Delivering Production ClickHouse® Apps on the Altinity Kubernetes Operator

Alexander Zaitsev - Altinity CTO
Robert Hodges - Altinity CEO
24 March 2026

Run Open Source ClickHouse® Better

Altinity.Cloud Enterprise Support

Introducing ClickHouse On Kubernetes

Demo Time!!

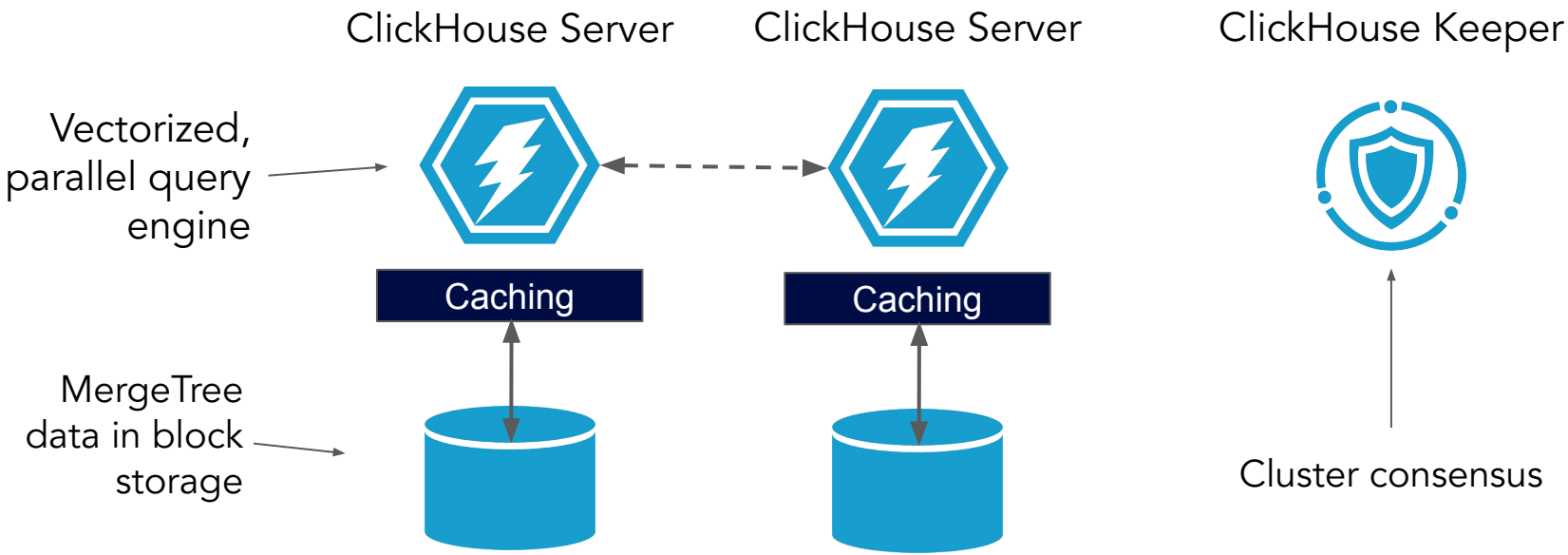
Demo script

```
kubectl create ns test

kubectl apply -f
https://raw.githubusercontent.com/Altinity/clickhouse-operator/master/deploy/operator/clickhouse-operator-install-bundle.ya
ml

cat /Users/alz/Documents/workspace/chi-configs/chi-chk-minimum.yaml

kubectl -n test create -f /Users/alz/Documents/workspace/chi-configs/chi-chk-minimum.yaml

kubectl -n test get pods -w

kubectl -n test get all

kubectl -n test exec -it chi-my-chi-cluster-default-0-1-0 -- clickhouse-client

CREATE TABLE ontime ON CLUSTER '{cluster}'
ENGINE = ReplicatedMergeTree
PARTITION BY Year ORDER BY (Carrier, FlightDate)
EMPTY AS
SELECT * FROM s3('https://altinity-clickhouse-data.s3.amazonaws.com/airline/data/ontime_parquet/2020.parquet');

INSERT INTO ontime
SELECT * FROM s3('https://altinity-clickhouse-data.s3.amazonaws.com/airline/data/ontime_parquet/2020.parquet') limit 100;

select * from ontime limit 1 format Vertical;
```

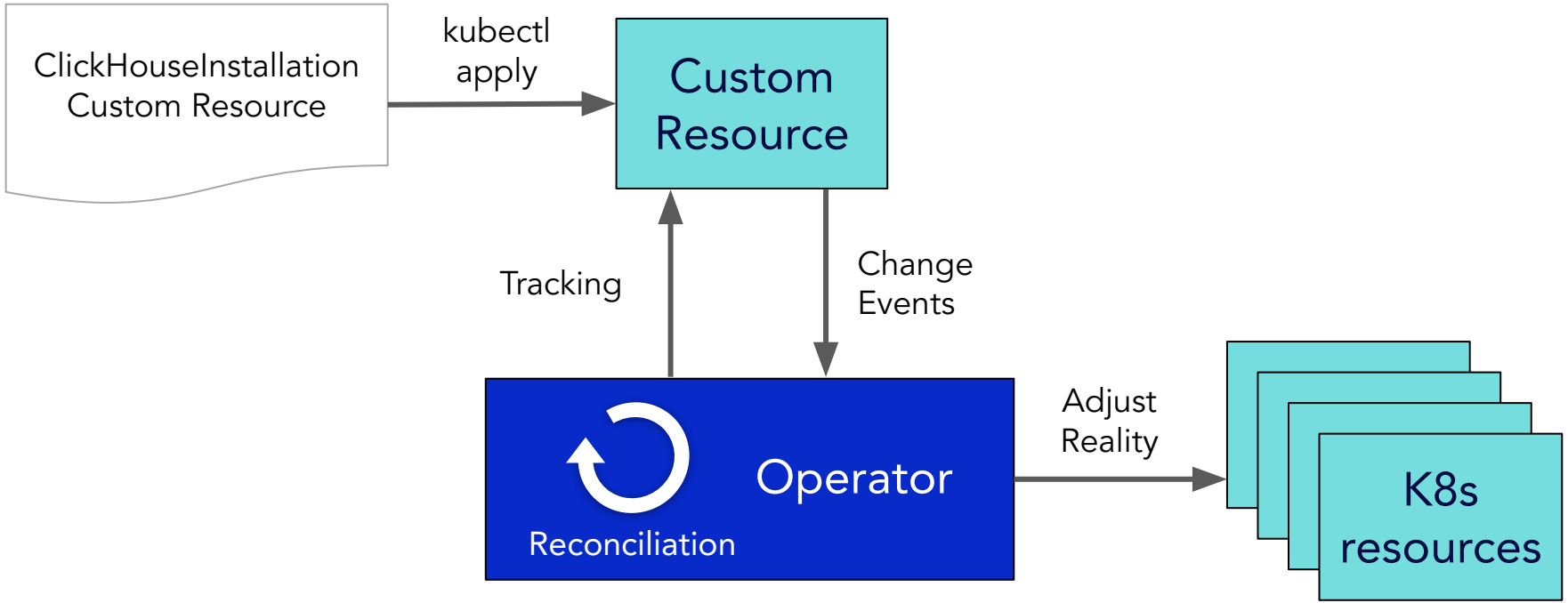
How it works

What is ClickHouse? What is Keeper? How does an operator help?

ClickHouse shared nothing architecture



Operators manage custom resources



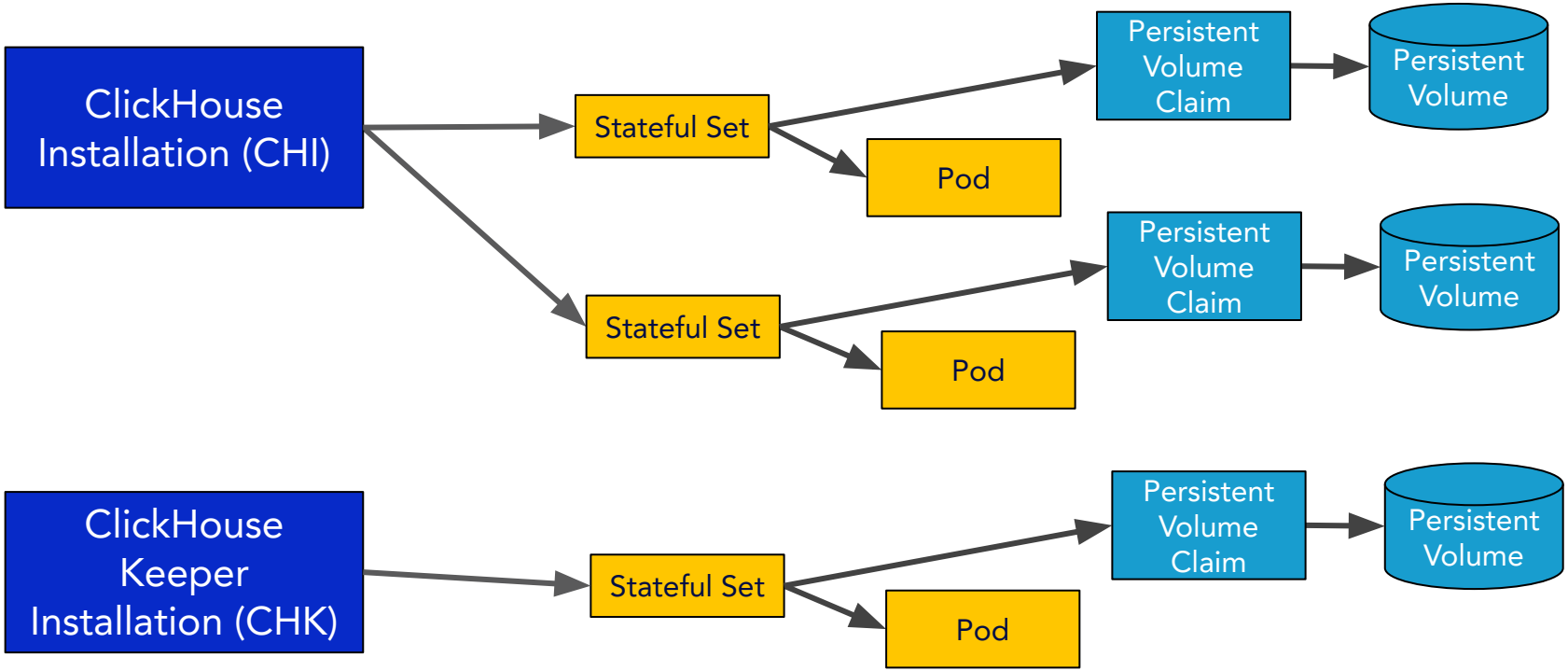
First Installation - ClickHouse Keeper

```
apiVersion: "clickhouse-keeper.altinity.com/v1"
kind: "ClickHouseKeeperInstallation"
metadata:
  name: my-chk
spec:
  defaults:
    templates:
      volumeClaimTemplate: default
  configuration:
    clusters:
      - name: default
  templates:
    volumeClaimTemplates:
      - name: default
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 1Gi
```

First Installation - ClickHouse Cluster

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: my-chi-cluster
spec:
  defaults:
    templates:
      volumeClaimTemplate: default
  configuration:
    zookeeper:
      nodes:
        - host: keeper-my-chk # default name for Keeper my-chk service
          port: 2181
    clusters:
      - name: default
        layout:
          replicasCount: 2
  templates:
    volumeClaimTemplates:
      - name: default
        reclaimPolicy: Retain
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 10Gi
```

We use a stateful set per server to map resources

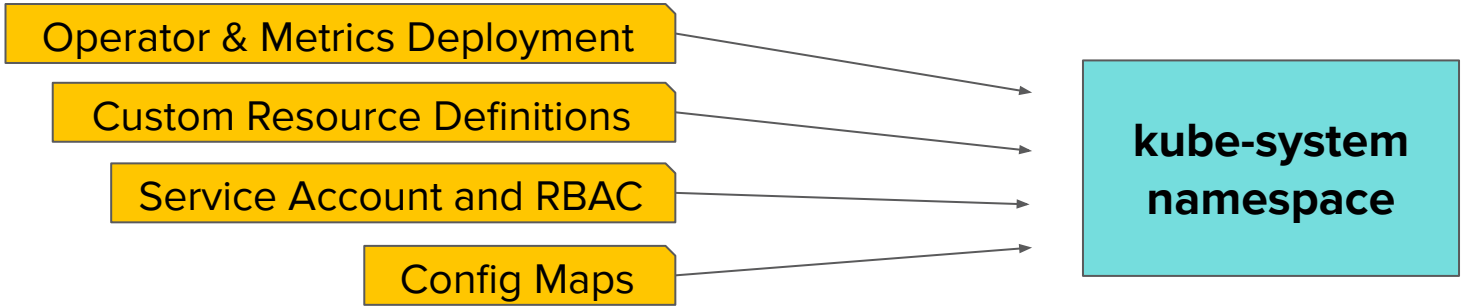


Managing ClickHouse and Keeper on Kubernetes

The basics

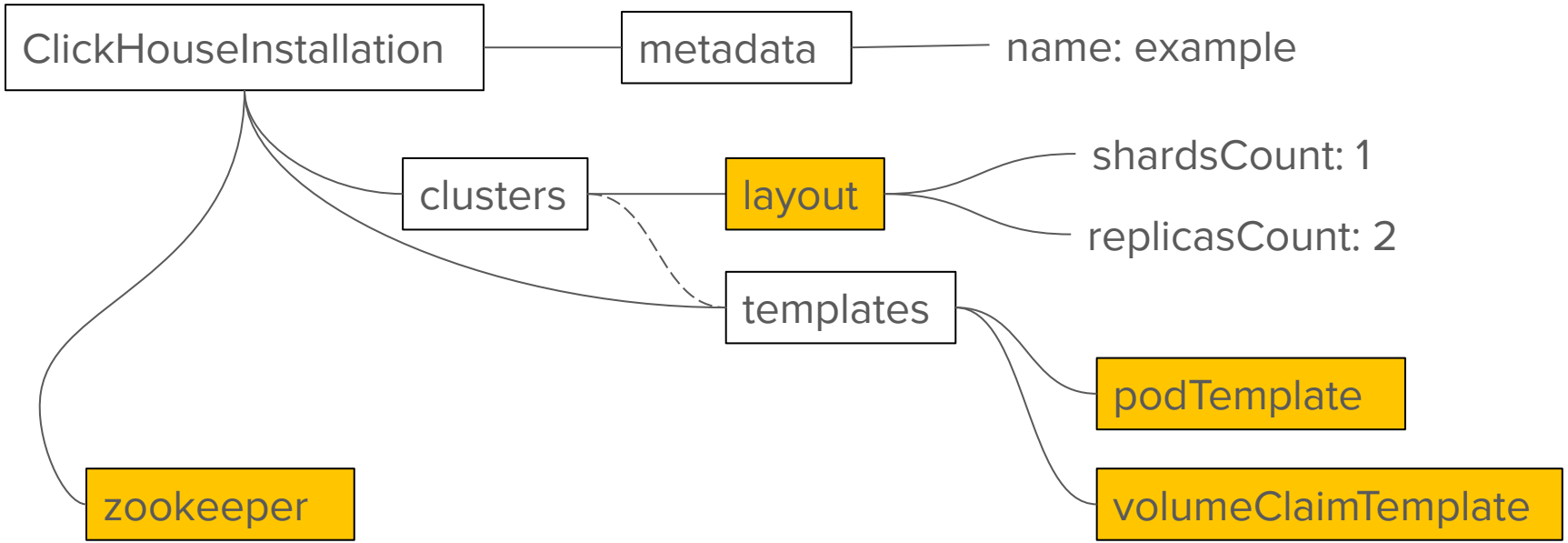
Install the Altinity Operator in 10 seconds or less

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



Details: https://github.com/Altinity/clickhouse-operator/blob/master/docs/quick_start.md

The ClickHouse Installation (CHI) resource



Cluster layouts and locating Keeper

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

Shards and replicas

Definitions for pods and storage

Where is [Zoo]Keeper?

Server settings and configuration files

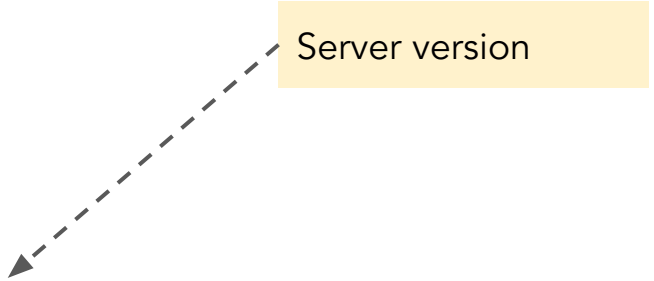
```
settings:
    max_concurrent_queries: 1000
files:
  config.d/filesystem_cache.xml: |
    <clickhouse>
      <filesystem_caches>
        <s3_parquet_cache>
          <path>/var/lib/clickhouse/s3_parquet_cache</path>
          <max_size>25Gi</max_size>
        </s3_parquet_cache>
      </filesystem_caches>
    </clickhouse>
```

Server config setting

Configuration file

Pod definition

```
templates:
  podTemplates:
    - name: replica
      spec:
        containers:
        - name: clickhouse
          image: altinity/clickhouse-server:25.8.16.10002.altinitystable
```



We support any ClickHouse builds: Altinity Stable, ClickHouse Official Builds, Altinity Antalya

We support any build version over 21.11

Storage definition

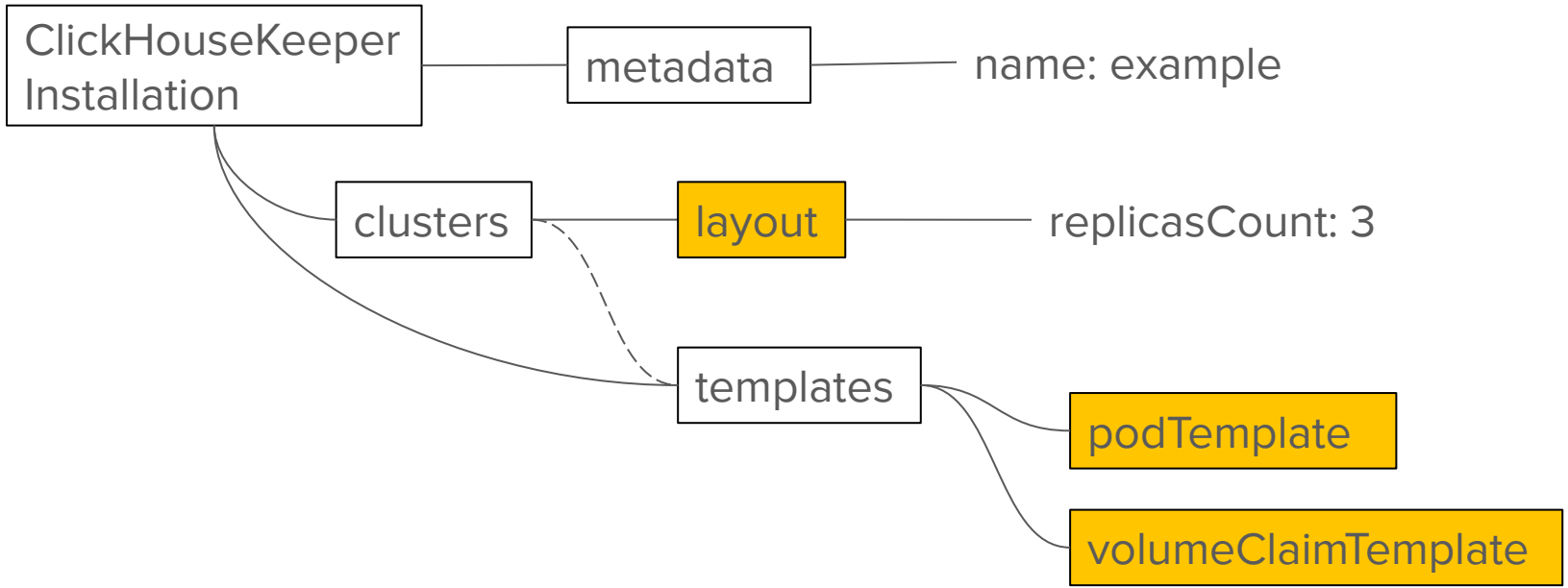
```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC if installation is dropped.
    reclaimPolicy: Retain
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

Protect storage from deletion

Storage size

Allocates storage using the default storage class

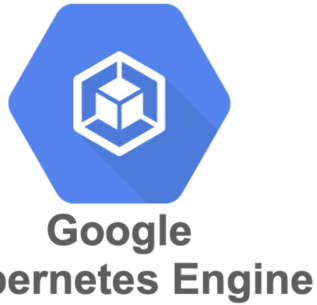
The ClickHouse Keeper Installation (CHK) is very similar



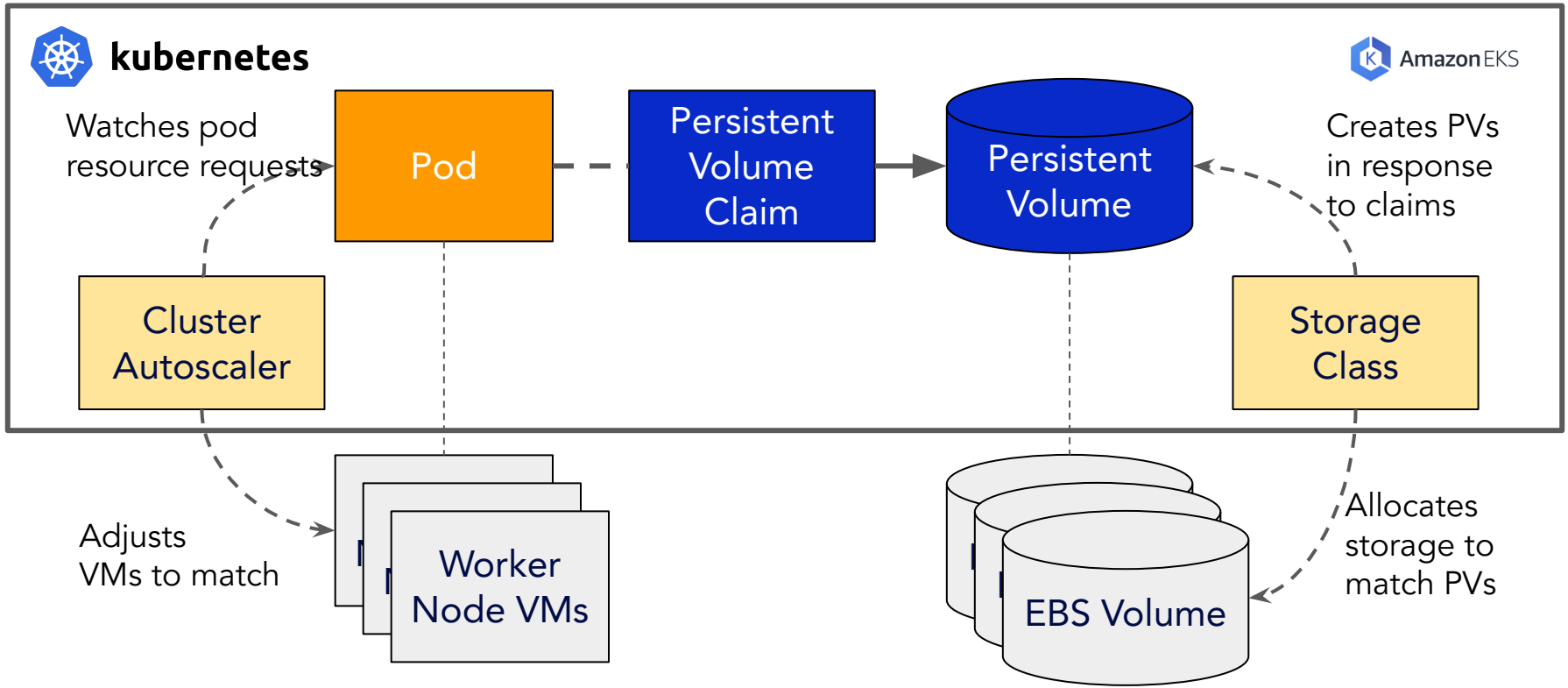
Deployment Tips

How to set up in production environments

Use managed Kubernetes – it's cheap and efficient



Learn how to wire up VM and storage allocation (AWS)



Allocate a separate host for each ClickHouse

```
templates:
  podTemplates:
    - name: replica
      spec:
        nodeSelector:
          node.kubernetes.io/instance-type: m7g.large
        containers:
        - name: clickhouse
          image: altinity/clickhouse-server:25.8.16.10002.altinitystable
        tolerations:
        - key: "dedicated"
          operator: "Equal"
          value: "clickhouse"
          effect : "NoSchedule"
      podDistribution:
        - type: ClickHouseAntiAffinity
          scope: ClickHouseInstallation
```

Host VM type

Only allow ClickHouse servers on this host

Force each server to a different host

Use StorageClasses to adjust storage to taste

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gp3-encrypted
provisioner: ebs.csi.aws.com
parameters:
  encrypted: 'true'
  fsType: ext4
  throughput: '1000'
  iops: 3000
  type: gp3
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
```

Parameters are applied only to new persistent volumes

1000 MiB/sec disk throughput

3000 IOPS

You can increase the size!

You can change storage on live systems

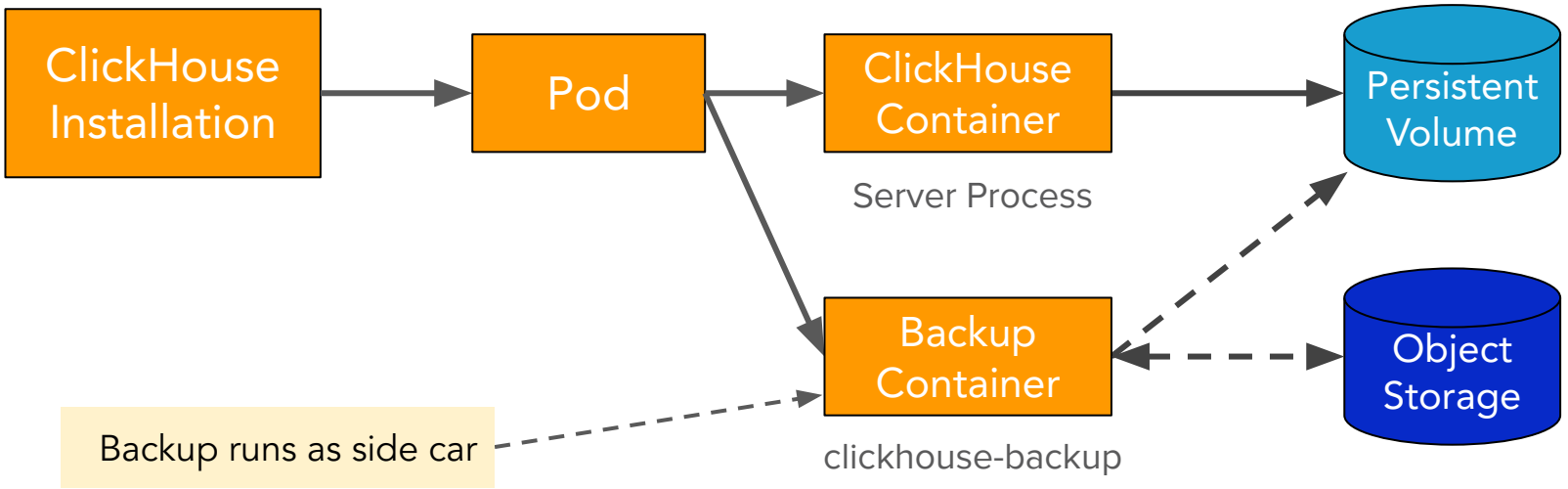
```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC if installation is dropped.
    reclaimPolicy: Retain
    spec:
      storageClassName: gp3-encrypted
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 100Gi
```

Use specific storage class

Patch the storage request to increase storage

See https://altinity.com/blog/managing-ebs-gp3-volumes-in-eks

Backup configuration with Altinity clickhouse-backup



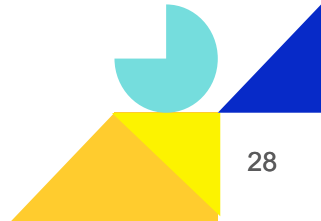
See https://github.com/Altinity/clickhouse-backup/blob/master/Examples.md

Advanced Features for Building Data Platforms

Five things every ClickHouse installation needs to cover

Templating

- Storage templates – define storage configuration
- Pod templates – define compute configuration and deployment
- Service templates – define network configuration
- ClickHouseInstallation templates – define any part of the spec!



Service Templates

```
  defaults:
    templates:
      serviceTemplate: service-template
      serviceTemplates:
          - service-template-internal
          - service-template
      replicaServiceTemplate:
replica-service-template
      clusterServiceTemplate:
cluster-service-template
```

serviceTemplate – one per CHI
replicaServiceTemplate – one pre host (replica or shard)
clusterServiceTemplate – one per cluster

```
  templates:
    serviceTemplates:
      - name: service-template
        generateName: "service-{chi}"
        spec:
          ports:
            - name: http
              port: 8123
            - name: tcp
              port: 9000
          type: LoadBalancer
          externalTrafficPolicy: Local
```

Supported macros in name generation:
{chi}/{chk}
{shard}
{replica}
{cluster}
..

ClickHouseInstallation templates

- Separate Kubernetes resource ClickHouseInstallationTemplate (CHIT)
- Spec is identical to ClickHouseInstallation
- One or many CHIT can be included to CHI to inject parts of the configuration
- Typical use cases:
  - Provide defaults
  - Provide common parts to multiple CHI
  - Override CHI behavior when modifying CHI directly is not possible

ClickHouseInstallation templates

```
apiVersion:
"clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
...
spec:
  useTemplates:
    - name: clickhouse-version
```

reference by name

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallationTemplate"
metadata:
  name: clickhouse-version
spec:
  defaults:
    templates:
      podTemplate: default
  templates:
    podTemplates:
      - name: default
        spec:
          containers:
            - name: clickhouse-pod
              image:
altinity/clickhouse-server:25.8.16.10002.altinitystab
              imagePullPolicy: IfNotPresent
```

```yaml
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallationTemplate"
metadata:
  name: postgresql-port
spec:
  defaults:
    templates:
      clusterServiceTemplate: postgresql
  templating:
    chiSelector:
      clickhouse.altinity.com/chi: my-cluster
    policy: auto
  configuration:
    settings:
      postgresql_port: 9005
  templates:
    serviceTemplates:
      - name: postgresql
        generateName: clickhouse-{chi}-postgresql
          ports:
            - name: postgresql
              port: 9005
              targetPort: 9005
          type: ClusterIP
    podTemplates:
      - name: clickhouse-replica-1
        spec:
          containers:
            - name: clickhouse-pod
              ports:
              - containerPort: 9005
                name: postgresql
                protocol: TCP
```

Applies to my-cluster CHI automatically

Enables port in ClickHouse

Enables port in Service

Enables port in Pod

Rescaling – adding replicas

```
clusters:
  - name: default
    layout:
      replicasCount: 2
```

```
clusters:
  - name: default
    layout:
      replicasCount: 3
```

- Creates new pods and storage
- Creates schema on new replicas
- Waits for replication to catch up
- Includes new nodes to load balancer

No restarts, can be done on a running cluster!

Rescaling – adding shards

```
clusters:
  - name: default
    layout:
      shardsCount: 2
```

```
clusters:
  - name: default
    layout:
      shardsCount: 4
```

- Creates new pods and storage
- Creates schema on new shards keeping the shard scope
- Waits for replication to catch up
- Includes new nodes to load balancer
- **DOES NOT** do re-sharding

Rescaling – removing shards or replicas

```
clusters:
  - name: default
    layout:
      replicasCount: 3
```

```
clusters:
  - name: default
    layout:
      replicasCount: 2
```

- Removes nodes from load balancer service
- Removes resources
- Removes replica from (Zoo)Keeper

Rescaling – adding more storage

```
volumeClaimTemplates:
  - name: storage
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

```
volumeClaimTemplates:
  - name: storage
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 100Gi
```

```
spec
  defaults:
    storageManagement:
      provisioner: StatefulSet | Operator
```

- StatefulSet provisioner (default) – manages PVC as a part of STS, requires restart
- Operator provisioner – manages PVC directly, no restart

Upgrades and Configuration Management

- ClickHouse image change – rolling pod restart
- ClickHouse podTemplate or storageTemplate change – rolling statefulset recreation
- Cluster configuration change:
  - Rolling restart via SYSTEM SHUTDOWN, or
  - No restart

Rolling restart logic

1. Excludes node from a load balancer service
2. Excludes node from remote_servers (if there are replicas) and waits for exclusion to happen
3. Waits for queries to finish
4. Re-starts or recreates the pod
5. Includes node to remote_servers
6. Includes node to a load balancer service

If there are many shards – multiple shards can be processed in parallel.

ClickHouse Configuration

Cluster level:

```
settings:
  dictionaries_config: config.d/dict_*.xml
  compression/case/method: zstd
files/config.d:
files/users.d:
files/conf.d:
```

Common per cluster

Can be customized per replica

Nested paths are supported

Replica level:

```
 configuration:
   clusters:
   - layout:
       replicas:
       - settings:
           placement/availability_zone:
us-east-1a
       - settings:
           placement/availability_zone:
us-east-1b
```

Maintenance

```
spec:
  stop: yes | no
  restart: RollingUpdate
  suspend: yes | no
  troubleshoot: yes | no
  taskID: my-task-1
```

Stops the cluster keeping the storage

Triggers rolling restart of all nodes

Suspends the reconciler. Any changes to CHI will be ignored

Starts ClickHouse pods with a bash loop, so it can be accessed to analyze the crash

Forces reconcile
TaskIDs are deduped

Reconciler configuration

```
apiVersion: clickhouse.altinity.com/v1
kind: ClickHouseOperatorConfiguration
metadata:
  name: custom
spec:
  reconcile:
    runtime:
       reconcileShardsThreadsNumber: 100
       reconcileShardsMaxConcurrencyPercent:
50
    host:
      wait:
        exclude: true
        queries: true
        include: false
      replicas:
        new: yes
        delay: 10
    statefulSet:
      update:
        timeout: 600
```

```
apiVersion: clickhouse.altinity.com/v1
kind: ClickHouseInstallation

spec:
  reconcile:
    statefulSet:
      recreate:
        onUpdateFailure: abort
    host:
      wait:
        probes:
          startup: yes
          readiness: no
```

Operator configuration provides global defaults

Can be altered at specific CHI

See https://github.com/Altinity/clickhouse-operator/blob/master/config/config.yaml

Monitoring

- metrics-exporter sidecar to operator
- Prometheus endpoint at 8888 port
- Collects metrics, events, table stats and more
- More metrics compared to built-in ClickHouse Prometheus endpoint
- Can export user-defined metrics from system.custom_metrics or other table

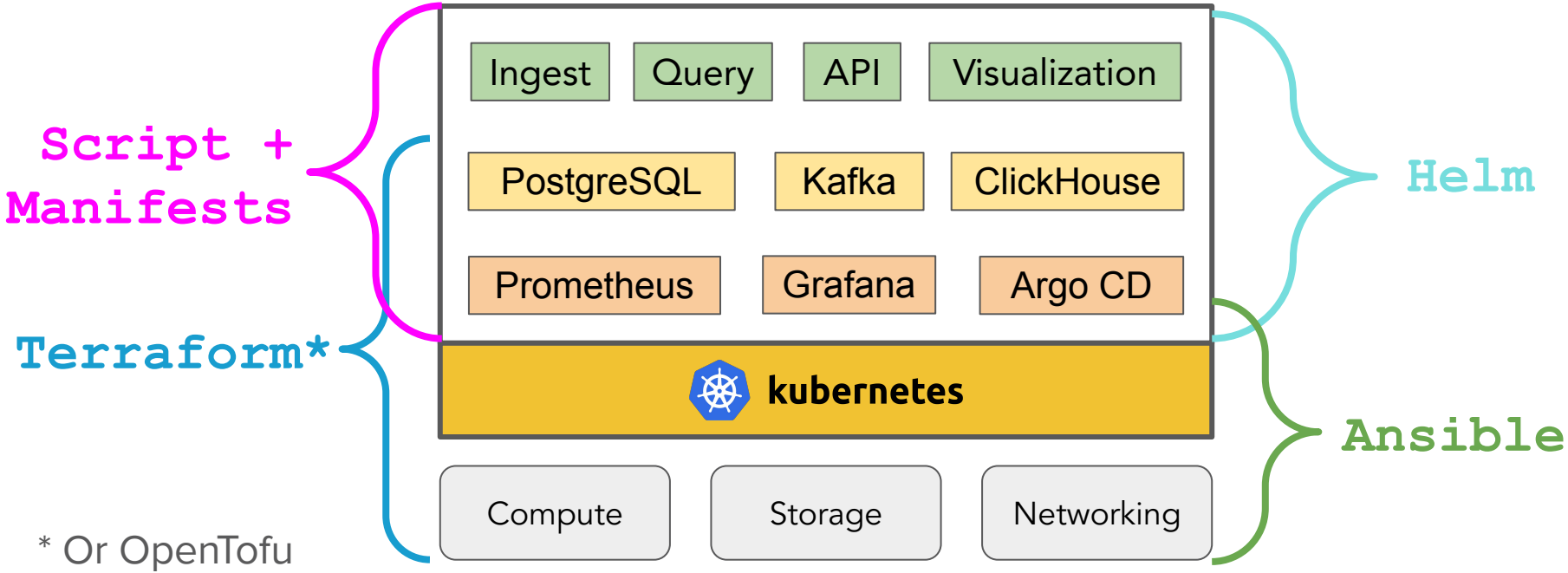
```
tablesRegexp: "^(metrics|custom_metrics)$"
```

- Grafana dashboards
  https://github.com/Altinity/clickhouse-operator/tree/master/grafana-dashboard

How to Automate

You love Terraform and Helm. So do we.

Terraform and Helm are the most popular ways to deploy



* Or OpenTofu

Terraform can deploy cloud infra *and* Kubernetes apps

1. Open source on AWS: Use the Altinity EKS terraform module

2. Altinity.Cloud on AWS/GCP/Azure: Use the BYOC* Terraform Module

* Bring Your Own Cloud

Helm can deploy Kubernetes applications

```
# Install Altinity helm charts
helm repo add altinity https://helm.altinity.com
helm repo update

# Install the operator
helm install clickhouse-operator \
  altinity/altinity-clickhouse-operator

# Install a sample ClickHouse installation
helm install my-example altinity/clickhouse \
  --set operator.enabled=false
```

More info: https://github.com/Altinity/helm-charts

Differences between the Altinity and ClickHouse Inc Operators

A handy cheatsheet

Altinity vs ClickHouse Inc Operators

	Altinity	**ClickHouse Inc**
Year introduced	2019	2026
Production use	Thousands of clusters, including Altinity.Cloud, eBay, OpenAI, etc.	very limited?
Certified k8s platforms	EKS, GKE, AKS, DKE, Linode, OpenShift	?
Certified Builds	CH Official Builds, Altinity Stable, Antalya	CH Official Builds
Configuration	Fully flexible	Limited
Supported DB engines	All, with automatic schema propagation	Only Replicated DBs
Supported coordination	ZooKeeper and Keeper	Keeper
Backup / restore	Altinity clickhouse-backup sidecar	Not implemented yet
Open Source	Apache 2.0	Apache 2.0

Major Planned New Features in 2026

- Post-start/pre-stop SQL hooks
- Better integration of CHI and CHK
- Use PV snapshots when adding new replicas
- Plugin architecture, e.g. for backup

Altinity Operator GitHub

Thank you!
Questions?

Contact us:

- Slack - https://altinity.com/slack
- Altinity Operator GitHub project
- Find out more - http://altinity.com

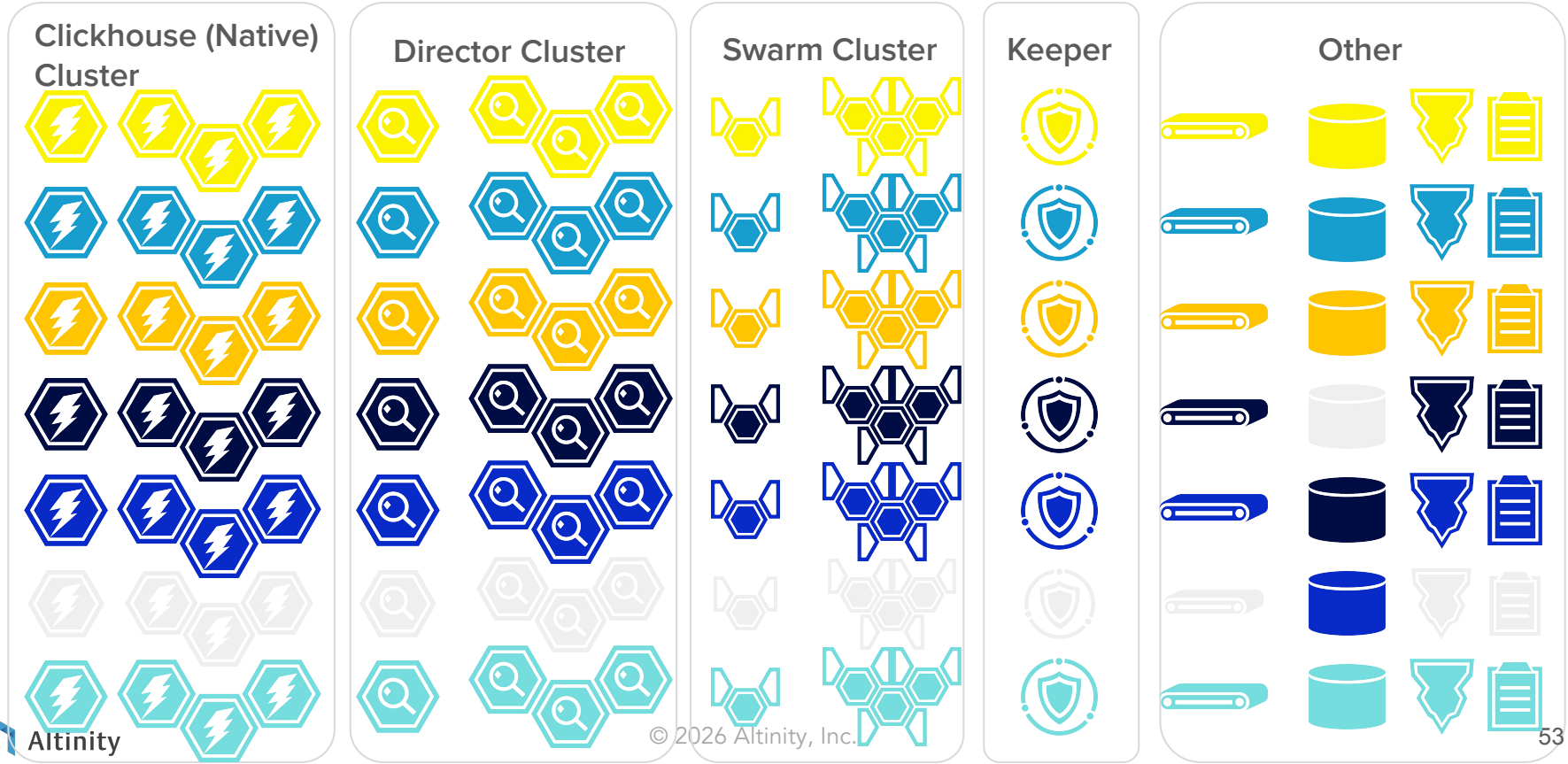
We're hiring!

ALTINITY

Ensure your compute and storage match database needs

1. AWS Graviton instances are fast and cheap
2. Block storage enables vertical scaling and can failover when host dies
3. NVMe SSD gives the best raw performance but is tied to a single host and hard to configure in Kubernetes
4. Object storage is permission to play for large analytic databases

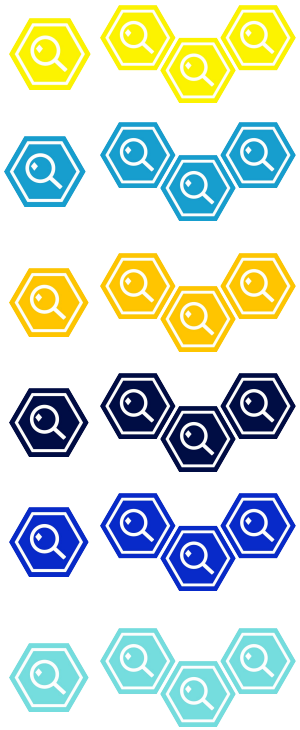
Icons- Transparent



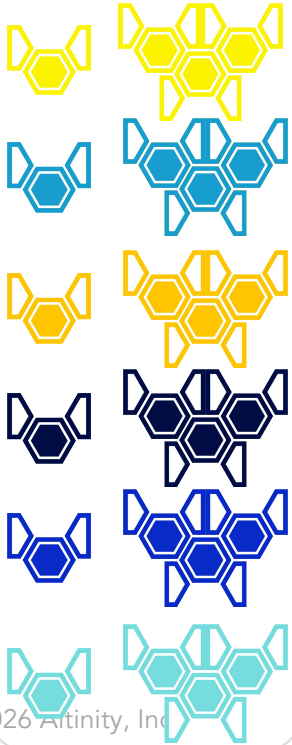
Icons-Stackable on White Backgrounds

Clickhouse (Native) Cluster

Director Cluster



Swarm Cluster



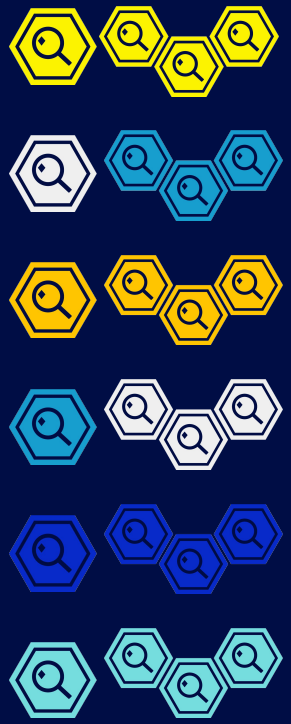
Keeper

Other

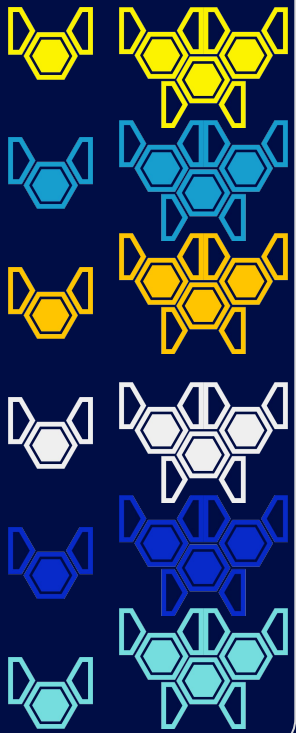
Icons-Stackable on Dark Backgrounds

Clickhouse (Native) Cluster

Director Cluster

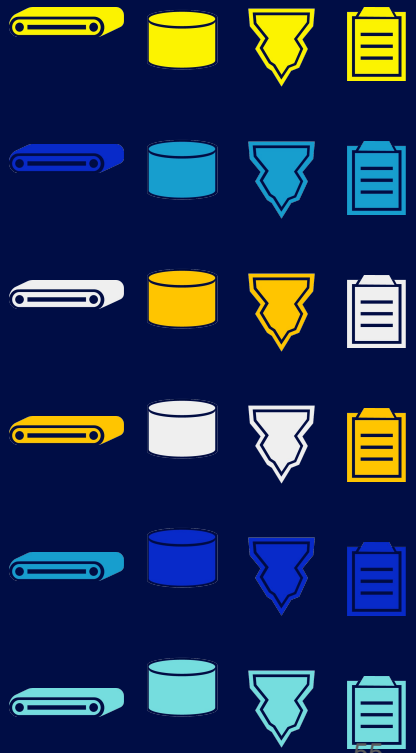


Swarm Cluster



Keeper

Other



Rescaling – adding more disks

```
templates:
  volumeClaimTemplates:
    - name: disk1
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

```
  settings:
    storage_configuration/disks/disk2/path: /var/lib/clickhouse2/
    storage_configuration/policies/default/volumes/default/disk: default
    storage_configuration/policies/default/volumes/disk2/disk: disk2
templates:
  volumeClaimTemplates:
    - name: disk1
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - name: disk2
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
  podTemplates:
    - name: default
      spec:
        containers:
          - name: clickhouse-pod
            volumeMounts:
              - name: disk1
                mountPath: /var/lib/clickhouse
              - name: disk2
                mountPath: /var/lib/clickhouse2
```

Restart rules

```
configurationRestartPolicy:
  rules:
    - version: "*"
      rules:
        - settings/*: "yes"
        # single values
        - settings/access_control_path: "no"
        - settings/dictionaries_config: "no"
        - settings/max_server_memory_*: "no"
        - settings/max_*_to_drop: "no"
        - settings/max_concurrent_queries: "no"
        - settings/models_config: "no"
        - settings/user_defined_executable_functions_config: "no"
        # structured XML
        - settings/logger/*: "no"
        - settings/macros/*: "no"
        - settings/remote_servers/*: "no"
        - settings/user_directories/*: "no"
        # these settings should not lead to pod restarts
        - settings/display_secrets_in_show_and_select: "no"
        - zookeeper/*: "no"
        - files/*.xml: "yes"
        - files/config.d/*.xml: "yes"
        - files/config.d/*dict*.xml: "no"
        - files/config.d/*no_restart*: "no"
        # exceptions in default profile
        - profiles/default/background_*_pool_size: "yes"
        - profiles/default/max_*_for_server: "yes"
```

Defined in operator configuration

Can be altered using ClickHouseOperatorConfiguration resource

Plan to utilize `system.server_settings.changeable_without_restart` eventually

Trick to avoid restarts for certain configuration files