FOSSY 2026

Cheap, Cheap, Cheap!

6 Best Practices to Save Money on Your ClickHouse® Bill

Robert Hodges - Altinity CEO

Run Open Source ClickHouse® Better

Altinity.Cloud Enterprise Support

ClickHouse, a real-time analytic database

Understands SQL

Runs on bare metal to cloud

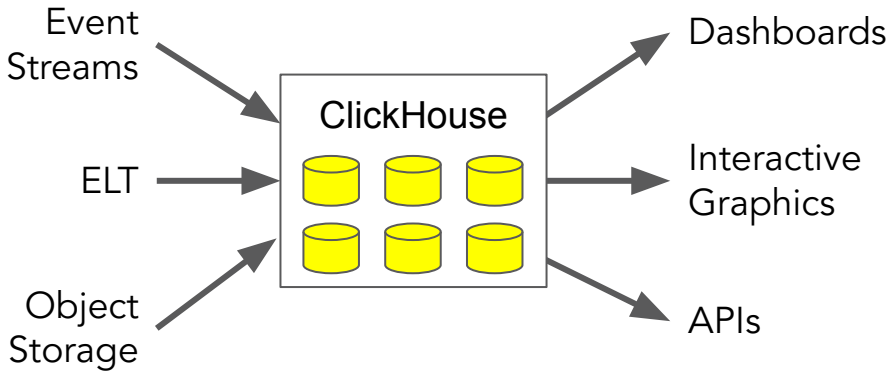
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

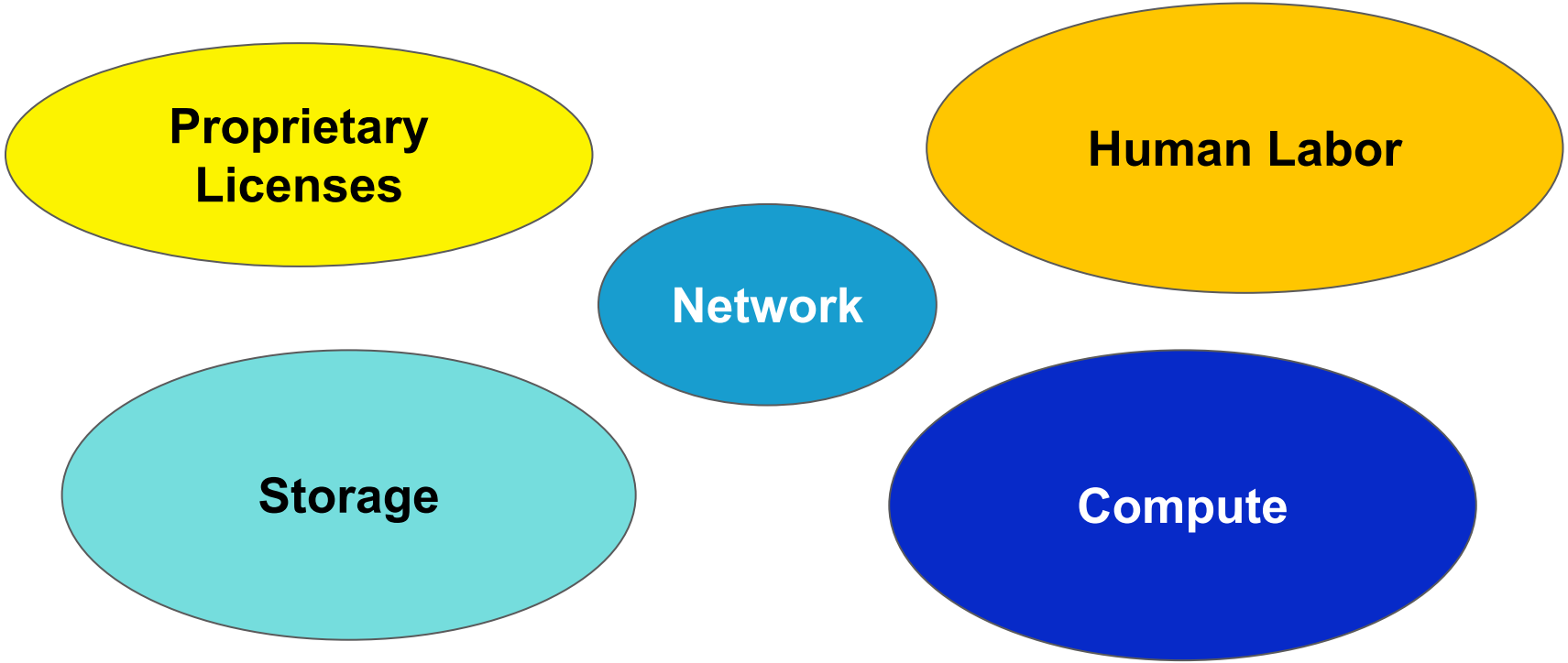
Scales to many petabytes

Is Open source (Apache 2.0)



It's the standard engine for low-latency analytics

What are the principle operating costs in analytic databases?



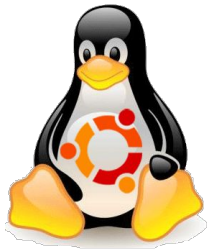
Develop on a laptop with 100% open source

© 2025 Altinity, Inc.

Three open source dev patterns that work on a laptop

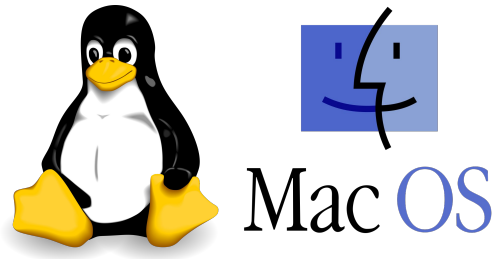
Install ClickHouse packages directly

```
sudo apt install -y
clickhouse-server
clickhouse-client
```



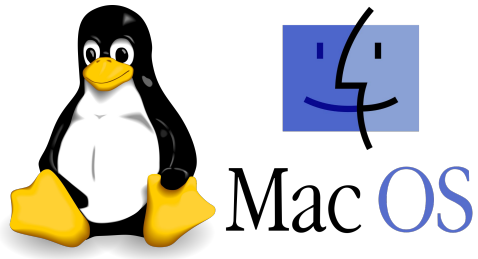
Run ClickHouse using docker

```
docker run -d --name
clickhouse-server
...
```



Run complete ClickHouse app with docker compose

```
docker compose up -d
```



Let’s build a ClickHouse “app” with Ubuntu + curl

```
cat > Dockerfile << END
# Simple analytic client with curl installed.
FROM ubuntu:22.04
RUN apt-get update && apt-get install -y curl
CMD ["sleep", "infinity"]
END

docker build -t myclient:latest - < Dockerfile
```

Create a compose file for ClickHouse and the client

```
cat > docker-compose.yml << END
services:
  antalya:
      image: altinity/clickhouse-server:26.3.13.20001.altinityantalya
      ports:
      - "8123:8123"
      - "9000:9000"
      volumes:
      - antalya-data:/var/lib/clickhouse
      configs:
      - source: network-access
        target: /etc/clickhouse-server/users.d/network-access.xml
  client:
      image: myclient:latest
volumes:
  antalya-data:
configs:
  network-access:
    content: |
      <clickhouse> <users> <default> <networks>
        <ip>::/0</ip>
      </networks> </default> </users> </clickhouse>
END
```

...And log in using your “application”

```
$ docker compose up -d
...

$ docker ps
CONTAINER ID   ...                   NAMES
dff28a725b38   altinity/clickh...    demo-antalya-1
23a641654ac2   myclient:latest...    demo-client-1

$ docker exec -it demo-client-1 curl \
  'http://demo-antalya-1:8123/?query=select+version()'
26.3.13.20001.altinityantalya
```

Use open source components to build your apps

Event streaming

- Apache Kafka
- Apache Pulsar
- Vectorized Redpanda

ETL

- Apache Airflow
- Apache Nifi
- Rudderstack

Rendering/Display

- Apache Superset
- Cube.js
- Grafana

Client Libraries

- C++ - ClickHouse CPP
- Golang - ClickHouse Go
- Java - ClickHouse JDBC
- Javascript/Node.js - Apla
- ODBC - ODBC Driver for ClickHouse
- Python - ClickHouse Driver, ClickHouse SQLAlchemy

More client library links HERE

Kubernetes

- Altinity Operator for ClickHouse

With AI agents you can now skip to full apps quickly

```
❯ Hi Claude, please do the following for me.
1. Create a directory called 'demo'.
2. Create a docker compose application consisting of the following
services:
  - An Altinity Antalya server (use highest available version).
  - An Ubuntu client container with clickHouse-client and curl.
3. Write a README.md file that describes how to start the
application, connect to the client container, and use
clickhouse-client to connect to the Antalya server.

o I'll check Docker Hub for the latest Antalya build first.
. . .
```

How much can we save?

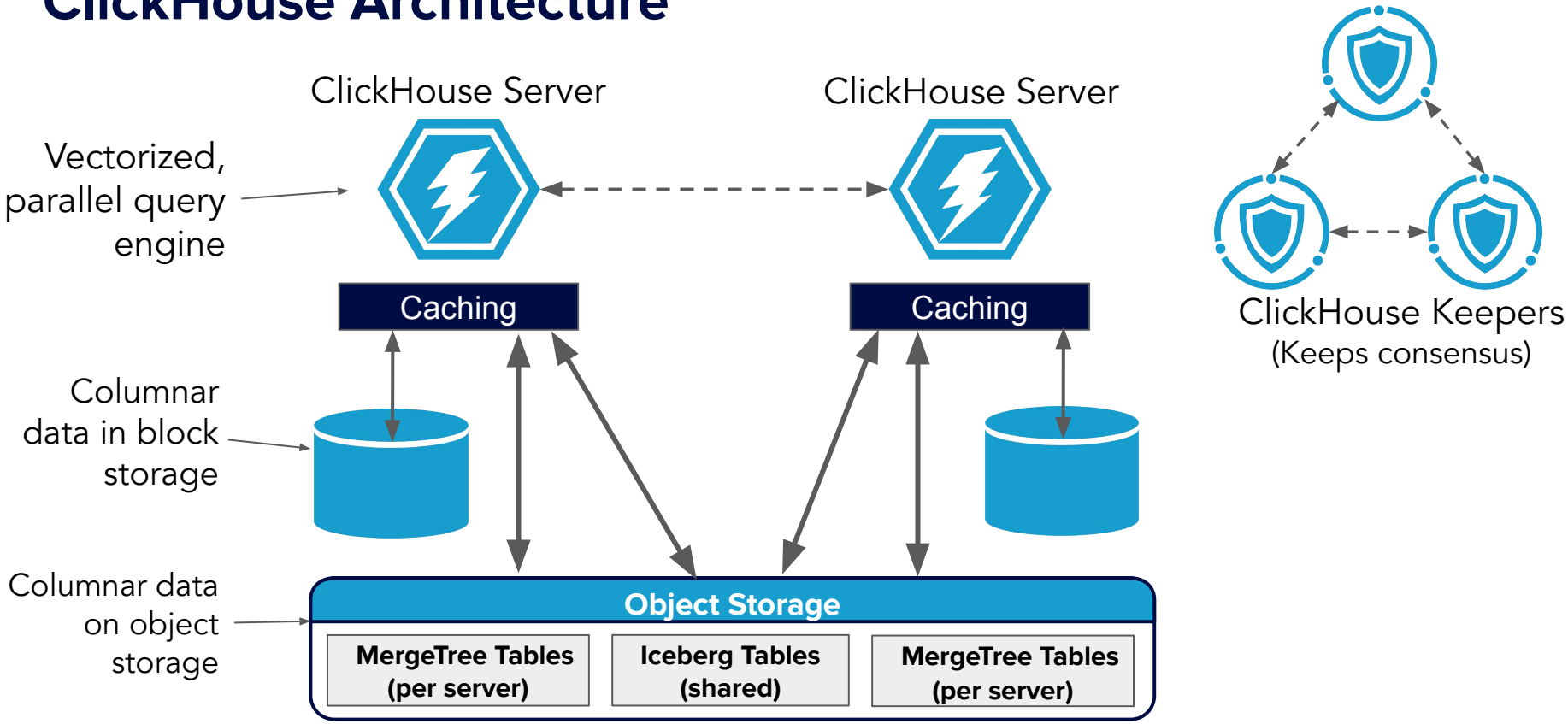
Develop on a laptop with 100% open source + agents

Savings on Licensing: 100% (Plus savings on labor)

Optimize MergeTree tables

© 2025 Altinity, Inc.

ClickHouse Architecture



Optimize schema design to reduce storage and I/O

```
CREATE TABLE IF NOT EXISTS readings_zstd (
  sensor_id Int32 Codec(DoubleDelta, ZSTD(1)),
  sensor_type UInt16 Codec(ZSTD(1)),
  location LowCardinality(String) Codec(ZSTD(1)),
  time DateTime Codec(DoubleDelta, ZSTD(1)),
  date ALIAS toDate(time),
  temperature Decimal(5,2) Codec(T64, ZSTD(10))
)
Engine = MergeTree
PARTITION BY toYYYYMM(time)
ORDER BY (location, sensor_id, time);
```

Optimized data types

Codecs + ZSTD compression

ALIAS column to save space

Time ordering to aid compression

Let's look at an example: IP addresses

```
CREATE TABLE ipdemo.real_string
(
    `ip` String
)
ENGINE = MergeTree
ORDER BY ip
```

Native type

```
IPv4
```

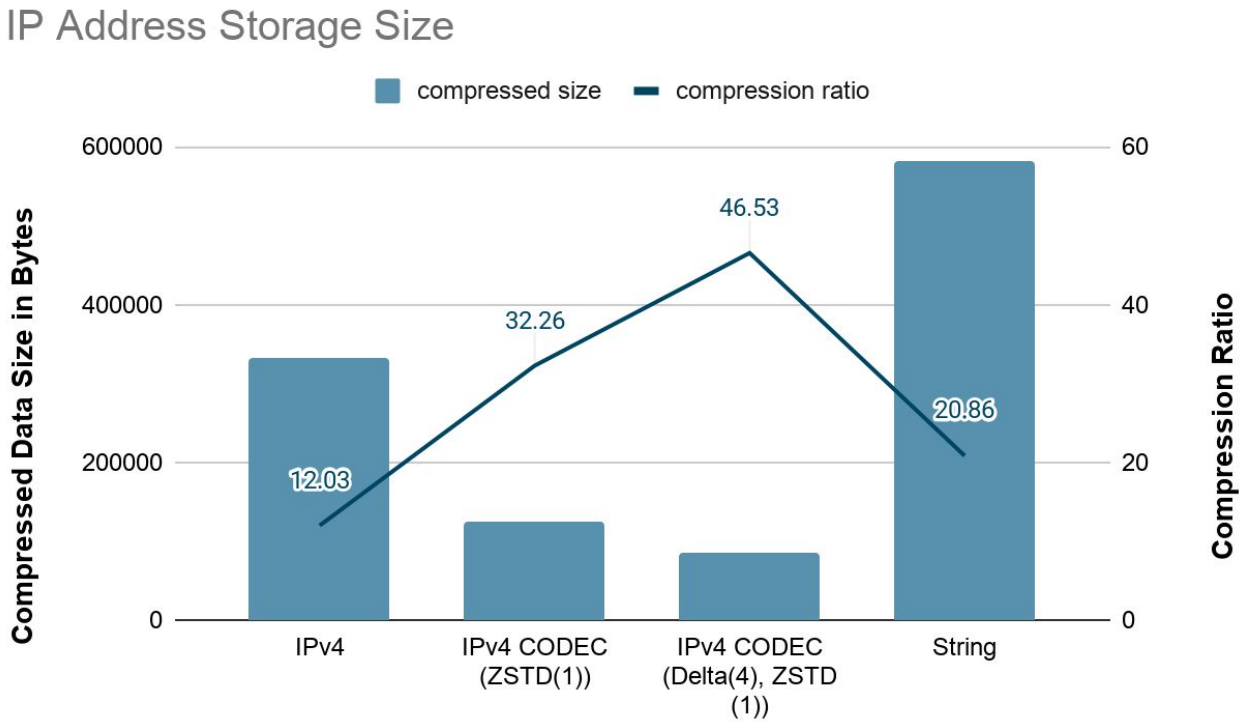
ZSTD compression, not LZ4

```
IPv4 CODEC(ZSTD(1))
```

Delta encoding (stores deltas, not values)

```
IPv4 CODEC(Delta(4), ZSTD(1))
```

On-disk table size for different IP address types



Learn to love ClickHouse system tables

```
SELECT
    table,
    sum(data_uncompressed_bytes) AS uncompressed,
    sum(data_compressed_bytes) AS compressed,
    round(sum(data_compressed_bytes) / sum(rows), 3)
        AS disk_bytes_per_row,
    round(sum(data_uncompressed_bytes) /
sum(data_compressed_bytes), 2) AS ratio
FROM system.parts
WHERE active AND database = 'ipdemo' AND table LIKE 'real_%'
GROUP BY table
ORDER BY table;
```

Don’t stop with schema: make queries more efficient, too

0.84 sec
1.6 KB RAM

```
SELECT Carrier,
  avg(DepDelay)AS Delay
FROM ontime
GROUP BY Carrier
ORDER BY Delay DESC
LIMIT 50
```

Simple aggregate, short GROUP BY key with few values

3.4 sec
2.4 GB RAM

```
SELECT Carrier, FlightDate,
  avg(DepDelay) AS Delay,
  uniqExact(TailNum) AS Aircraft
FROM ontime
GROUP BY Carrier, FlightDate
ORDER BY Delay DESC
LIMIT 50
```

More complex aggregates, longer GROUP BY with more values

System.query_log is another delightful table

```
SELECT
    event_time,
    type,
    query_duration_ms / 1000 AS duration,
    read_rows,
    read_bytes,
    result_rows,
    formatReadableSize(memory_usage) AS memory,
    query
FROM system.query_log
WHERE (user = 'test') AND (type = 'QueryFinish')
ORDER BY event_time DESC
LIMIT 50
```

How much can we save?

Optimize table schema and queries

Savings on storage:
Up to 50%

Use TTLs to limit data growth

© 2025 Altinity, Inc.

TTLs can time out rows

```
CREATE TABLE default.web_events_with_ttl_2 (
    `time` DateTime,
    . . .
    `float_value` Float32
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(time)
ORDER BY (user_id, toStartOfDay(time), session_id, time)
TTL time + INTERVAL 12 MONTH DELETE
```

TTLs can also move, aggregate, and recompress data

```
CREATE TABLE default.web_events_with_ttl_2 (
    `time` DateTime,
    . . .
    `float_value` Float32
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(time)
ORDER BY (user_id, toStartOfDay(time), session_id, time)
TTL time + INTERVAL 1 MONTH RECOMPRESS CODEC (ZSTD(1)),
    time + INTERVAL 6 MONTH RECOMPRESS CODEC (ZSTD(10)),
    time + INTERVAL 12 MONTH DELETE
```

Let's prove it works!

```
SELECT partition, name, rows,
    data_compressed_bytes AS compressed,
    data_uncompressed_bytes AS uncompressed
FROM system.parts
WHERE (table = 'web_events_with_ttl_2') AND active
ORDER BY name DESC
```

partition	name	rows	compressed	uncompressed
202304	202304_1_1_0	50000	613930	1388890
202302	202302_2_2_1	50000	327461	1388890
202208	202208_3_3_1	50000	264054	1388890

How much can we save?

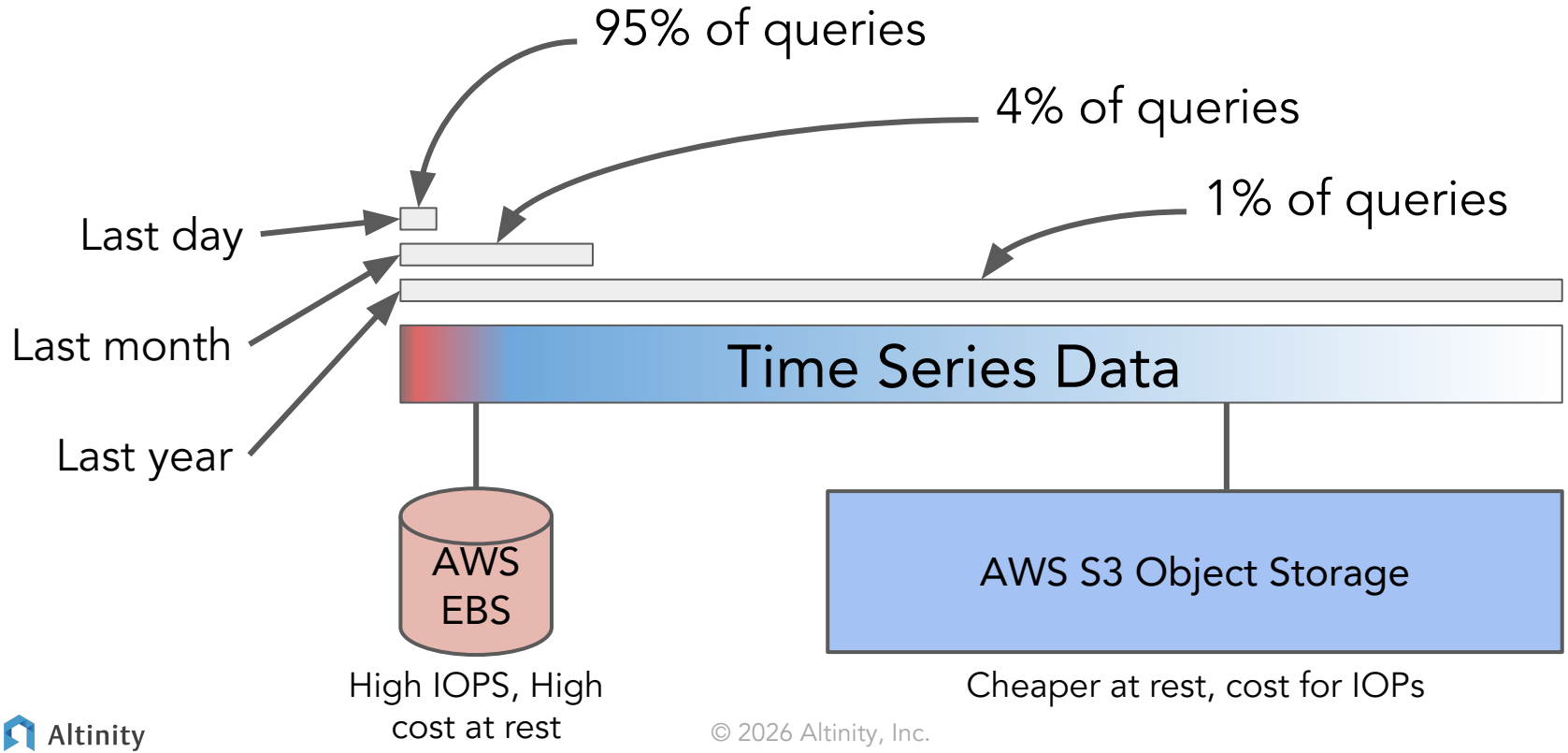
Use TTLs to limit data growth

Savings on Storage:
Up to 20%

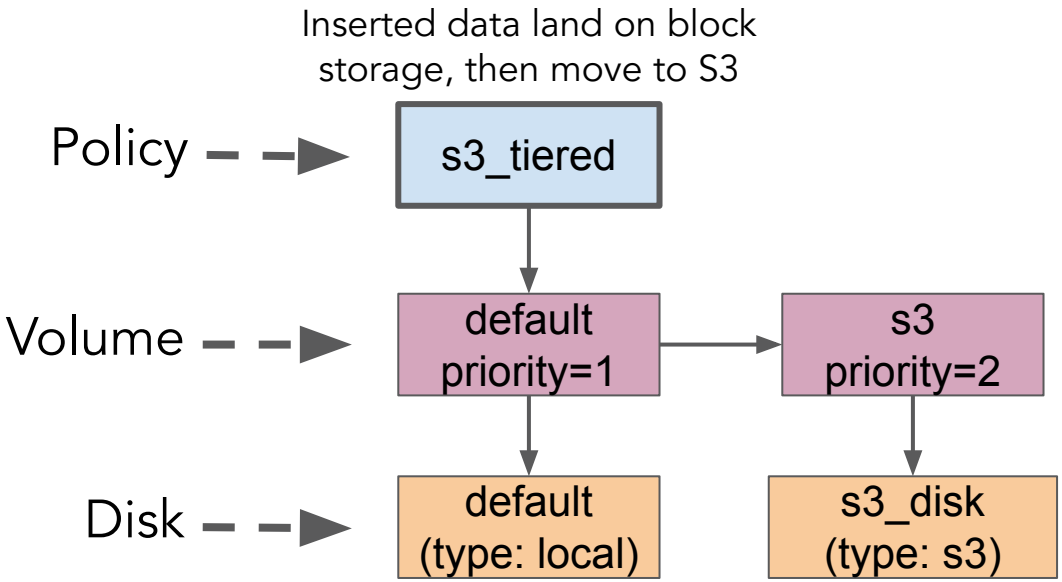
Use tiered storage for older data

© 2025 Altinity, Inc.

Tiered storage matches storage cost to access level



Tiered storage on ClickHouse: policies, volumes, disks



Configuration for tiered storage

```
<clickhouse> <storage_configuration>
  <disks>
    <s3_disk> . . . </s3_disk>
  </disks>
  <policies>
    <s3_tiered>
      <volumes>
          <default> <disk>default</disk>
                <move_factor>0.1</move_factor></default>
          <s3> <disk>s3_disk</disk> </s3>
      </volumes>
    </s3_tiered>
  </policies>
</storage_configuration> </clickhouse>
```

Moving from block storage to S3 using a TTL

```
CREATE TABLE test_s3_tiered(
    `A` Int64,
    `S` String,
    `D` Date
)
ENGINE = MergeTree
PARTITION BY D
ORDER BY A
TTL D + INTERVAL 7 DAY TO VOLUME 's3'
  D + INTERVAL 365 DAY DELETE
SETTINGS storage_policy = 's3_tiered';
```

How much can we save?

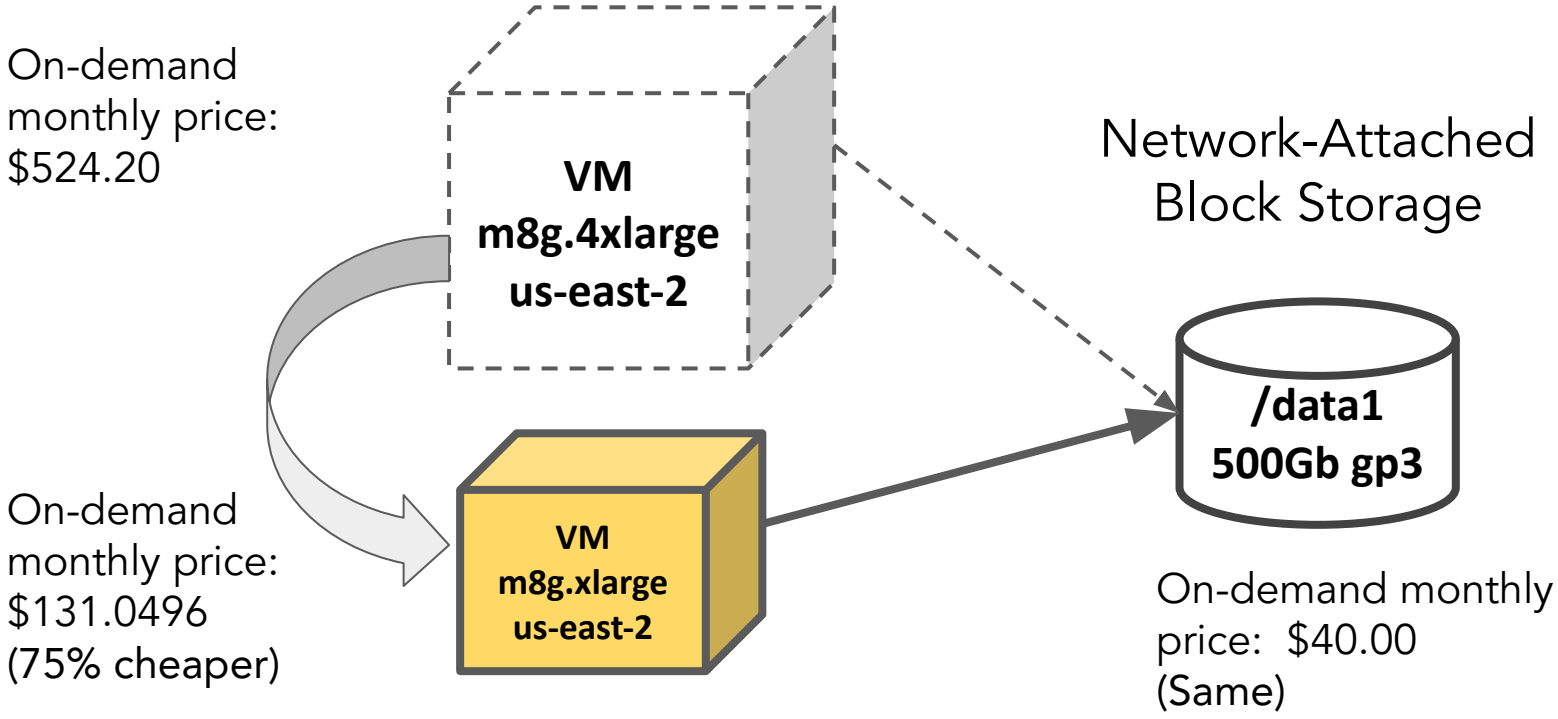
Use tiered storage for older data

Savings on Storage:
Up to 30%

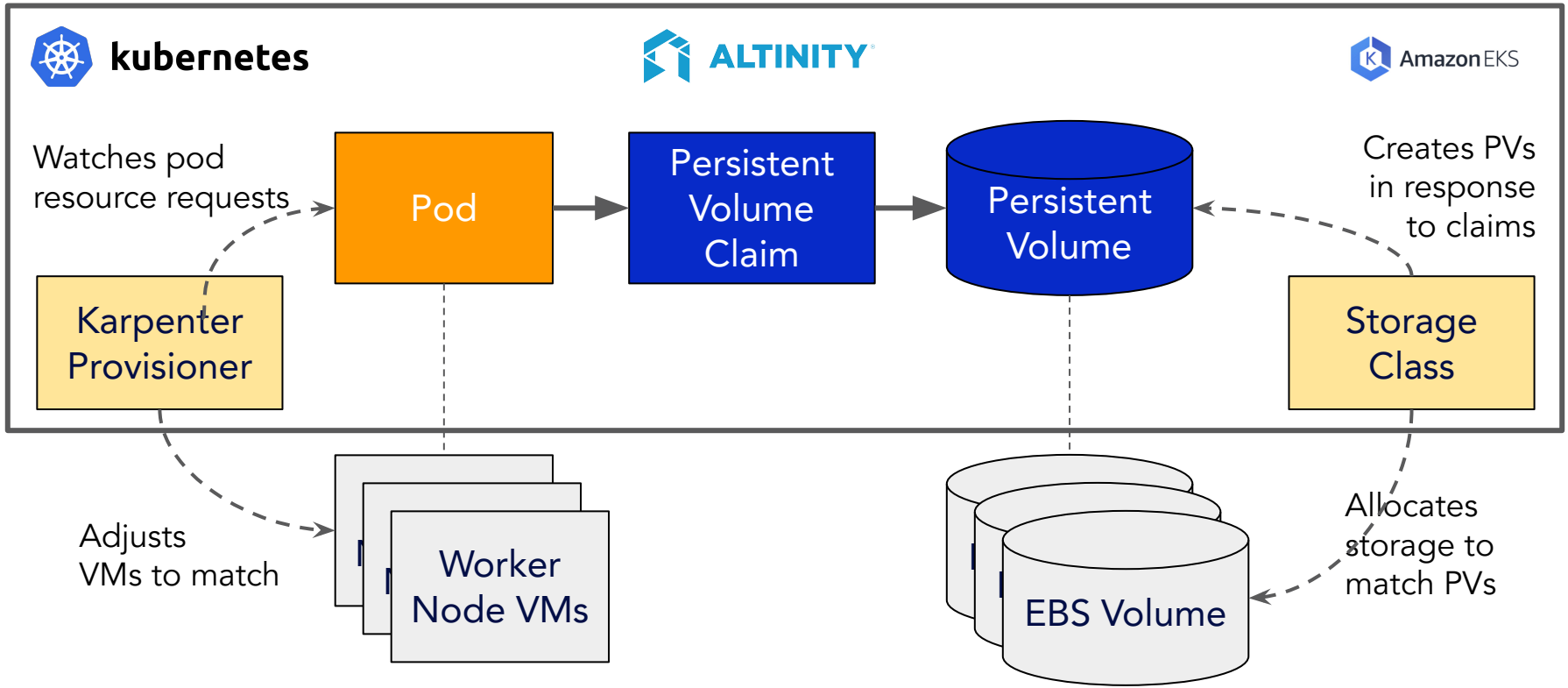
Scale compute capacity down when not needed

© 2025 Altinity, Inc.

Re-scaling compute lowers cloud costs dramatically

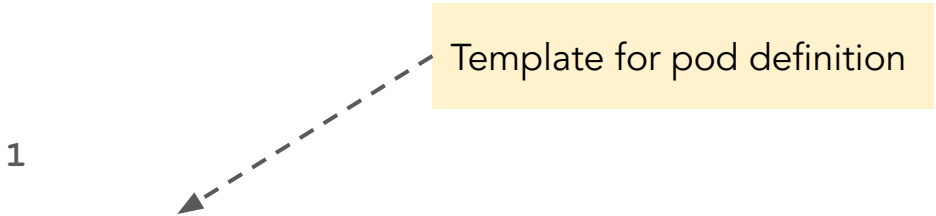


Kubernetes + Altinity operator makes rescaling easy



Use Altinity operator pod templates to set pod properties

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



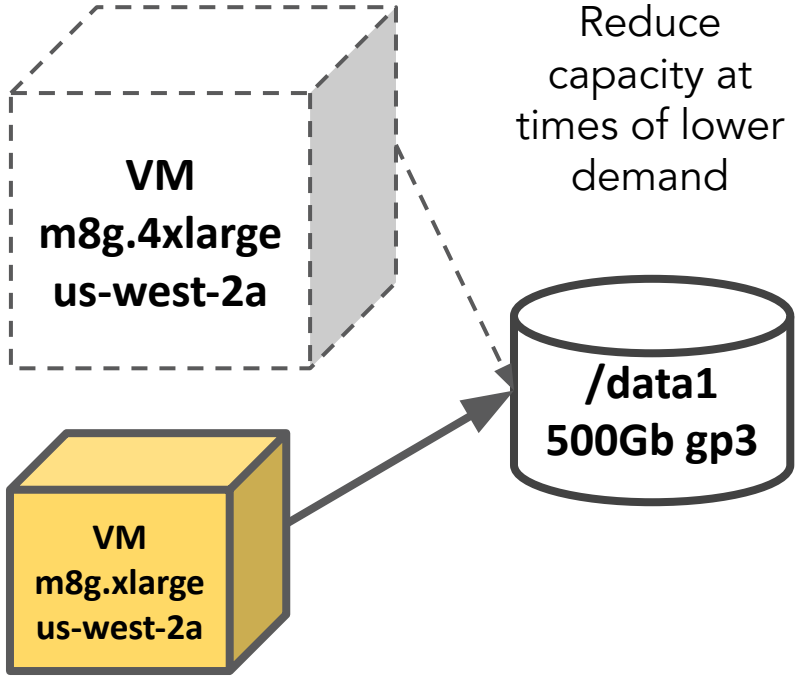
Node selectors and instance types force pods to nodes

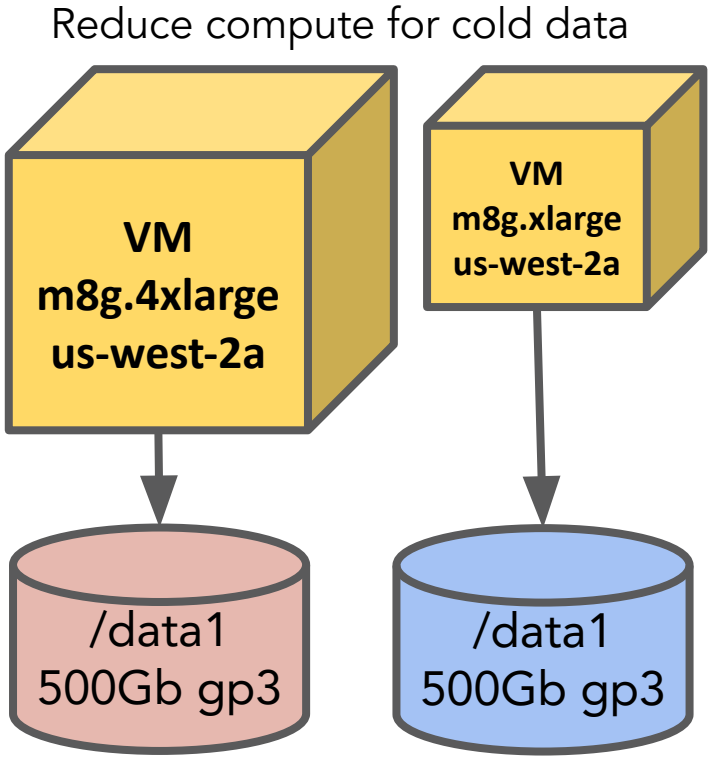
```
podTemplates:
 - name: clickhouse-zone-2a
   spec:
     containers:
     - name: clickhouse
       image: altinity/clickhouse-server:26.3.13.20001.altinityantalya
     nodeSelector:
       node.kubernetes.io/instance-type: m8g.xlarge
   zone:
     key: topology.kubernetes.io/zone
     values:
     - us-west-2a
```

Requires a node with m8g.xlarge VM type

Requires a node in zone us-west-2a

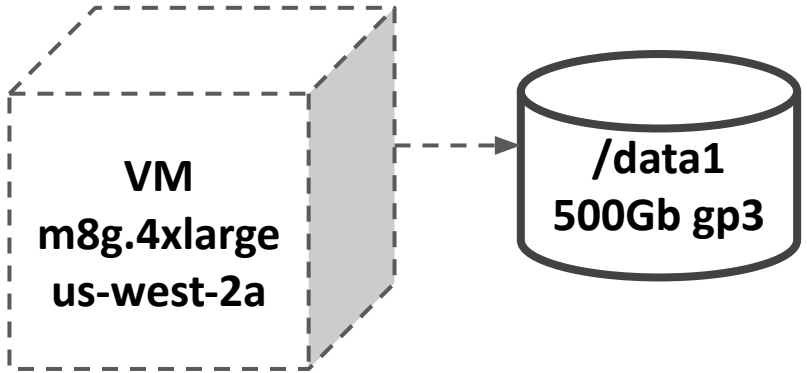
Patterns for using compute rescaling





You can also turn off compute completely!

Turn off idle development hosts



Kubernetes hack: edit replica, set replicas to 0

https://altinity.com/blog/cut-compute-costs-by-scaling-clickhouse-servers-to-zero-on-kubernetes

How much can we save?

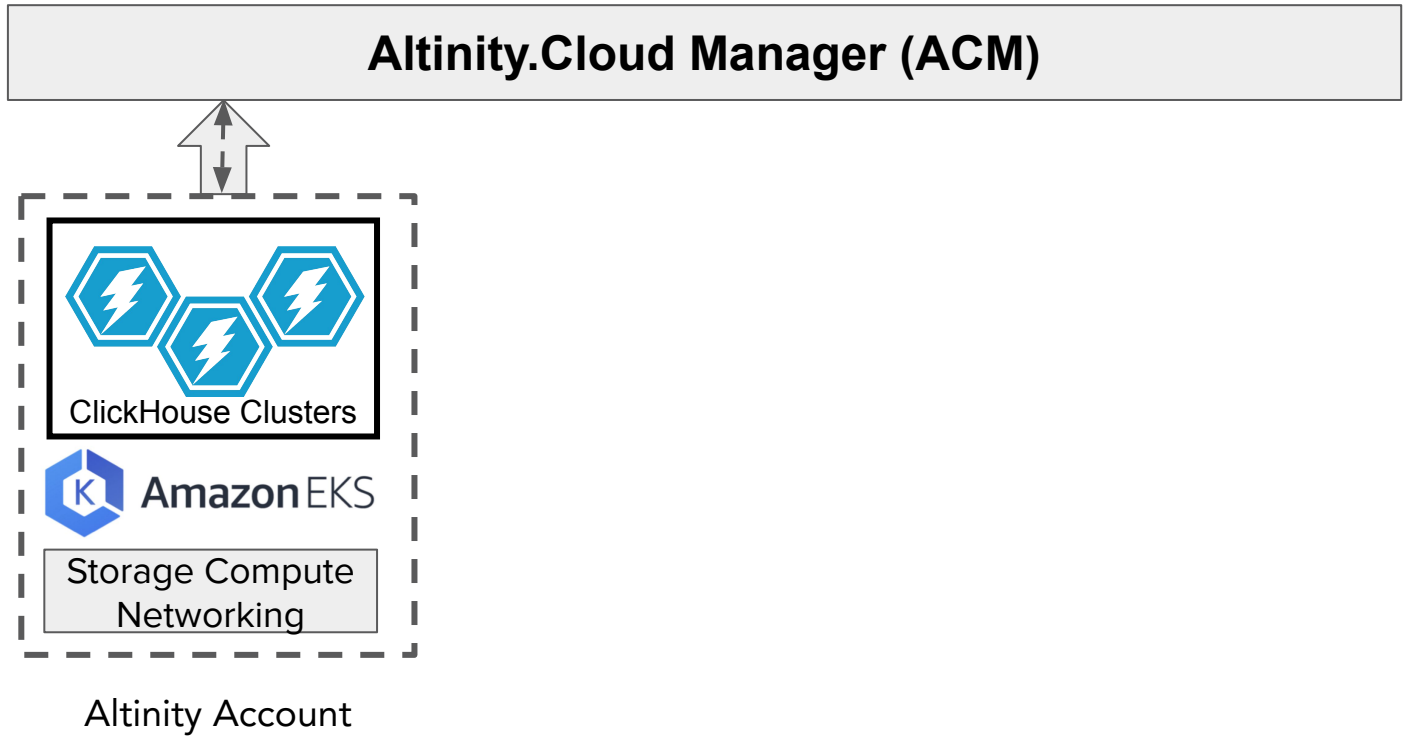
Scale capacity down when not needed

Savings on Compute: Up to 40%

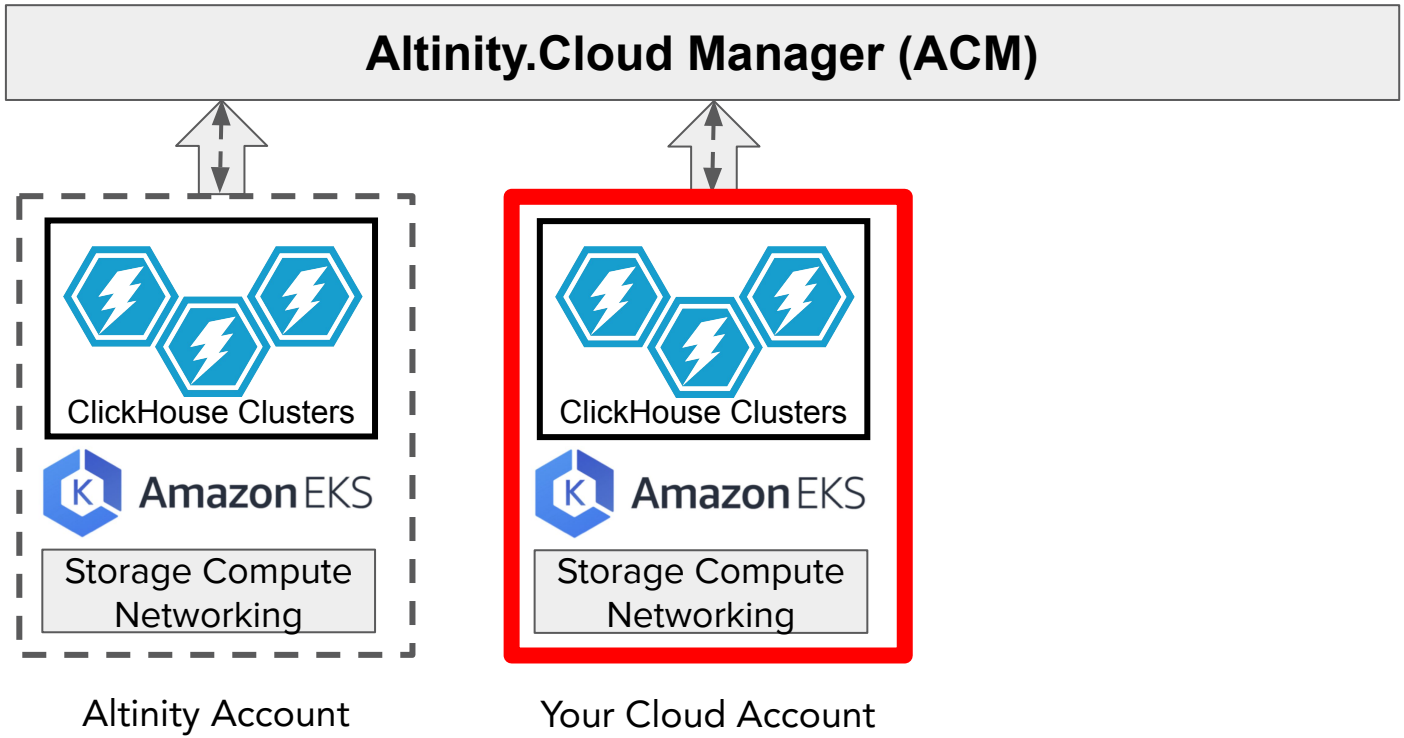
Use BYOC Management

© 2025 Altinity, Inc.

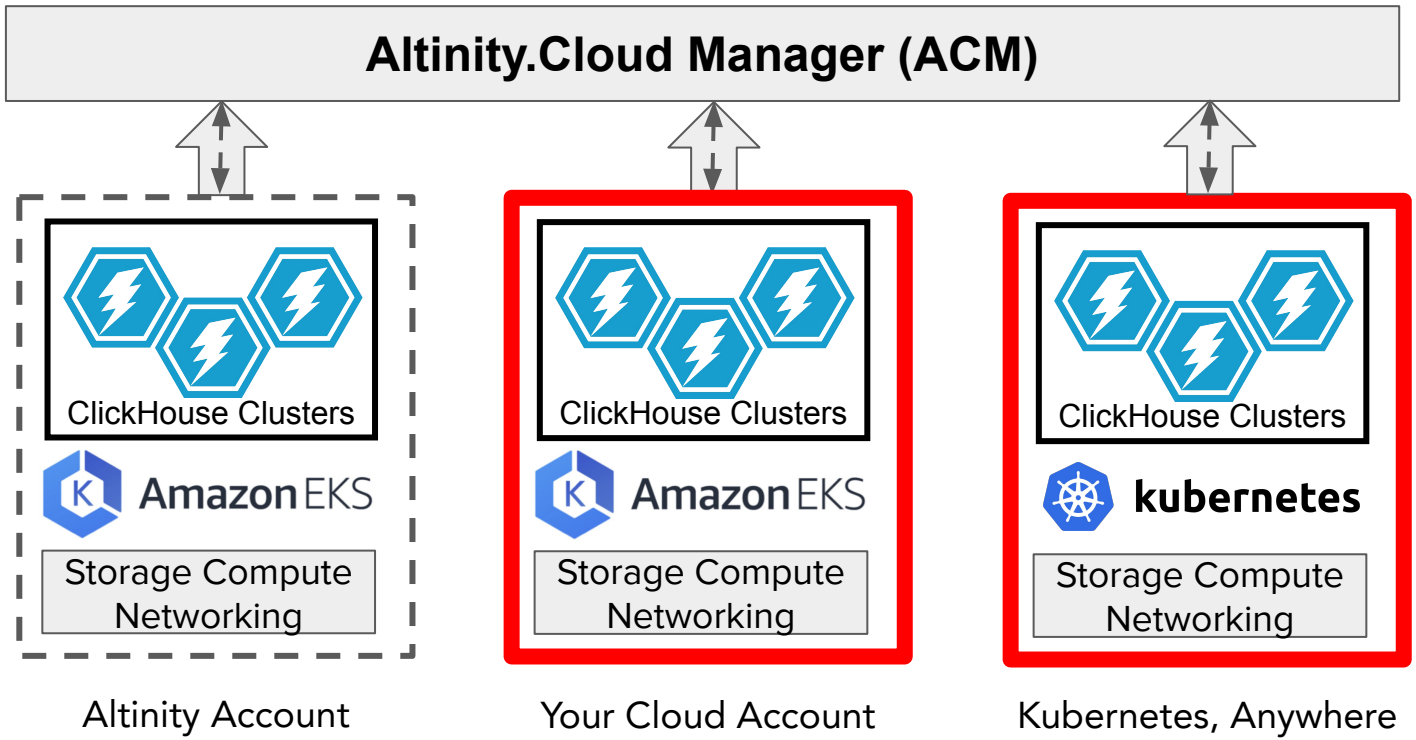
BYOC == Bring Your Own Cloud



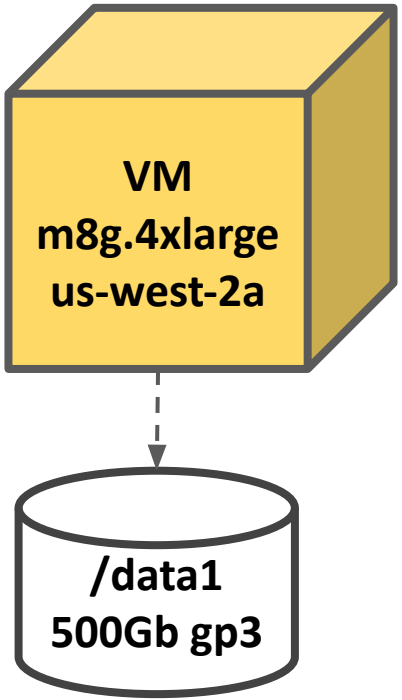
BYOC == Bring Your Own Cloud



BYOC == Bring Your Own Cloud



BYOC Opportunity: Mix & match storage types

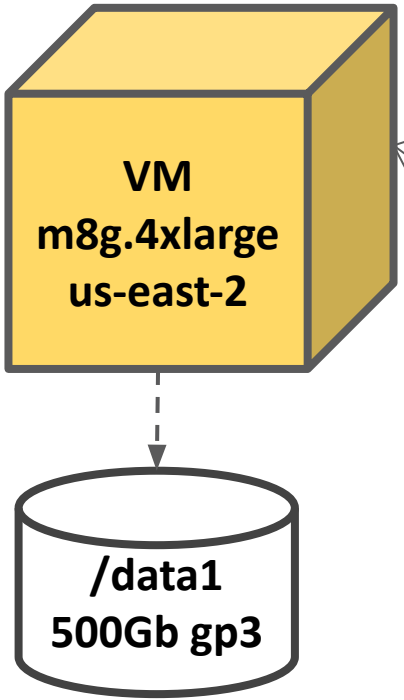


AWS gp3 storage is 20% cheaper than gp2

Operator command to select storage

```
volumeClaimTemplates:
  - name: storage
    # Do not delete PVC
    reclaimPolicy: Retain
    spec:
      storageClassName: gp3
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
```

BYOC opportunity: Savings plans to reduce costs



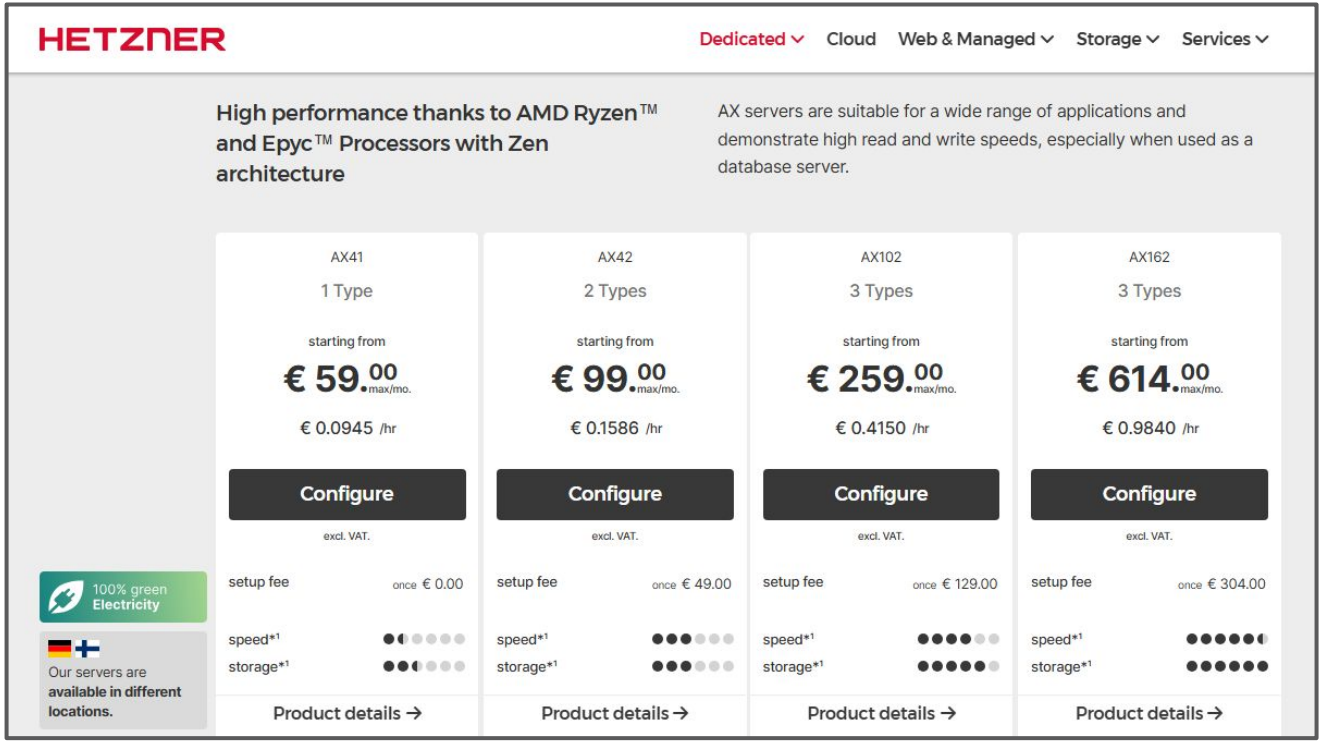
On-demand monthly price:
$524.20

Monthly price for one-year savings plan, no pre-pay:
$385.11 (27% cheaper)

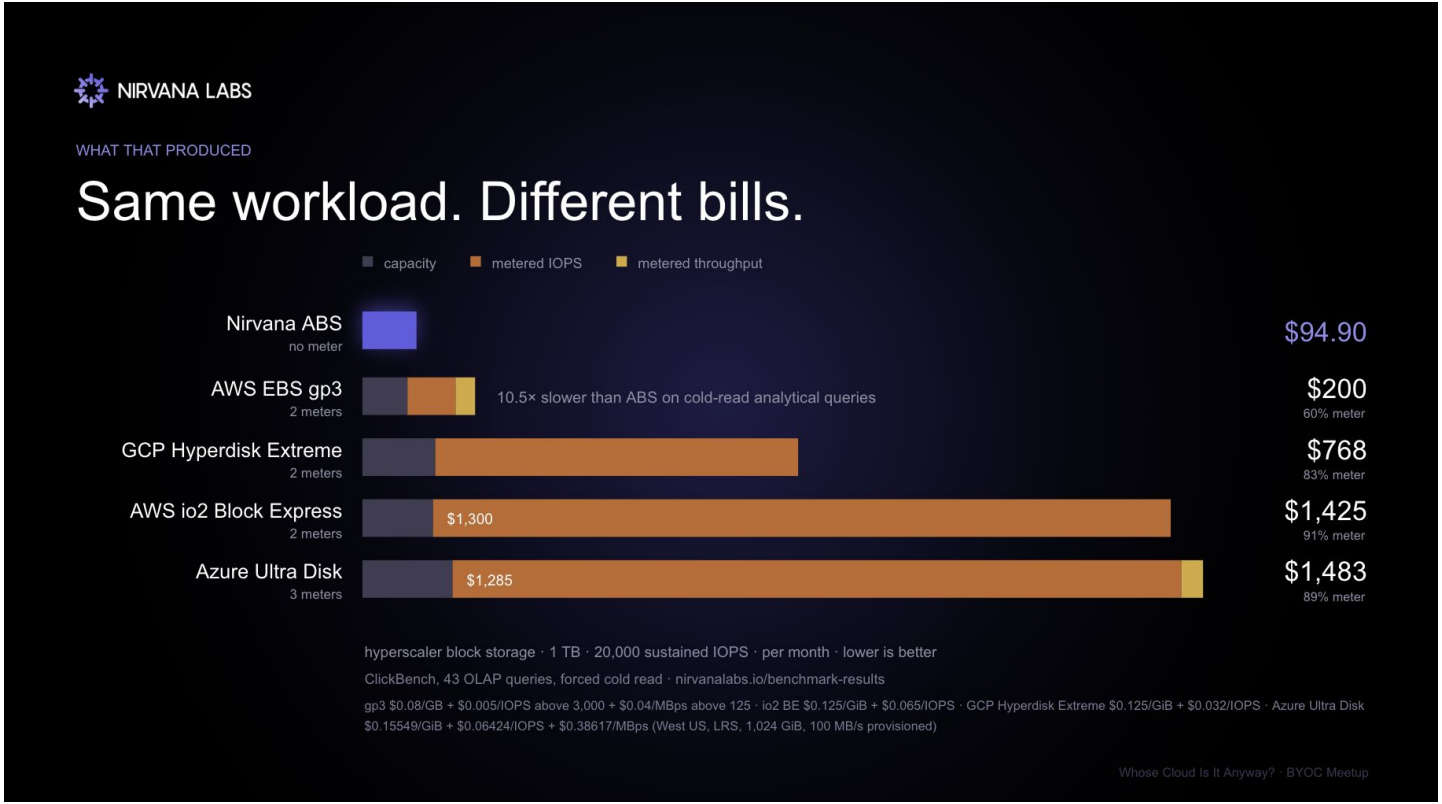
Price for three-year savings plan, full-prepay:
$8684.23
(=$241.23 per month, 54% cheaper)

BYOC Opportunity: Low-cost infrastructure providers

Up to 65% cheaper than AWS EC2 on-demand prices



BYOC opportunity: Specialized high performance clouds



How much can we save?

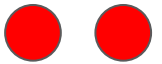
Bring Your Own Cloud (BYOC)

Savings on Infra:
15 to 65%

Conclusion

© 2025 Altinity, Inc.

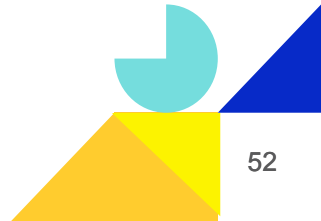
A wallet-sized list of ways to save costs on ClickHouse

Short-term Savings	Time-honored cost-saving practice	Long-term impact
100%	Develop on laptop with 100% open source software	
15% to 65%	Bring Your Own Cloud (BYOC)	
Up to 50%	Optimize MergeTree tables	
Up to 40%	Scale capacity down when not needed	
Up to 30%	Use tiered storage to S3 for older data	
Up to 20%	Use TTLs to limit data growth	

Learn more!

- Altinity Website:
  - https://altinity.com
- Antalya Builds:
  - https://altinity.com/project-antalya-real-time-data-lakes/
- Why BYOC is the New Management Paradigm for Analytic Databases
  - https://altinity.com/blog/why-byoc-for-analytic-databases

Watch out for our new guide: ClickHouse® Cost Efficiency Best Practices



Thank you! Questions?

Altinity BYOC

Robert Hodges

Contact me on LinkedIn

We love open source
and we are hiring!!!

Robert's LinkedIn