Five Things Every New ClickHouse® User Should Know

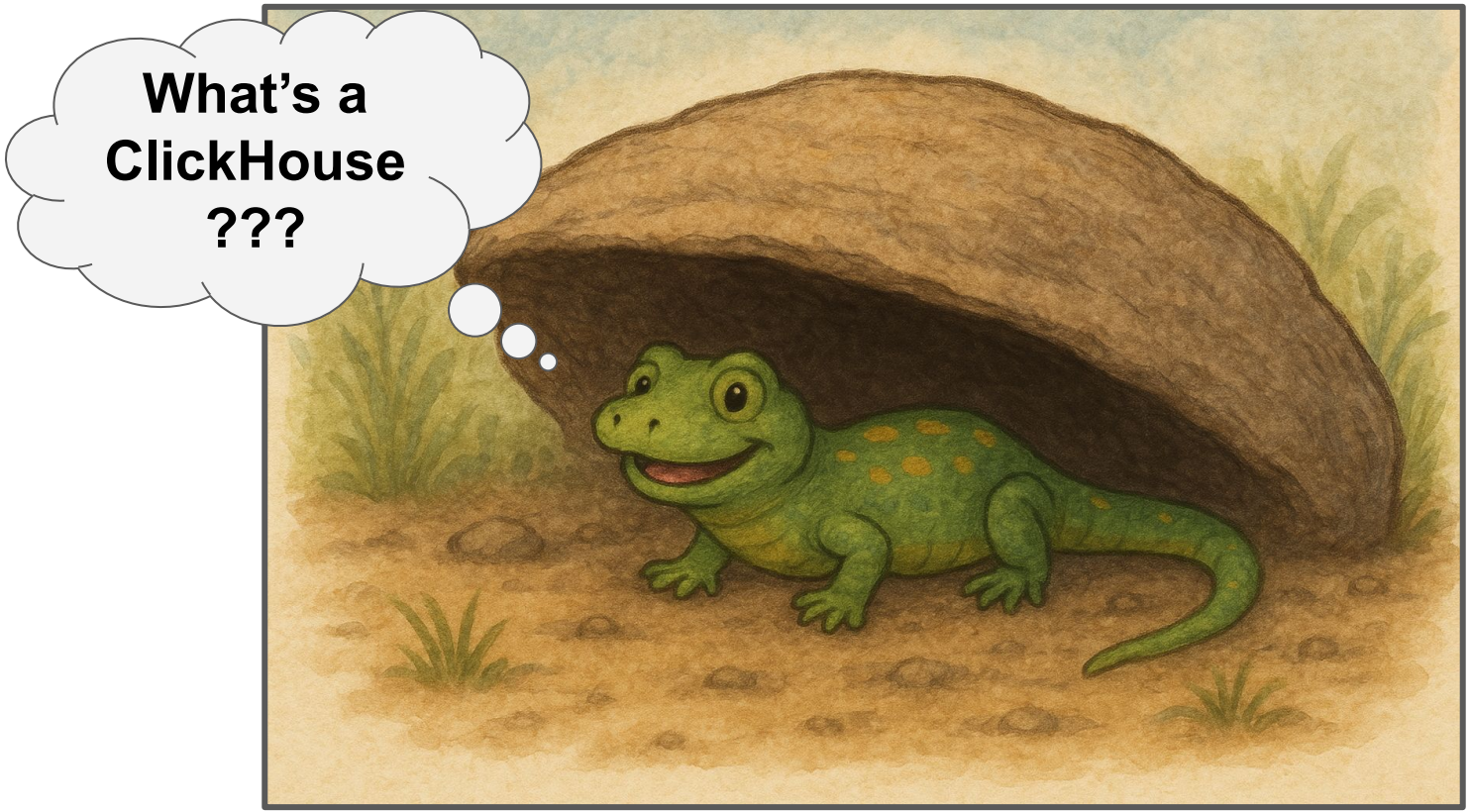
Part 2: Administration

Robert Hodges - Altinity CEO
Tatiana Saltykova - Altinity Support
20 August 2025

ALTINITY®

Run Open Source ClickHouse® Better

Altinity.Cloud **Enterprise Support**



ClickHouse® is a real-time analytic database

Understands SQL

Runs on bare metal to cloud

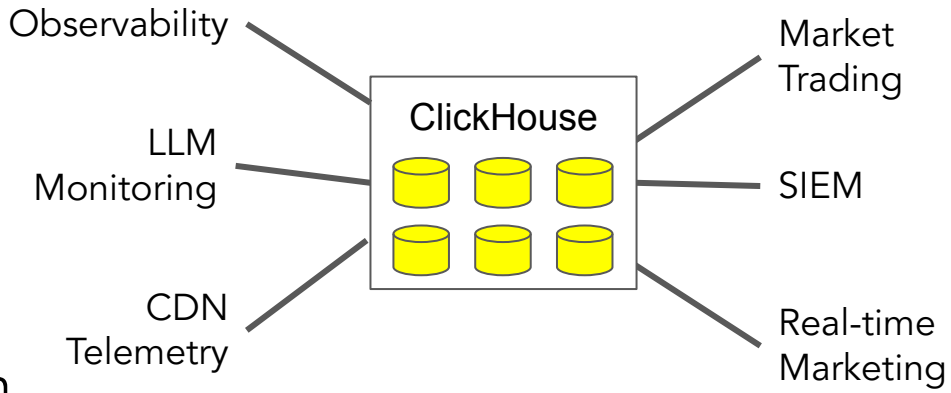
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

Is Open source (Apache 2.0)

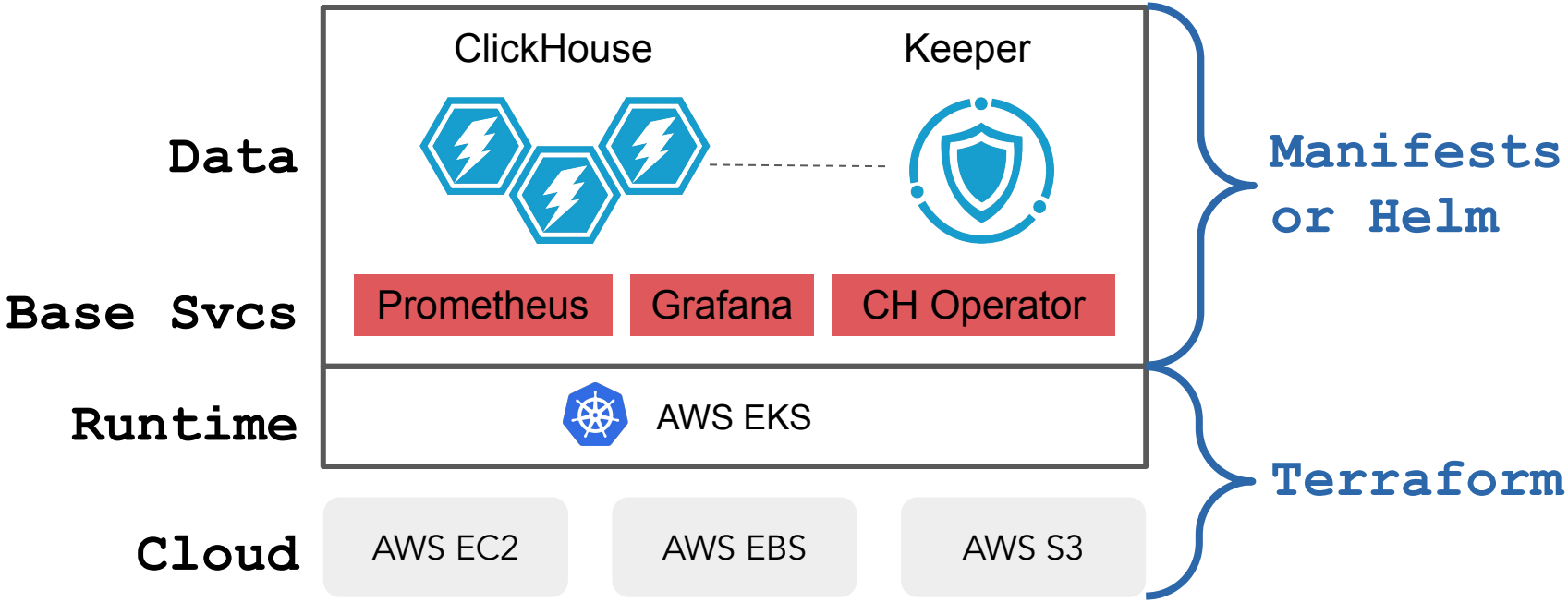


41.8k GitHub Watchers
Can't Be Wrong!

Lesson #1

Pick an environment and understand the ins and outs of operation.

Standard SaaS deployment – ClickHouse on AWS EKS



Bring up ClickHouse on Kubernetes from EKS blueprint

1. Visit blueprint https://github.com/Altinity/terraform-aws-eks-clickhouse
   a. Install prerequisites
   b. Copy example to main.tf

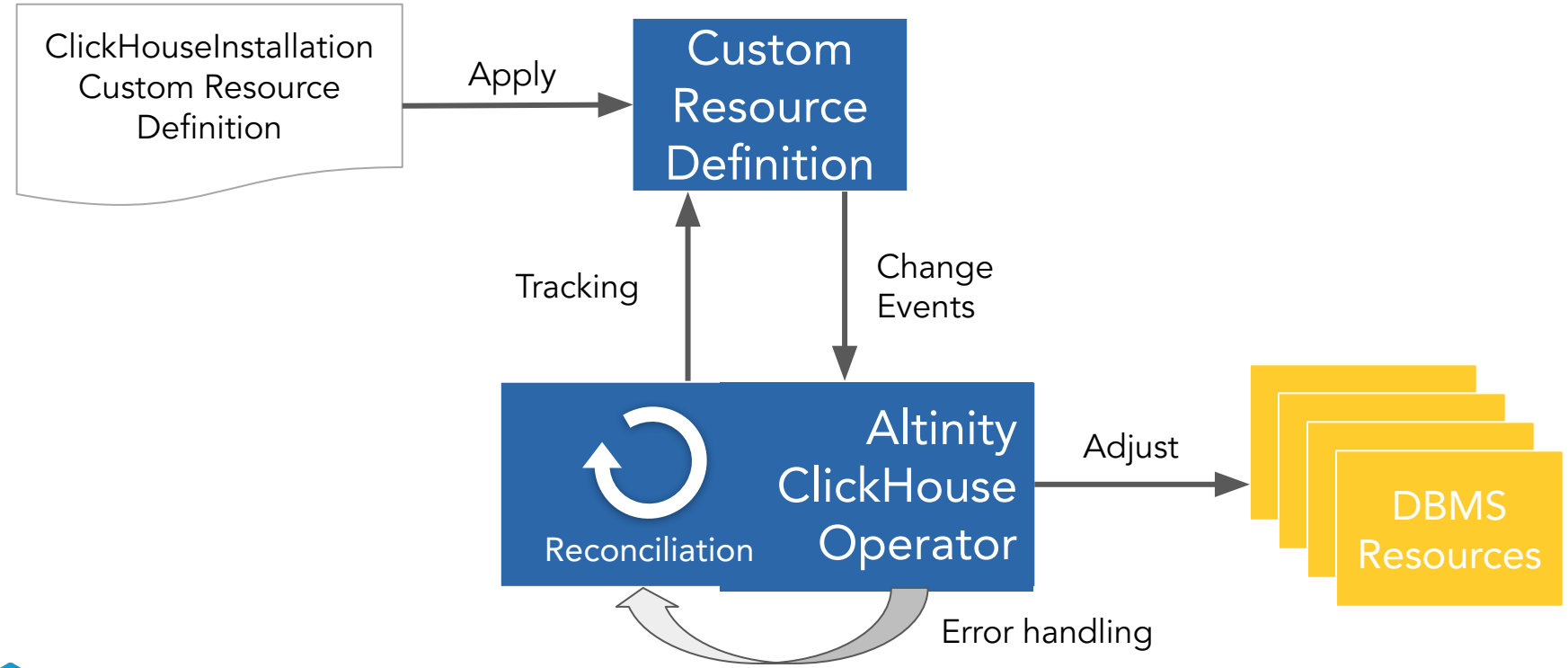
2. Bring up EKS

```
terraform init
terraform apply
```

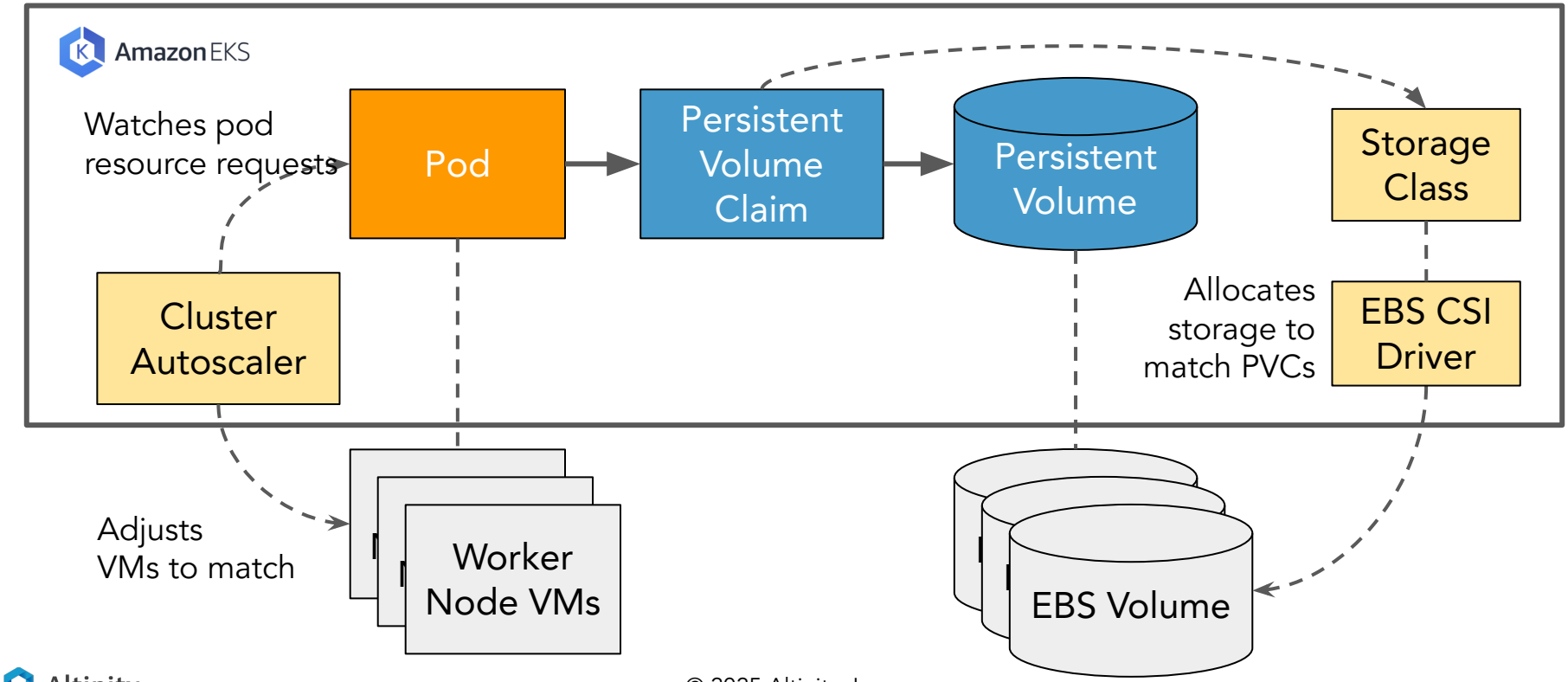
3. Update kubeconfig and start having fun!

```
aws eks update-kubeconfig --region us-east-1 --name my-cluster
kubectl exec -it chi-eks-dev-0-0-0 -n clickhouse -- clickhouse-client
```

Introducing the Altinity Operator for ClickHouse



Behind the curtain: VM and storage allocation on EKS



Simple example of an ClickHouse installation

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "demo"
spec:
  templates:
    podTemplates:
      - name: replica-1
        spec:
          containers:
          - name: clickhouse
            image: altinity/clickhouse-server:24.3.5.47.altinitystable
  configuration:
    clusters:
      - name: "shard1-repl2"
        layout:
. . .
```

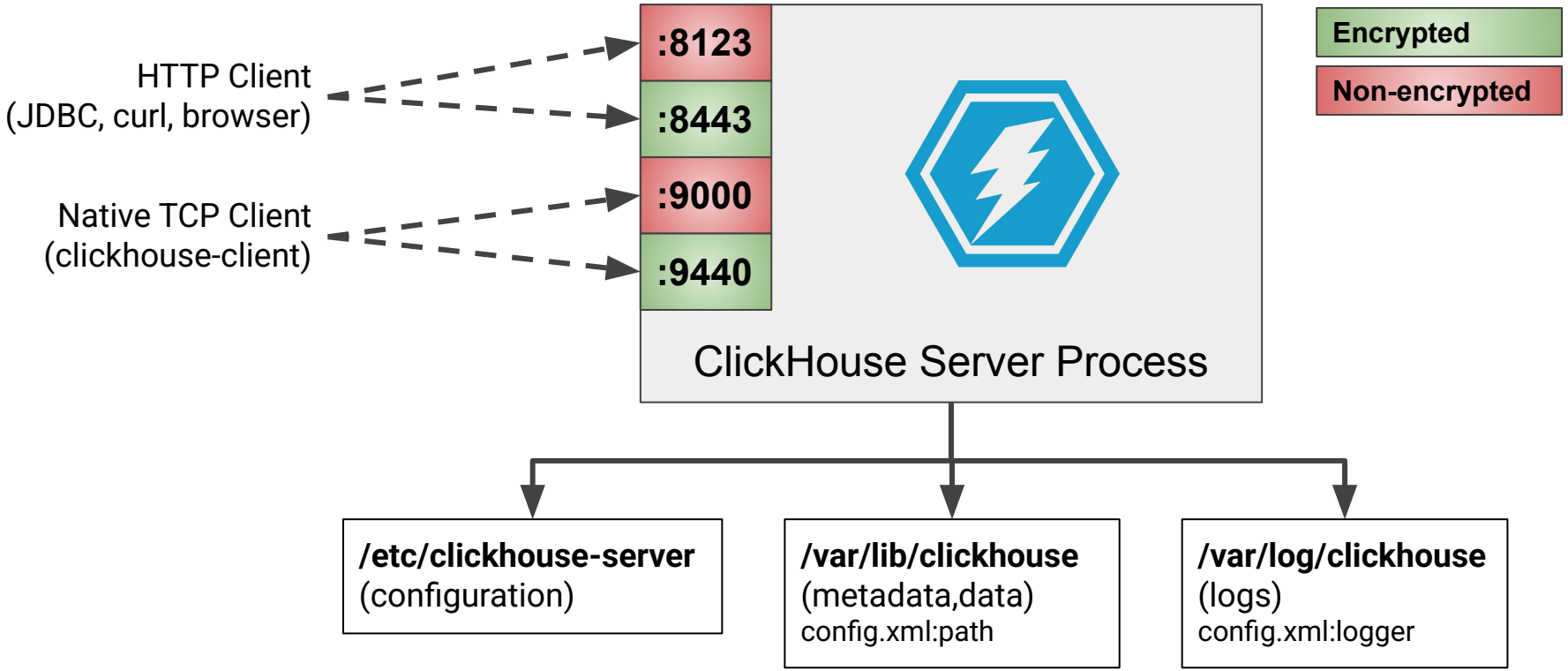
How to install ClickHouse directly on host

Production Deployments: ClickHouse can run on any Linux, FreeBSD, or macOS with x86-64, ARM, or PowerPC64LE CPU architecture

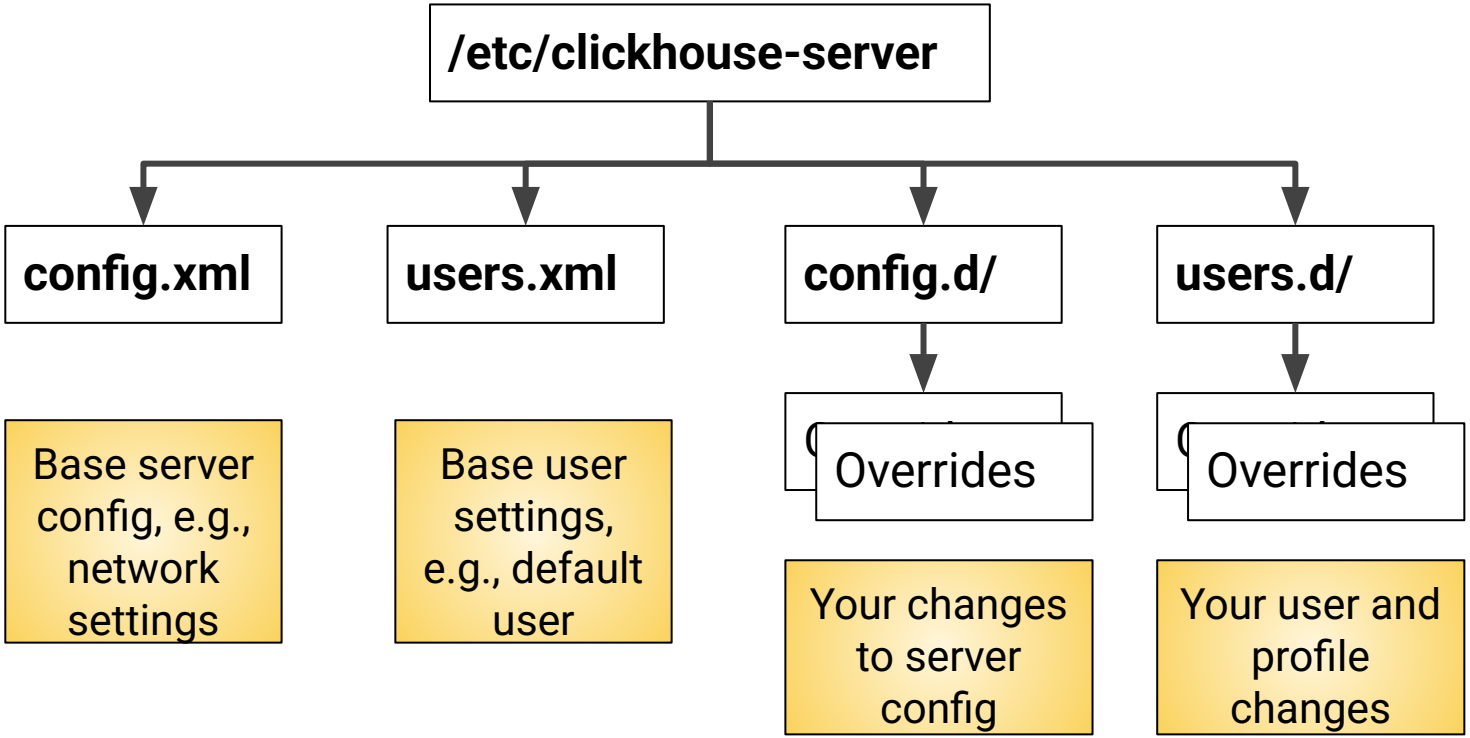
- DEB
- RPM
- Tgz Archives
- Docker Image: use the official Docker image in Docker Hub

Altinity Stable Builds are located at https://builds.altinity.cloud

ClickHouse as a process



Use overrides in config.d and users.d



See the result in /var/lib/clickhouse/preprocessed_configs/

Clouds trade convenience for cost & performance

ClickHouse Cloud
https://clickhouse.com

SaaS version of ClickHouse with Snowflake-like convenience and built-in tools.

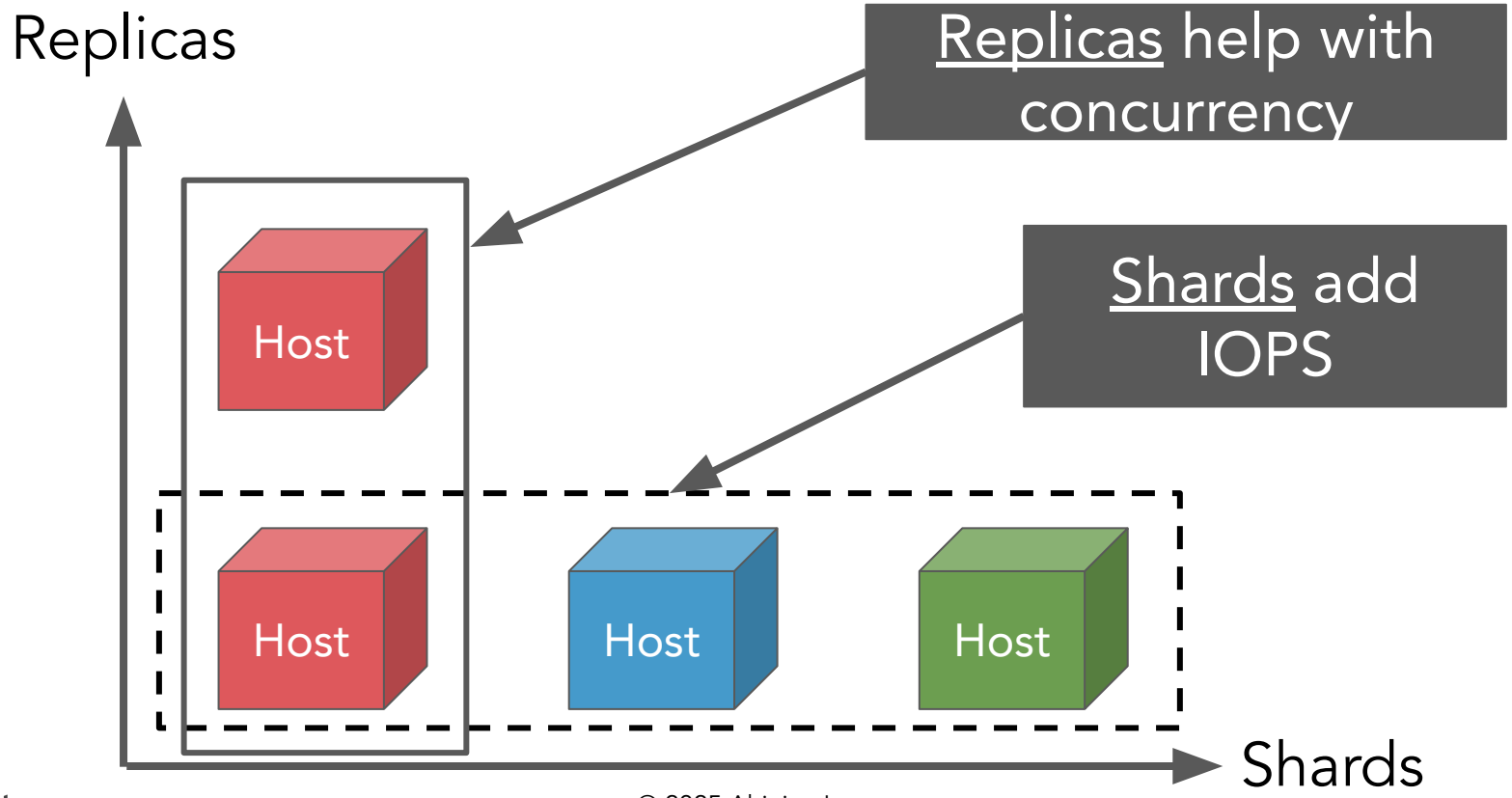
Altinity.Cloud
https://altinity.com

Cloud platform with SaaS and BYOC models. Runs any version of open source ClickHouse.

Lesson #2

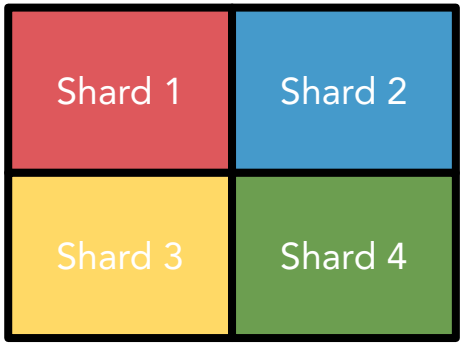
Understand how replication works and how to manage it

Clusters introduce horizontal scaling



Different sharding and replication patterns

All Sharded



Data sharded 4 ways without replication

All Replicated

Data replicated 4 times without sharding

Sharded and Replicated

Data sharded 2 ways and replicated 2 times

What is a replicated table?

```
CREATE TABLE table_name (...)
Engine = ReplicatedMergeTree('zPath','{replica}')
PARTITION BY …
```

Sharded and Replicated pattern

```
ReplicatedMergeTree('/clickhouse/tables/shard1/ontime','replica1')
ReplicatedMergeTree('/clickhouse/tables/shard1/ontime','replica2')
ReplicatedMergeTree('/clickhouse/tables/shard2/ontime','replica1')
ReplicatedMergeTree('/clickhouse/tables/shard2/ontime','replica2')
```

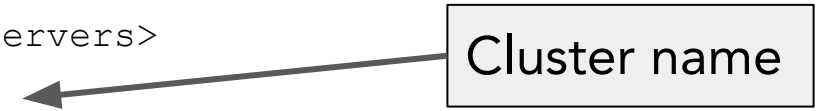
Replicated pattern

```
ReplicatedMergeTree('/clickhouse/tables/red/ontime','host1')
ReplicatedMergeTree('/clickhouse/tables/red/ontime','host2')
ReplicatedMergeTree('/clickhouse/tables/red/ontime','host3')
ReplicatedMergeTree('/clickhouse/tables/red/ontime','host4')
```

What is a cluster?

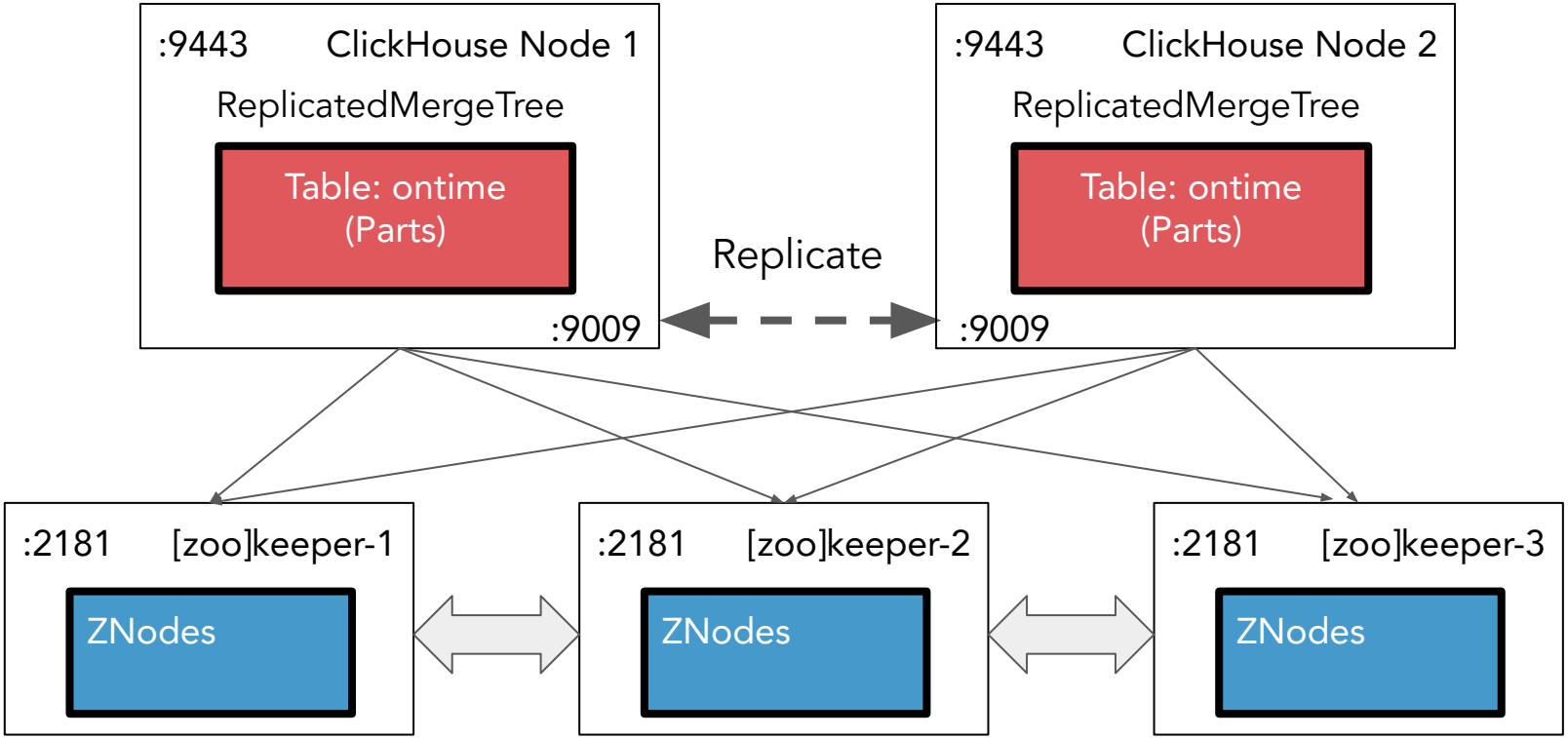
/etc/clickhouse-server/config.d/remote_servers.xml:

```
<clickhouse>
  <remote_servers>
    <demo>
      <shard>
        <replica><host>10.0.0.71</host><port>9000</port></replica>
        <replica><host>10.0.0.72</host><port>9000</port></replica>
        <internal_replication> true</internal_replication>
      </shard>
      <shard>
        . . .
      </shard>
    </demo>
  </remote_servers>
</clickhouse>
```



"It's a cluster because I said so!"

How replication works



Keeper vs. ZooKeeper

ZooKeeper	Keeper
Independent Apache Project (Java)	Bundled with ClickHouse server (C++)
“Extremely stable” - Minimal development	“Very stable” - Adding new features
Supports cross-site promotion of nodes (DR)	Node promotion / ensemble changes may have problems
Does not support some ClickHouse features	S3Queue requires Keeper
No FIPS support	FIPS version available (from Altinity)
If you have it in prod now, keep it!	**Use for new deployments!**

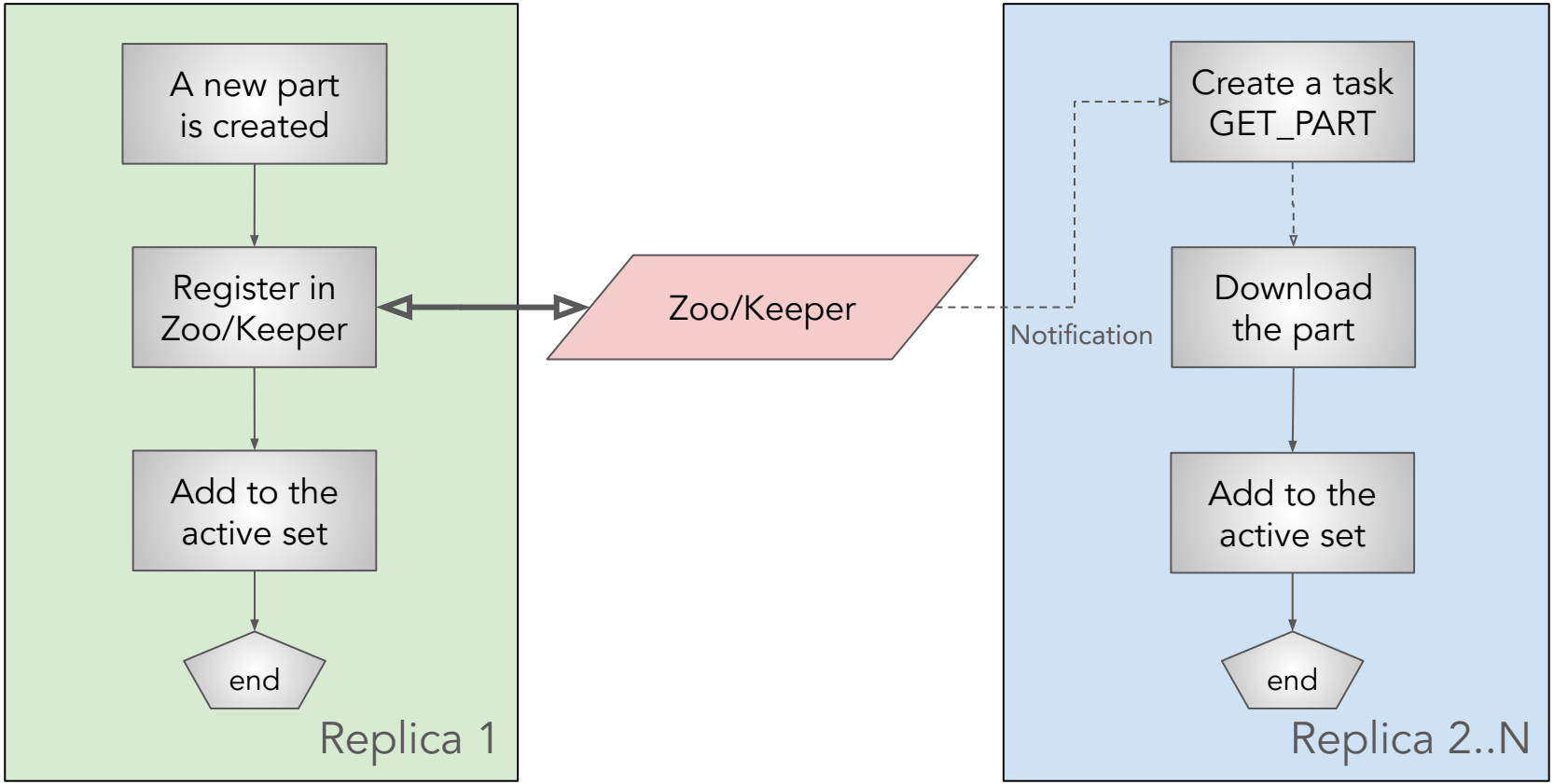
Configure connection to Zoo/Keeper

```
/etc/clickhouse-server/config.d/zookeeper.xml:
<clickhouse><zookeeper>
    <node>
        <host>example1</host>
        <port>2181</port>
    </node>
    <node>
        <host>example2</host>
        <port>2181</port>
    </node>
    <!-- Optional. Chroot suffix. Should exist. -->
    <root>/path/to/zookeeper/node</root>
    <zookeeper_load_balancing>random</zookeeper_load_balancing>
</zookeeper></clickhouse>
```

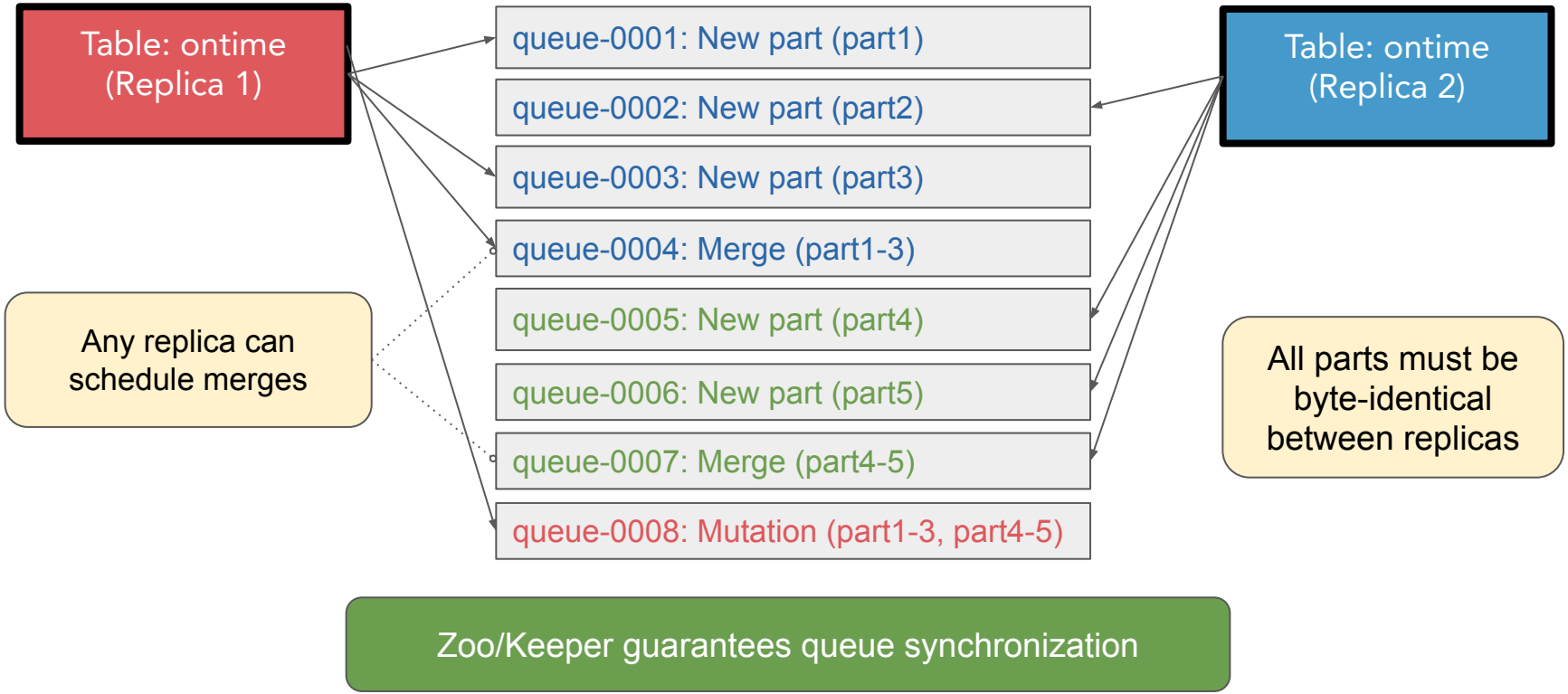
Check connectivity: **`select * from system.zookeeper where path='/'`**

https://clickhouse.com/docs/en/operations/server-configuration-parameters/settings#server-settings_zookeeper

INSERT INTO a ReplicatedMergeTree table



Replication is asynchronous but sequential



Replication queue

```
select * from system.replication_queue limit 1 format Vertical

Row 1:
──────
database:                default
table:                   my_metric_log
replica_name:            chi-github-github-0-0
position:                4
node_name:               queue-0002314922
type:                    MERGE_PARTS
create_time:             2023-07-22 16:26:33
required_quorum:         0
source replica:          chi-github-github-0-1
new_part_name:           20230722_4038_4041_1
. . .
```

https://clickhouse.com/docs/en/operations/system-tables/replication_queue

Lesson #3

ClickHouse schema is easy to change except when it isn’t

Things you can't change easily in MergeTree tables

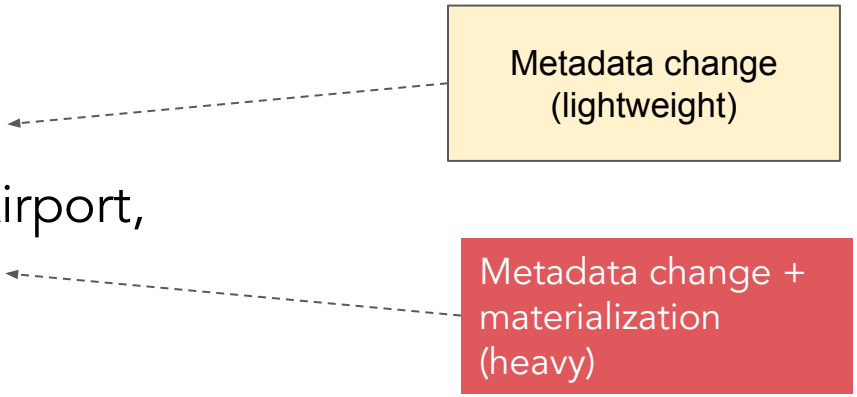
```
CREATE TABLE default.ontime_ref(
    `Year` UInt16,
    `Quarter` UInt8,
    `Month` UInt8,
    `FlightDate` Date,
    `Carrier` FixedString(2),
    . . .
)
ENGINE = MergeTree
PARTITION BY Year
ORDER BY (Carrier, FlightDate)
```

Cannot be changed

Cannot be changed except when adding new columns

Schema changes can be very lightweight or heavy

```
ALTER TABLE ontime_ref
  ADD COLUMN DestTemp Float32,
  RENAME COLUMN Dest TO DestAirport,
  MODIFY COLUMN DestWac Int16,
  DROP COLUMN DestCity,
  MODIFY SETTING storage_policy='policy1',
  RESET SETTING ttl_only_drop_parts;
```



Schema changes that alter data generate mutations

Mutations can be very heavy.
However they simply CLONE the parts that don't need to be changed.

- Mutate all columns' files
  - ALTER TABLE … DELETE
  - ALTER TABLE … MATERIALIZE TTL
- Mutate one or several columns' files
  - ALTER TABLE … UPDATE
  - ALTER TABLE … MATERIALIZE COLUMN
  - ALTER TABLE … MODIFY COLUMN (change data type)
  - DELETE FROM … (lightweight delete)
- Drop old and/or create new files
  - ALTER TABLE … MATERIALIZE/DROP INDEX/PROJECTION
  - ALTER TABLE … DROP/RENAME/CLEAR COLUMN

Mutations are always applied to the whole table.

Lesson #4

Learn the philosophies and best practices for upgrade

Two different upgrade strategies

Leading Edge

Upgrade quickly to new ClickHouse releases
Be prepared to revert even faster

Trailing Edge

Upgrade at longer intervals to stable LTS versions
Test very carefully so that upgrade always succeeds

ClickHouse Release Schedule

A new release every month: gets bug fixes for three months

A new LTS (long term support) in March and August: gets bug fixes for one year

How to Choose Between ClickHouse Releases?

Examples:	The release branch is closed:
v23.7.5.30-stable	When 23.10 is released
v23.8.3.48-lts	When 24.8 is released
v23.9.1.1854-stable	When 23.12 is released

Altinity Stable Build schedule

A new Stable Build for each upstream LTS release; gets bug fixes for 3 years

Stable Builds appear ~3 months after upstream LTS

Release notes incorporate Altinity support feedback

Examples:	Release Date	Upstream EOL	Stable Build EOL
v23.8	27 Dec 2023	31 Aug 2024	27 Dec 2026
v24.3	23 Jul 2024	31 Mar 2025	23 Jul 2027
v24.8	31 Jan 2025	31 Aug 2025	31 Jan 2028

Commands to upgrade ClickHouse

```
# Ubuntu example

# Get the latest stable build:
sudo apt upgrade clickhouse-server clickhouse-client

# Get a specific release:
version=22.6.3.35
sudo apt install clickhouse-client=$version \
                 clickhouse-server=$version \
                 clickhouse-common-static=$version

sudo systemctl restart clickhouse-server
```

Official guidelines and step by step plan
Altinity Step-by-step guide to upgrading ClickHouse®

Upgrade Plan

1. Pick a target release level
2. Carefully read the release notes/changelog(s)
3. Test your applications!!!
4. You may need to change some configuration settings for better compatibility, etc
    - Review the configuration changes that you have in the old cluster
    - Review the configuration changes between your current release and the target release
5. Upgrade staging/development/test systems first to verify all systems are working
6. Make sure your schema/queries work properly
7. Prepare and test downgrade procedures to see if the server can be returned to the previous version
8. Upgrade the production system

Rolling upgrade

Mixing several versions working together in the same cluster may often lead to different degradations. It's not recommended to have a big delay between upgrading different nodes on the same cluster.

1. Start with a "canary" update. Pick one replica of one shard and upgrade it to make sure that the procedure works.
    - Verify that everything works properly. Check for any errors in monitoring / logs / system.errors.
2. If everything is working well, update the rest of the cluster.
    - Update binaries on all nodes; Stop ingestion if possible
    - Switch incoming queries to the 'even' replicas (load balancer or edit 'remote_servers' section)
    - Restart the 'odd' replicas and wait for them to come online
    - Switch incoming queries to the 'odd' replicas
    - Restart the 'even' replicas
    - Restart ingestion

Lesson #5

System tables tell you what ClickHouse is doing, in sickness and health

Handy system tables for monitoring

Metrics:

- system.asynchronous_metrics
  - Values are regularly updated
  - Example: Uptime, NumberOfDatabases, OSInterrupts
- system.metrics
  - Reflects current values
  - Example: Query, OpenFileForRead
- system.events
  - Counters always increase since the start of the instance
  - Example: Query, QueriesWithSubqueries

Automatically exported by Altinity Kubernetes Operator!

I/O investigations: where to look

Numbers of parts and compression levels:

- system.columns
- system.tables
- system.parts

What is going on right now:

- system.processes
- system.merges
- system.moves
- system.replicated_fetches (on the receiving replica)

History/statistics (can be turned off):

- system.part_log stores all events that change the set of parts
- system.query_log stores all queries
  - can be configured (`select * from system.settings where name ilike 'log%'`)
- system.query_thread_log stores all queries' threads
  - grows very fast, it's better to keep it disabled

CPU investigations: where to look

How many processes are running / threads are in use right now:

- system.asynchronous_metrics
- system.metrics
- system.processes
- system.merges
- system.moves
- system.replicated_fetches
- system.distribution_queue
- system.kafka_consumers
- system.stack_trace

History/statistics (can be turned off):

- system.asynchronous_metric_log
- system.metric_log
- system.part_log the number of merges and fetches
- system.query_log / query_thread_log / query_views_log the number of queries
- system.processors_profile_log how much time was spent in each step of a query
- system.trace_log stack traces collected by the sampling query profiler

Example: Check data compression levels

```
SELECT table,
  formatReadableSize(sum(data_compressed_bytes)) tc,
  formatReadableSize(sum(data_uncompressed_bytes)) tu,
  sum(data_compressed_bytes) / sum(data_uncompressed_bytes) as ratio
FROM system.columns
WHERE database = currentDatabase()
GROUP BY table ORDER BY table
```

Example: Investigate resource-heavy queries

```
SELECT
    normalized_query_hash,
    any(query),
    count(),
    sum(ProfileEvents['OSCPUVirtualTimeMicroseconds']) AS
OSCPUVirtualTime
FROM clusterAllReplicas('{cluster}', system.query_log)
WHERE event_time between ...
  AND type in (2,4)
GROUP BY normalized_query_hash
ORDER BY OSCPUVirtualTime DESC
LIMIT 30
FORMAT Vertical
```

Groups similar queries!

Shows one sample

Shows the top of 'metric'-intensive

More complicated example: https://kb.altinity.com/altinity-kb-useful-queries/query_log/

Example: Check S3 stats for a distributed query

```
SELECT hostName() host, k, v
FROM clusterAllReplicas('all', system.query_log)
ARRAY JOIN ProfileEvents.keys AS k, ProfileEvents.values AS v
WHERE
  initial_query_id = '5737ecca-c066-42f8-9cd1-a910a3d1e0b4'
  AND type = 2
AND k ilike '%S3%'
ORDER BY host, k
```

Wrap-up and Questions

Summary of 5 things every beginner should know

- ClickHouse can run anywhere – understand the details of your chosen platform
- Replication is powerful but requires careful management
- ClickHouse handles online schema change well. Materialization adds load
- Upgrade success factors: planning, testing, automation
- System tables in ClickHouse are great. They are your friends

Key sources of information

- ClickHouse docs - https://clickhouse.com/docs
- Altinity docs - https://docs.altinity.com/
- Altinity kb - https://kb.altinity.com/
- Blogs (ClickHouse Inc, Altinity, Tinybird, …)

Beware of LLMs bearing gifts.

Доверяй, но проверяй. (Trust, but verify.)

Thank you! Questions?

Contact us to learn more about Altinity.Cloud and Enterprise Support

https://altinity.com

https://altinity.com/slack

https://altinity.com/clickhouse-training/

We are hiring!!!

ALTINITY