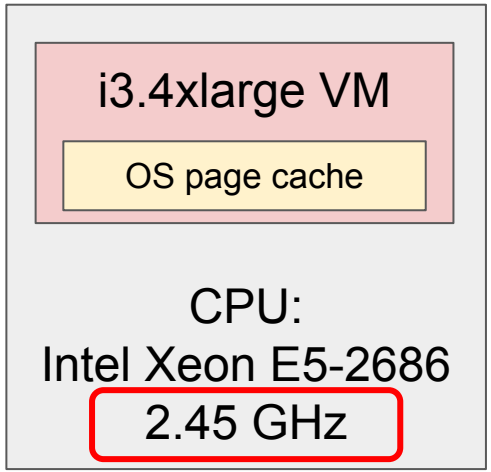
Comparing cached query response for NVMe and EBS



Surprise!
m6i.4xlarge is faster in every query

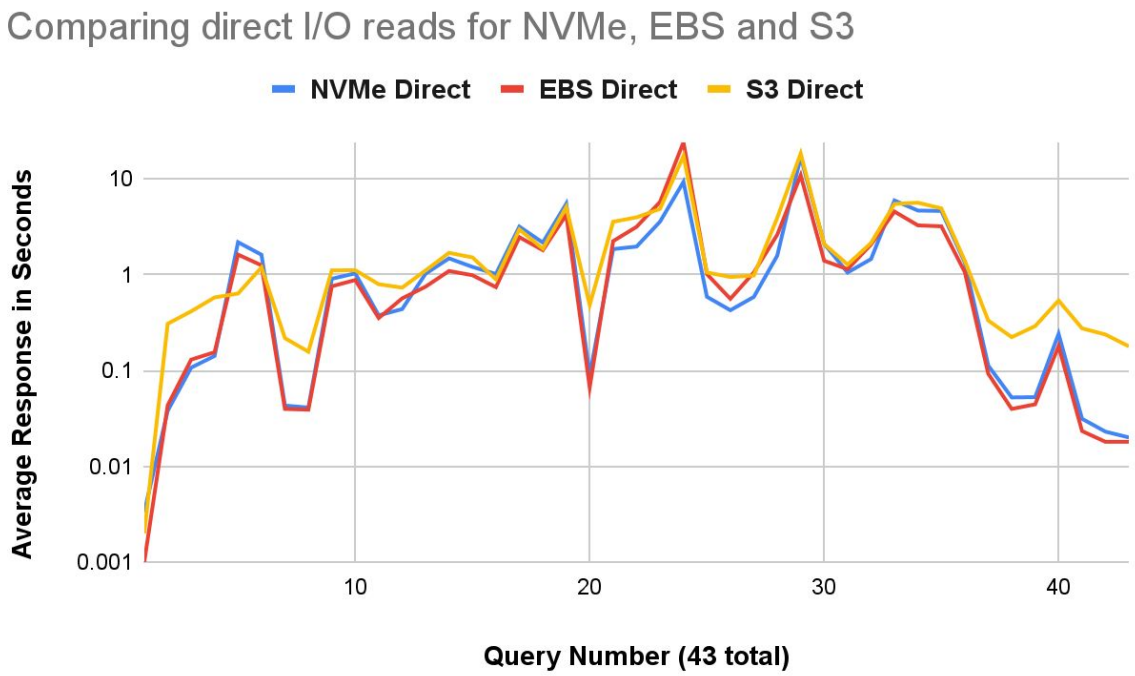
Why is the EBS host is faster?





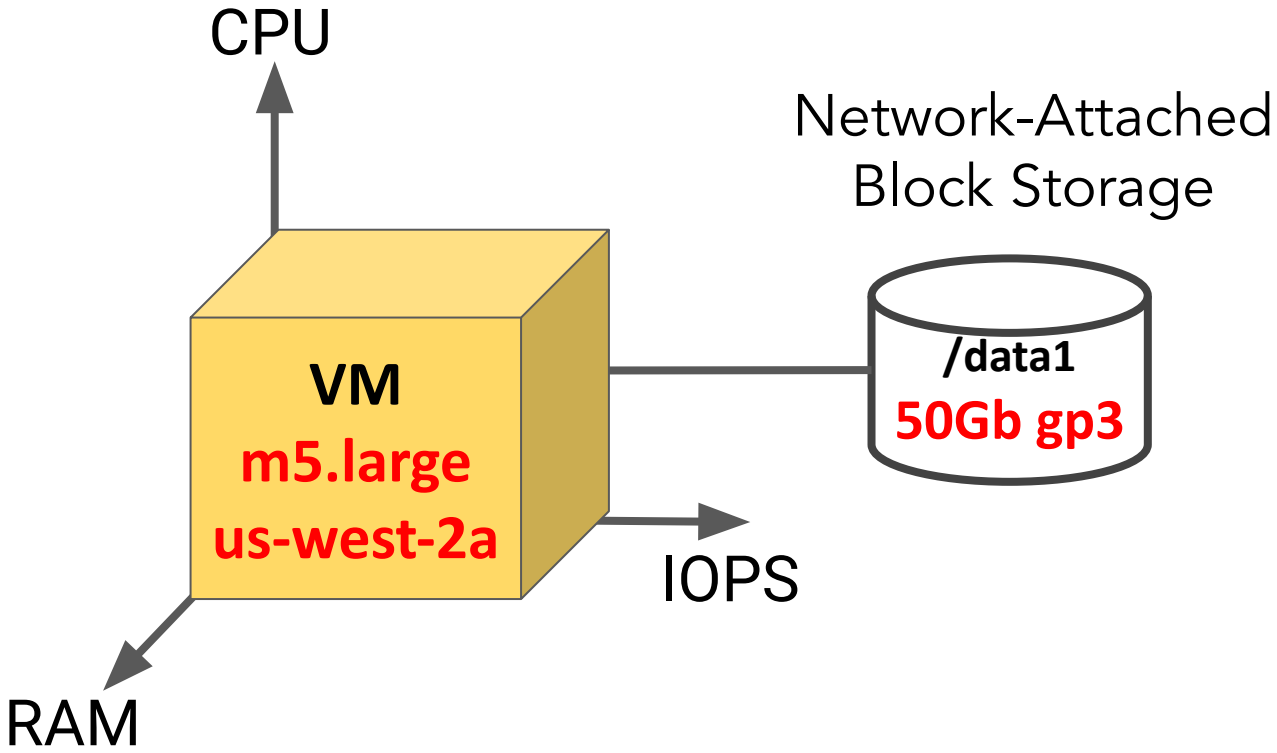
D'Oh! 39% faster clock speed!

Uncached query response for NVMe, EBS, and S3

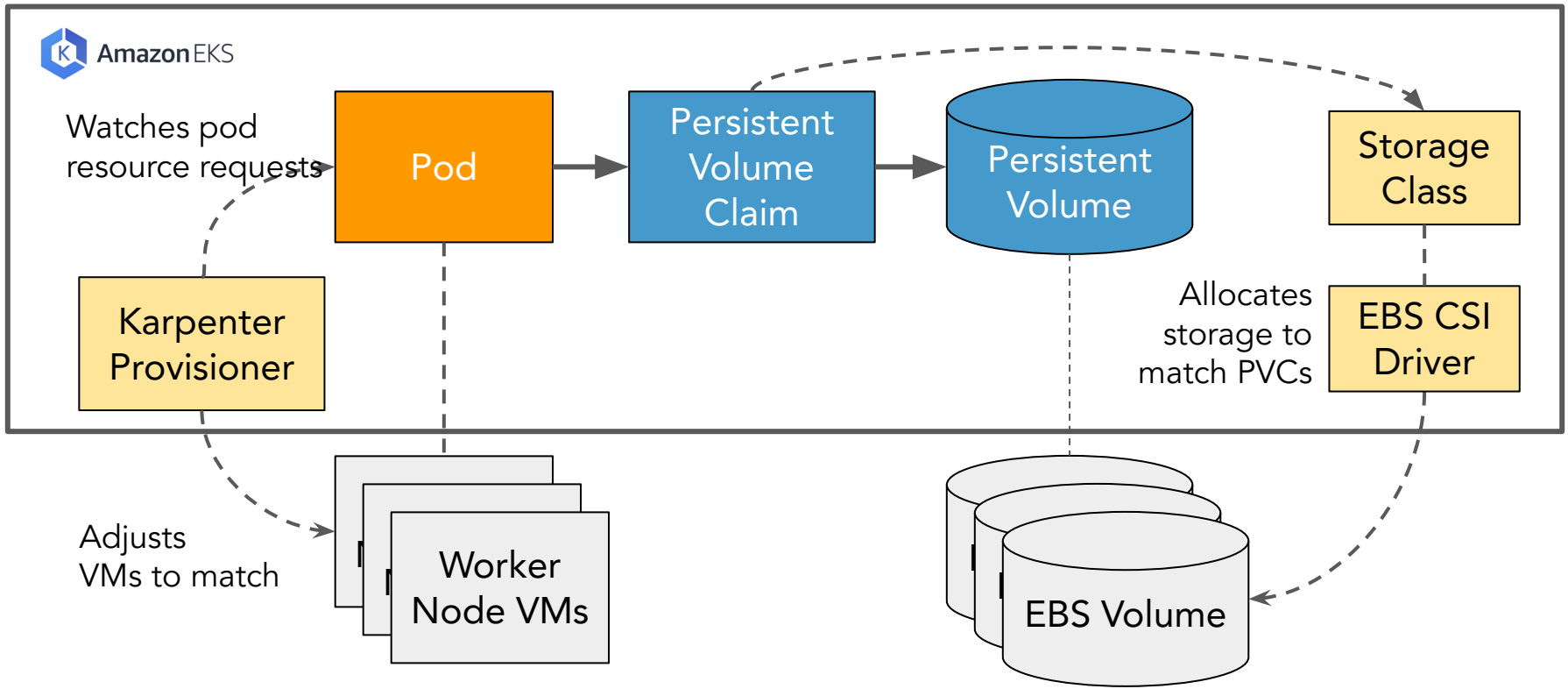


Separation of Storage and Compute

Goal: scale compute and storage independently



Behind the curtain: VM and storage allocation



Instance types force pods to specific VMs

```
podTemplates:
 - name: clickhouse-zone-2a
   spec:
     containers:
     - name: clickhouse
       image: altinity/clickhouse-server:23.3.8.22.altinitystable
     nodeSelector:
       node.kubernetes.io/instance-type: m5.large
   zone:
     key: topology.kubernetes.io/zone
     values:
     - us-west-2a
```

Requires a node with m5.large VM type

Volume claim templates allocate storage for pods

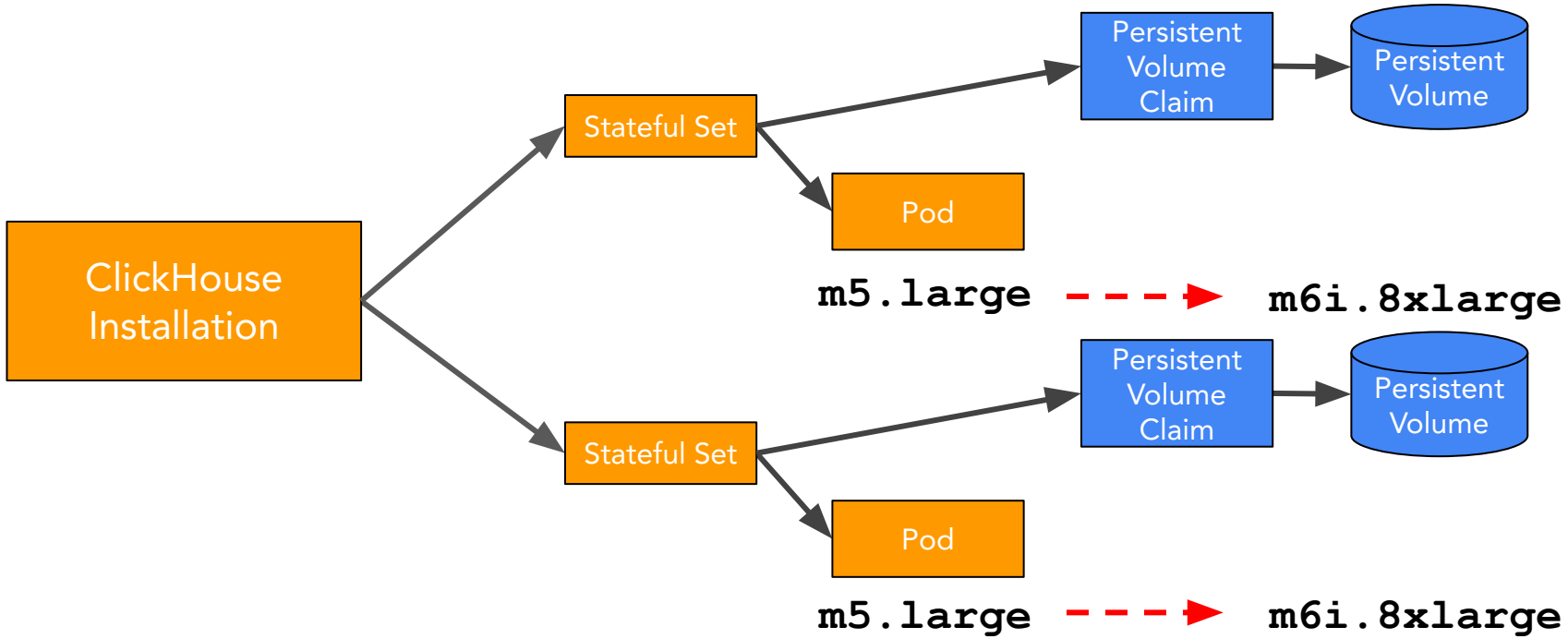
```
volumeClaimTemplates:
  - name: storage
    reclaimPolicy: Retain
    spec:
      storageClassName: gp3-encrypted
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

Do not delete storage if cluster is deleted

Set up storage classes for the storage types that you want

Amount of storage requested

We use a stateful set per server to map resources



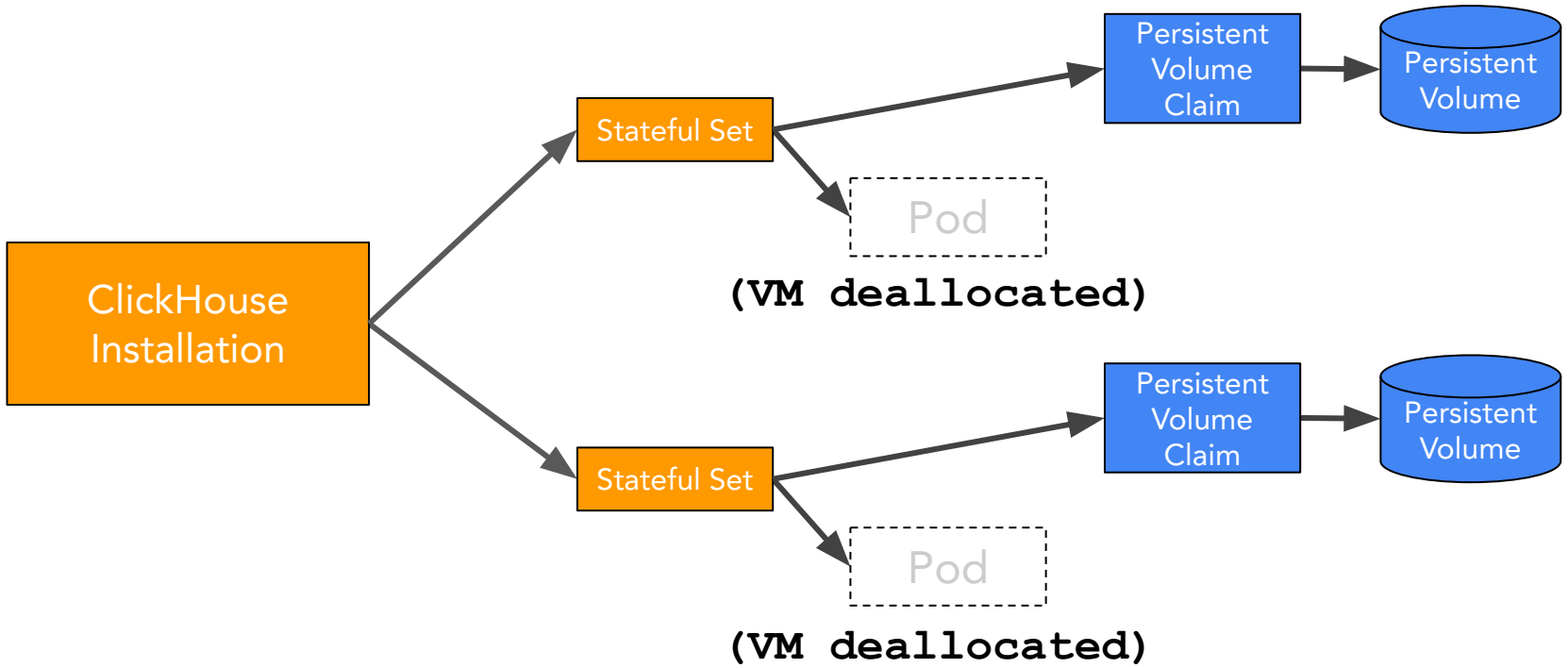
Zero out stateful set replicas to shut off compute

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"metadata:
  name: "prod"
spec:
  stop: "yes"
  configuration:
    clusters:
      - name: "ch"
```


```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: chi-argocd-demo-0-0
spec:
  podManagementPolicy:
OrderedReady
  replicas: 0
  revisionHistoryLimit: 10
```

Turn off compute

Voila! Pods go away



More fiendish tricks to bend storage to your indomitable will

AWS EBS gp3 storage has lots of useful parameters!

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gp3-encrypted
provisioner: ebs.csi.aws.com
parameters:
  encrypted: 'true'
  fsType: ext4
  throughput: '500'
  iops: 3000
  type: gp3
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
```

Parameters are applied only to new persistent volumes

500 MiB/sec disk throughput

3000 IOPS

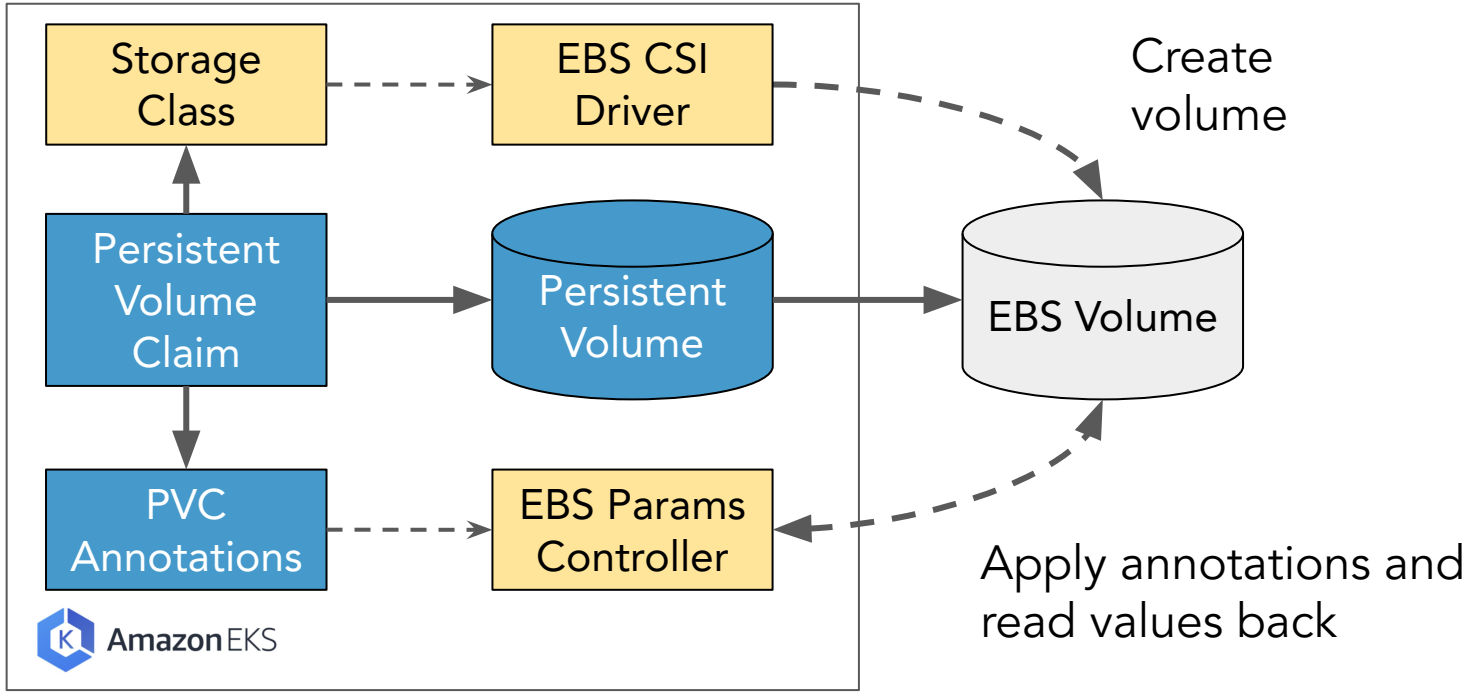
You can increase the size!

We want to change parameter values on demand!

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-gp3-volume
  annotations:
    spec.epc.altinity.com/throughput: 1000
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: gp3-encrypted
```

Apply value to new persistent volumes or whenever values change

Introducing the Altinity EBS Params Controller



EBS Params controller also fetches current values

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-gp3-volume
  annotations:
    spec.epc.altinity.com/throughput: '1000'
    status.epc.altinity.com/iops: '3000'
    status.epc.altinity.com/mod-end-time: '2023-05-22T09:1...'
    status.epc.altinity.com/mod-start-time: '2023-05-22T09...'
    status.epc.altinity.com/mod-state: completed
    status.epc.altinity.com/throughput: '1000'
    status.epc.altinity.com/type: gp3
```

Another nice trick: alter many volumes at once

```
kubectl annotate --overwrite pvc \
-n my-namespace \
-l clickhouse.altinity.com/cluster=my-clickhouse \
spec.epc.altinity.com/throughput=1000
```

Tricks to avoid a restart when extending block storage

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"metadata:
  name: "prod"
spec:
  defaults:
    storageManagement:
      provisioner: Operator
  configuration:
    clusters:
      - name: "ch"
        layout:
          replicas:
          - templates:
              podTemplate: clickhouse-zone-2a
. . .
```

Operator manages storage without using stateful set template

Avoids a restart when extending EBS volumes!

Final words

Our learnings in Kubernetes storage management

- Build on existing Kubernetes resources where possible
- Test performance carefully! The results may surprise you
- Kubernetes + cloud block storage = separated storage and compute
- Use idiomatic Kubernetes tricks like custom controllers to reach out to storage directly

- Where we are going next:
  - Object storage for sure, using NVMe SSD for local cache
  - Disk snapshots maybe

References and appreciations

- https://github.com/Altinity/clickhouse-operator - Altinity Operator
- https://github.com/Altinity/ebs-params-controller - Altinity EBS Params Controller
- https://github.com/ClickHouse/ClickBench - ClickHouse Performance Test
- Why CSI drivers are essential in Kubernetes storage - Fernando Lozano

Special thanks to Alexander Zaitsev and Vlad Klimenko!

Thank you!

Any Questions?

Robert Hodges

rhodges at altinity dot com
LinkedIn
Data on Kubernetes Community Slack