ClickHouse Server Two-Node Cluster Installation Guide

This guide provides step-by-step instructions for installing and configuring ClickHouse Server in a replicated 2-node cluster mode on Ubuntu 24.04 LTS.

Overview

This installation sets up ClickHouse in a distributed cluster with:

- **Primary Node**: Runs primary ClickHouse server with ZooKeeper
- **Secondary Node**: Runs secondary ClickHouse server with ZooKeeper
- **Replication**: Automatic data replication between nodes
- **High Availability**: Continued operation if one node fails
- **Load Distribution**: Queries can be distributed across nodes

Installation Steps Overview

1. **Step 1**: Verify Hadoop cluster is running (prerequisite)
2. **Step 2**: Verify SSH connectivity between nodes (reuses Hadoop setup)
3. **Step 3**: Check port availability and avoid conflicts with Hadoop
4. **Step 4**: Install dependencies and add ClickHouse repository
5. **Step 5**: Install and configure ZooKeeper ensemble
6. **Step 6**: Install ClickHouse server on both nodes
7. **Step 7**: Configure ClickHouse cluster settings
8. **Step 8**: Start and verify ClickHouse cluster
9. **Step 9**: Test replication and distributed queries
10. **Step 10**: Final verification with both systems running

Prerequisites

System Requirements

Primary Node:

- Ubuntu 22.04 LTS or 24.04 LTS
- At least 4GB RAM (8GB recommended)
- 20GB of available disk space for data
- Root or sudo access
- Internet connection for downloading packages

Secondary Node:

- Ubuntu 22.04 LTS or 24.04 LTS
- At least 4GB RAM (8GB recommended)
- 20GB of available disk space for data
- Root or sudo access

- Internet connection for downloading packages

Network Requirements

- Both nodes must be able to communicate via network
- Static IP addresses recommended
- **ClickHouse Ports**: 8123, 9001, 9004, 9005, 9009, 9010 must be open between nodes (Note: 9001 used instead of 9000 to avoid Hadoop conflict)
- **ZooKeeper Ports**: 2181, 2888, 3888 must be open for ZooKeeper
- **Hadoop Port Compatibility**: Ensure ClickHouse ports don't conflict with existing Hadoop ports (8020-8040, 9000, 9870, 8088, 19888)
- **SSH Access**: Passwordless SSH between nodes (Hadoop SSH setup can be reused)
- **Prerequisites**: Verify Hadoop cluster is already working before proceeding with ClickHouse installation

Step 1: Verify Hadoop Cluster is Running (Prerequisite)

Before installing ClickHouse, verify your Hadoop cluster is working properly:

From Hadoop Master Node:

```
# Check Hadoop services
jps

# Expected output should include: NameNode, ResourceManager, DataNode, NodeManager, JobHistoryServer

# Check HDFS cluster status
hdfs dfsadmin -report

# Verify HDFS operations
hdfs dfs -ls /
```

From Hadoop Worker Node:

```
# Check Hadoop services
jps

# Expected output should include: DataNode, NodeManager

# Test connectivity to master
ping -c 2 hadoop-master
```

Why verify Hadoop first? This ensures:

- Hadoop configuration is intact and working
- Network connectivity is already established

- SSH access between nodes is functional
- We won't break existing Hadoop functionality with ClickHouse installation

2. Check Port Availability

On both nodes, check if ClickHouse ports are available:

```
# Check if ClickHouse ports are in use
sudo netstat -tulpn | grep -E '(8123|9004|9005|9009|9010|2181|2888|3888)'

# Check existing Hadoop ports
sudo netstat -tulpn | grep -E '(8020|8040|9000|9870|8088|19888)'
```

Note: If port 9000 is already used by Hadoop, we'll configure ClickHouse to use port 9001 instead.

Step 4: Install Dependencies and Add ClickHouse Repository

1. Install Required Dependencies

On both nodes, run:

```
sudo apt update
sudo apt install -y apt-transport-https ca-certificates curl gnupg
```

Why install these dependencies?

- `apt-transport-https`: Required for secure repository access
- `ca-certificates`: Certificate authorities for HTTPS connections
- `curl`: Download ClickHouse repository key
- `gnupg`: GPG key management for repository verification

2. Add ClickHouse Repository

On both nodes, run:

```
# Add ClickHouse GPG key
curl -fsSL https://packages.clickhouse.com/gpg.key | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/clickhouse.gpg

# Add ClickHouse repository
echo "deb [signed-by=/etc/apt/trusted.gpg.d/clickhouse.gpg] https://packages.clickhouse.com/deb stable main" | sudo tee /etc/apt/sources.list.d/clickhouse.list

# Update package list
sudo apt update
```

Why add repository on both nodes?

- Ensures both nodes have access to the same ClickHouse version
- Consistent package sources across the cluster
- Simplifies maintenance and upgrades

Step 2: Verify SSH Connectivity Between Nodes

Prerequisite: This guide assumes you already have passwordless SSH configured for Hadoop. If SSH is not working, please set up Hadoop SSH configuration first.

1. Verify SSH is Working (Reusing Hadoop Setup)

From Primary Node (hadoop-master/clickhouse-primary):

```
# Test SSH to secondary node using existing Hadoop setup
ssh hadoop-worker1 'echo SSH connection to worker works'

# Test SSH to localhost
ssh localhost 'echo SSH to localhost works'
```

From Secondary Node (hadoop-worker1/clickhouse-secondary):

```
# Test SSH to primary node using existing Hadoop setup
ssh hadoop-master 'echo SSH connection to master works'

# Test SSH to localhost
ssh localhost 'echo SSH to localhost works'
```

2. Verify Hostname Resolution

From Primary Node:

```
# Test ClickHouse hostnames resolve correctly
ssh clickhouse-secondary 'hostname'
ssh clickhouse-primary 'hostname'
```

From Secondary Node:

```
# Test ClickHouse hostnames resolve correctly
ssh clickhouse-primary 'hostname'
ssh clickhouse-secondary 'hostname'
```

3. Required SSH Connections for ClickHouse

✅ **All these commands must work without passwords:**

```
# From primary node:
ssh hadoop-worker1 'hostname'  # Using existing Hadoop setup
ssh clickhouse-secondary 'hostname'  # New ClickHouse hostname
ssh localhost 'hostname'

# From secondary node:
ssh hadoop-master 'hostname'  # Using existing Hadoop setup
ssh clickhouse-primary 'hostname'  # New ClickHouse hostname
ssh localhost 'hostname'
```

4. Troubleshooting (If SSH Fails)

If any SSH command asks for password:

```
# Check SSH permissions
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys

# Verify SSH keys exist
ls -la ~/.ssh/

# Test basic connectivity
ping hadoop-master
ping hadoop-worker1
```

Why verify SSH connectivity?

- ClickHouse uses SSH for cluster management and administration
- Reuses existing Hadoop SSH configuration for consistency
- Ensures both systems can coexist with same network setup

Proceed to next step only when ALL SSH connections work without passwords.

System Setup (Both Nodes)

1. Verify Operating System

```
lsb_release -a
```

Why do this? This command verifies your Ubuntu version and helps ensure compatibility with ClickHouse. Both nodes should run the same Ubuntu version for consistency and to avoid compatibility issues.

2. Install Required Dependencies

On **both nodes**, run:

```
sudo apt update
sudo apt install -y apt-transport-https ca-certificates curl gnupg
```

Why install these dependencies?

- `apt-transport-https`: Required for secure repository access
- `ca-certificates`: Certificate authorities for HTTPS connections
- `curl`: Download ClickHouse repository key
- `gnupg`: GPG key management for repository verification

3. Add ClickHouse Repository

On **both nodes**, run:

```
# Add ClickHouse GPG key
curl -fsSL https://packages.clickhouse.com/gpg.key | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/clickhouse.gpg

# Add ClickHouse repository
echo "deb [signed-by=/etc/apt/trusted.gpg.d/clickhouse.gpg]
https://packages.clickhouse.com/deb stable main" | sudo tee
/etc/apt/sources.list.d/clickhouse.list

# Update package list
sudo apt update
```

Why add repository on both nodes?

- Ensures both nodes have access to the same ClickHouse version
- Consistent package sources across the cluster
- Simplifies maintenance and upgrades

Step 5: Install and Configure ZooKeeper Ensemble

ClickHouse cluster requires ZooKeeper for coordination and replication. We'll install ZooKeeper on both nodes in an ensemble.

1. Install Java Development Kit (JDK)

On **both nodes**, run:

```
sudo apt install -y openjdk-11-jdk
```

Why install Java? ZooKeeper requires Java to run, and OpenJDK 11 provides stable support for ZooKeeper.

2. Verify Java Installation

On **both nodes**, run:

```
java -version
javac -version
```

3. Install ZooKeeper

On **both nodes**, run:

```
# Download ZooKeeper
cd /tmp
wget https://archive.apache.org/dist/zookeeper/zookeeper-3.8.3/apache-
zookeeper-3.8.3-bin.tar.gz

# Extract ZooKeeper
sudo tar -xzf apache-zookeeper-3.8.3-bin.tar.gz -C /opt/
sudo mv /opt/apache-zookeeper-3.8.3-bin /opt/zookeeper

# Create ZooKeeper user
sudo useradd -r -s /bin/false zookeeper
sudo chown -R zookeeper:zookeeper /opt/zookeeper

# Create data directory
sudo mkdir -p /var/lib/zookeeper
sudo chown zookeeper:zookeeper /var/lib/zookeeper
```

Why install ZooKeeper on both nodes?

- ZooKeeper ensemble provides high availability coordination
- Odd number of nodes (3 recommended) for quorum, but 2 works for basic setup
- Both nodes participate in leader election and coordination

4. Configure ZooKeeper

On Primary Node, create `/opt/zookeeper/conf/zoo.cfg`:

```
sudo nano /opt/zookeeper/conf/zoo.cfg
```

Add the following configuration:

```
# The number of milliseconds of each tick
tickTime=2000

# The number of ticks that the initial synchronization phase can take
initLimit=10

# The number of ticks that can pass between sending a request and getting an acknowledgement
syncLimit=5

# the directory where the snapshot is stored
dataDir=/var/lib/zookeeper

# the port at which the clients will connect
clientPort=2181

# the maximum number of client connections
maxClientCnxns=60

# server numbers and their addresses
server.1=clickhouse-primary:2888:3888
server.2=clickhouse-secondary:2888:3888
```

On Secondary Node, create the same configuration file with identical content:

```
sudo nano /opt/zookeeper/conf/zoo.cfg
```

Paste the same configuration as above.

5. Create ZooKeeper Server IDs

On Primary Node:

```
echo "1" | sudo tee /var/lib/zookeeper/myid
sudo chown zookeeper:zookeeper /var/lib/zookeeper/myid
```

On Secondary Node:

```
echo "2" | sudo tee /var/lib/zookeeper/myid
sudo chown zookeeper:zookeeper /var/lib/zookeeper/myid
```

Why different server IDs?

- Each ZooKeeper node must have a unique ID from 1 to n
- IDs must match the server numbers in zoo.cfg

- ZooKeeper uses these IDs for leader election and coordination

6. Create ZooKeeper Service

On **both nodes**, create systemd service file:

```
sudo nano /etc/systemd/system/zookeeper.service
```

Add the following content:

```
[Unit]
Description=Apache ZooKeeper server
Documentation=https://zookeeper.apache.org
After=network.target

[Service]
Type=simple
User=zookeeper
Group=zookeeper
ExecStart=/opt/zookeeper/bin/zkServer.sh start-foreground
ExecStop=/opt/zookeeper/bin/zkServer.sh stop
WorkingDirectory=/var/lib/zookeeper
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

7. Start ZooKeeper Services

On **both nodes**, run:

```
# Reload systemd
sudo systemctl daemon-reload

# Start ZooKeeper
sudo systemctl start zookeeper

# Enable ZooKeeper to start on boot
sudo systemctl enable zookeeper

# Check status
sudo systemctl status zookeeper
```

8. Verify ZooKeeper Ensemble

On **both nodes**, run:

```
# Check if ZooKeeper is running
sudo systemctl status zookeeper

# Check ZooKeeper status
/opt/zookeeper/bin/zkServer.sh status
```

One node should show "Mode: leader" and the other should show "Mode: follower".

Why verify ZooKeeper status?

- Confirms the ensemble is working correctly
- Shows which node is the leader and follower
- Essential before proceeding with ClickHouse installation

Step 6: Install ClickHouse Server

1. Install ClickHouse Server

On **both nodes**, run:

```
# Install ClickHouse server and client
sudo apt install -y clickhouse-server clickhouse-client

# Stop ClickHouse (we'll configure first)
sudo systemctl stop clickhouse-server
```

Why install on both nodes?

- Each node needs ClickHouse server for distributed processing
- ClickHouse client for querying and management
- Both nodes participate in data storage and query processing

2. Create ClickHouse Configuration Directory Structure

On **both nodes**, run:

```
# Create config directories
sudo mkdir -p /etc/clickhouse-server/config.d
sudo mkdir -p /etc/clickhouse-server/users.d

# Set ownership
sudo chown -R clickhouse:clickhouse /etc/clickhouse-server
```

3. Configure ClickHouse Cluster Settings

On Primary Node, create cluster configuration:

```
sudo nano /etc/clickhouse-server/config.d/cluster.xml
```

Add the following configuration:

```
<?xml version="1.0"?>
<clickhouse>
    <!-- Remote servers definition -->
    <remote_servers>
        <cluster_2shards_1replicas>
            <shard>
                <replica>
                    <host>clickhouse-primary</host>
                    <port>9001</port>
                </replica>
            </shard>
            <shard>
                <replica>
                    <host>clickhouse-secondary</host>
                    <port>9001</port>
                </replica>
            </shard>
        </cluster_2shards_1replicas>

        <!-- Replicated cluster for high availability -->
        <cluster_1shard_2replicas>
            <shard>
                <internal_replication>true</internal_replication>
                <replica>
                    <host>clickhouse-primary</host>
                    <port>9001</port>
                    <user>default</user>
                </replica>
                <replica>
                    <host>clickhouse-secondary</host>
                    <port>9001</port>
                    <user>default</user>
                </replica>
            </shard>
        </cluster_1shard_2replicas>
    </remote_servers>

    <!-- ZooKeeper configuration -->
    <zookeeper>
        <node index="1">
            <host>clickhouse-primary</host>
            <port>2181</port>
        </node>
        <node index="2">
            <host>clickhouse-secondary</host>
            <port>2181</port>
        </node>
    </zookeeper>

    <!-- Macros for cluster configuration -->
    <macros>
        <shard>1</shard>
        <replica>clickhouse-primary</replica>
    </macros>

    <!-- Enable distributed_ddl -->
    <distributed_ddl>
        <path>/clickhouse/task_queue/ddl</path>
    </distributed_ddl>

    <!-- Compression settings -->
    <compression>
        <case>
            <min_part_size>10000000000</min_part_size>
            <min_part_size_ratio>0.01</min_part_size_ratio>
            <method>lz4hc</method>
        </case>
    </compression>

    <!-- Merge tree settings -->
    <merge_tree>
        <max_suspicious_broken_parts>5</max_suspicious_broken_parts>
    </merge_tree>
</clickhouse>
```

On Secondary Node, create similar configuration with different replica macro:

```
sudo nano /etc/clickhouse-server/config.d/cluster.xml
```

Add the following configuration (note the different replica macro):

```
<?xml version="1.0"?>
<clickhouse>
    <!-- Remote servers definition -->
    <remote_servers>
        <cluster_2shards_1replicas>
            <shard>
                <replica>
                    <host>clickhouse-primary</host>
                    <port>9001</port>
                </replica>
            </shard>
            <shard>
                <replica>
                    <host>clickhouse-secondary</host>
                    <port>9001</port>
                </replica>
            </shard>
        </cluster_2shards_1replicas>

        <!-- Replicated cluster for high availability -->
        <cluster_1shard_2replicas>
            <shard>
                <internal_replication>true</internal_replication>
                <replica>
                    <host>clickhouse-primary</host>
                    <port>9001</port>
                    <user>default</user>
                </replica>
                <replica>
                    <host>clickhouse-secondary</host>
                    <port>9001</port>
                    <user>default</user>
                </replica>
            </shard>
        </cluster_1shard_2replicas>
    </remote_servers>

    <!-- ZooKeeper configuration -->
    <zookeeper>
        <node index="1">
            <host>clickhouse-primary</host>
            <port>2181</port>
        </node>
        <node index="2">
            <host>clickhouse-secondary</host>
            <port>2181</port>
        </node>
    </zookeeper>

    <!-- Macros for cluster configuration -->
    <macros>
        <shard>1</shard>
        <replica>clickhouse-secondary</replica>
    </macros>

    <!-- Enable distributed_ddl -->
    <distributed_ddl>
        <path>/clickhouse/task_queue/ddl</path>
    </distributed_ddl>

    <!-- Compression settings -->
    <compression>
        <case>
            <min_part_size>10000000000</min_part_size>
            <min_part_size_ratio>0.01</min_part_size_ratio>
            <method>lz4hc</method>
        </case>
    </compression>
    <!-- Merge tree settings -->
    <merge_tree>
        <max_suspicious_broken_parts>5</max_suspicious_broken_parts>
    </merge_tree>
</clickhouse>
```

Why different replica macros?

- Each replica must have a unique identifier in the cluster
- ClickHouse uses macros to identify replicas for replication
- `replica` macro must be unique across all replicas in the same shard

4. Configure ClickHouse Network Settings

On **both nodes**, create network configuration:

```
sudo nano /etc/clickhouse-server/config.d/network.xml
```

Add the following configuration:

```
<?xml version="1.0"?>
<clickhouse>
    <!-- Listen on all interfaces -->
    <listen_host>::</listen_host>

    <!-- HTTP interface -->
    <http_port>8123</http_port>

    <!-- TCP interface -->
    <!-- Use port 9001 to avoid conflict with Hadoop NameNode port 9000 -->
    <tcp_port>9001</tcp_port>

    <!-- MySQL interface (optional) -->
    <mysql_port>9004</mysql_port>

    <!-- PostgreSQL interface (optional) -->
    <postgresql_port>9005</postgresql_port>

    <!-- Interserver HTTP for replication -->
    <interserver_http_host>clickhouse-primary</interserver_http_host>
    <interserver_http_port>9009</interserver_http_port>

    <!-- Maximum connections -->
    <max_connections>4096</max_connections>

    <!-- Keep alive timeout -->
    <keep_alive_timeout>3</keep_alive_timeout>

    <!-- Connection pool size -->
    <connect_timeout>10</connect_timeout>
    <receive_timeout>300</receive_timeout>
    <send_timeout>300</send_timeout>
</clickhouse>
```

Important: On the secondary node, change `interserver_http_host` to `clickhouse-secondary`:

```
<interserver_http_host>clickhouse-secondary</interserver_http_host>
```

5. Configure ClickHouse Data Storage

On **both nodes**, create data storage configuration:

```
sudo nano /etc/clickhouse-server/config.d/storage.xml
```

Add the following configuration:

```
<?xml version="1.0"?>
<clickhouse>
    <!-- Path to data directory -->
    <path>/var/lib/clickhouse/</path>

    <!-- Path to temporary data -->
    <tmp_path>/var/lib/clickhouse/tmp/</tmp_path>

    <!-- Path to user files -->
    <user_files_path>/var/lib/clickhouse/user_files/</user_files_path>

    <!-- Path to access control -->
    <access_control_path>/var/lib/clickhouse/access/</access_control_path>

    <!-- MergeTree settings -->
    <merge_tree>
        <max_suspicious_broken_parts>5</max_suspicious_broken_parts>

<max_bytes_to_merge_at_max_space_in_pool>10485760000</max_bytes_to_merge_at_max_space_in_pool>
    </merge_tree>

    <!-- Storage configuration -->
    <storage_configuration>
        <disks>
            <default>
                <path>/var/lib/clickhouse/</path>
                <keep_free_space_bytes>10485760</keep_free_space_bytes>
            </default>
        </disks>
    </storage_configuration>
</clickhouse>
```

6. Create Data Directories

On **both nodes**, run:

```
# Create ClickHouse data directories
sudo mkdir -p /var/lib/clickhouse
sudo mkdir -p /var/lib/clickhouse/tmp
sudo mkdir -p /var/lib/clickhouse/user_files
sudo mkdir -p /var/lib/clickhouse/access
sudo mkdir -p /var/lib/clickhouse/preprocessed_configs
sudo mkdir -p /var/log/clickhouse-server

# Set ownership
sudo chown -R clickhouse:clickhouse /var/lib/clickhouse
sudo chown -R clickhouse:clickhouse /var/log/clickhouse-server

# Set permissions
sudo chmod 750 /var/lib/clickhouse
sudo chmod 750 /var/log/clickhouse-server
```

7. Configure Users and Access

On **both nodes**, create user configuration:

```
sudo nano /etc/clickhouse-server/users.d/custom_users.xml
```

Add the following configuration:

```
<?xml version="1.0"?>
<clickhouse>
    <!-- Default user profile -->
    <profiles>
        <default>
            <max_memory_usage>10000000000</max_memory_usage>
            <max_memory_usage_for_user>0</max_memory_usage_for_user>

<max_memory_usage_for_all_queries>0</max_memory_usage_for_all_queries>

<max_bytes_before_external_group_by>5000000000</max_bytes_before_external_group_by>

<max_bytes_before_external_sort>5000000000</max_bytes_before_external_sort>
            <max_execution_time>3600</max_execution_time>
            <max_result_rows>1000000</max_result_rows>
            <max_result_bytes></max_result_bytes>
            <max_rows_in_set></max_rows_in_set>
            <max_bytes_in_set></max_bytes_in_set>
            <transfer_overflow_mode>throw</transfer_overflow_mode>

<empty_result_for_aggregation_by_empty_set>1</empty_result_for_aggregation_by_empty_set>
            <load_balancing>random</load_balancing>
        </default>
    </profiles>

    <!-- Custom user for cluster operations -->
    <users>
        <cluster_user>
            <password>cluster_password</password>
            <networks>
                <ip>::/0</ip>
            </networks>
            <profile>default</profile>
            <quota>default</quota>
            <databases>
                <database>cluster_db</database>
            </databases>
        </cluster_user>
    </users>

    <!-- Quotas -->
    <quotas>
        <default>
            <interval>
                <duration>3600</duration>
                <queries>0</queries>
                <errors>0</errors>
                <result_rows>0</result_rows>
                <read_rows>0</read_rows>
                <execution_time>0</execution_time>
            </interval>
        </default>
    </quotas>
</clickhouse>
```

Step 8: Start and Verify ClickHouse Cluster

1. Start ClickHouse Server

On **both nodes**, run:

```
# Start ClickHouse server
sudo systemctl start clickhouse-server

# Enable ClickHouse to start on boot
```

```
sudo systemctl enable clickhouse-server

# Check status
sudo systemctl status clickhouse-server
```

2. Verify ClickHouse Services

On **both nodes**, check if ClickHouse is running:

```
# Check process
ps aux | grep clickhouse

# Check logs
sudo tail -f /var/log/clickhouse-server/clickhouse-server.log

# Check listening ports
sudo netstat -tulpn | grep clickhouse
```

3. Test ClickHouse Connectivity

From Primary Node:

```
# Test local connection
clickhouse-client --query "SELECT 'ClickHouse Primary is working' as message"

# Test connection to secondary node
clickhouse-client --host clickhouse-secondary --query "SELECT 'Connection to Secondary works' as message"
```

From Secondary Node:

```
# Test local connection
clickhouse-client --query "SELECT 'ClickHouse Secondary is working' as message"

# Test connection to primary node
clickhouse-client --host clickhouse-primary --query "SELECT 'Connection to Primary works' as message"
```

Step 9: Test Replication and Distributed Queries

1. Create Distributed Databases

From Primary Node, connect to ClickHouse and create databases:

```
clickhouse-client --host clickhouse-primary
```

Execute the following SQL:

```
-- Create standard database
CREATE DATABASE cluster_db;

-- Create distributed database (will be automatically replicated)
CREATE DATABASE replicated_db ON CLUSTER cluster_1shard_2replicas ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/database/{database}', '{replica}');

-- Use the database
USE cluster_db;
```

2. Create Replicated Tables

From Primary Node, create replicated tables:

```
-- Create a replicated MergeTree table
CREATE TABLE replicated_table ON CLUSTER cluster_1shard_2replicas
(
    id UInt64,
    timestamp DateTime,
    message String,
    value Float64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/cluster_db/replicated_table', '{replica}')
ORDER BY id
PARTITION BY toYYYYMM(timestamp);

-- Create a distributed table for queries
CREATE TABLE distributed_table AS replicated_table
ENGINE = Distributed(cluster_2shards_1replicas, cluster_db, replicated_table, rand());

-- Show cluster configuration
SELECT * FROM system.clusters WHERE cluster LIKE '%cluster%';
```

3. Test Replication

From Primary Node, insert test data:

```
-- Insert some test data
INSERT INTO replicated_table VALUES
    (1, now(), 'First message from primary', 100.5),
    (2, now(), 'Second message from primary', 200.75),
    (3, now(), 'Third message from primary', 300.25);

-- Verify data on primary
SELECT * FROM replicated_table ORDER BY id;

-- Check replication status
SELECT * FROM system.replicas WHERE database = 'cluster_db' AND table =
'replicated_table';
```

From Secondary Node, verify data was replicated:

```
clickhouse-client --host clickhouse-secondary
```

```
-- Check if data was replicated
SELECT * FROM cluster_db.replicated_table ORDER BY id;

-- Check replication status from secondary perspective
SELECT * FROM system.replicas WHERE database = 'cluster_db' AND table =
'replicated_table';

-- Check cluster status
SELECT * FROM system.clusters;
```

4. Test Distributed Queries

From Primary Node, test distributed functionality:

```
-- Test distributed table query
SELECT 'Distributed query test' as test_message;
SELECT * FROM distributed_table ORDER BY id;

-- Test query distribution across cluster
SELECT
    cluster() as cluster_name,
    shardNum() as shard_number,
    replicaNum() as replica_number,
    count(*) as row_count
FROM replicated_table
GROUP BY cluster(), shardNum(), replicaNum();

-- Test inserting data through distributed table
INSERT INTO distributed_table VALUES
```

```
    (4, now(), 'Insert through distributed table', 400.5),
    (5, now(), 'Another distributed insert', 500.75);

-- Verify data exists on all replicas
SELECT 'After distributed insert:' as message;
SELECT * FROM replicated_table ORDER BY id;
```

5. Test High Availability

Test Primary Node Failure Simulation:

1. **Stop ClickHouse on Primary Node**:

   ```
   # On primary node
   sudo systemctl stop clickhouse-server
   ```

2. **Query Secondary Node Only**:

   ```
   # From secondary node or any client
   clickhouse-client --host clickhouse-secondary --query "SELECT COUNT(*) as row_count FROM cluster_db.replicated_table"
   ```

3. **Restart Primary Node**:

   ```
   # On primary node
   sudo systemctl start clickhouse-server

   # Check replication status
   clickhouse-client --host clickhouse-primary --query "SELECT * FROM system.replicas WHERE database = 'cluster_db' AND table = 'replicated_table'"
   ```

Why test high availability?

- Verifies automatic failover capabilities
- Confirms data remains available during node failures
- Tests recovery mechanisms when nodes come back online

Cluster Management

Starting and Stopping the Cluster

To Start the Complete Cluster:

```
# From primary node (or any node with SSH access)
ssh clickhouse-primary "sudo systemctl start clickhouse-server"
ssh clickhouse-secondary "sudo systemctl start clickhouse-server"
ssh clickhouse-primary "sudo systemctl start zookeeper"
ssh clickhouse-secondary "sudo systemctl start zookeeper"
```

To Stop the Cluster:

```
# From primary node (or any node with SSH access)
ssh clickhouse-secondary "sudo systemctl stop clickhouse-server"
ssh clickhouse-primary "sudo systemctl stop clickhouse-server"
ssh clickhouse-secondary "sudo systemctl stop zookeeper"
ssh clickhouse-primary "sudo systemctl stop zookeeper"
```

Cluster Health Monitoring

From any node:

```
# Check ClickHouse service status
ssh clickhouse-primary "sudo systemctl status clickhouse-server"
ssh clickhouse-secondary "sudo systemctl status clickhouse-server"

# Check ZooKeeper status
ssh clickhouse-primary "sudo systemctl status zookeeper"
ssh clickhouse-secondary "sudo systemctl status zookeeper"

# Check cluster status
clickhouse-client --host clickhouse-primary --query "SELECT * FROM
system.clusters"
clickhouse-client --host clickhouse-secondary --query "SELECT * FROM
system.replicas"
```

Monitoring Queries:

```
-- Check cluster nodes
SELECT host_name, port, status, shard_num, replica_num
FROM system.clusters
WHERE cluster = 'cluster_2shards_1replicas';

-- Check replication status
SELECT database, table, is_leader, is_readonly, absolute_delay, queue_size
FROM system.replicas;

-- Check table sizes across cluster
SELECT
    database,
    table,
    formatReadableSize(sum(bytes)) as size,
    sum(rows) as total_rows
FROM system.parts
WHERE database = 'cluster_db'
GROUP BY database, table;
```

Data Distribution and Rebalancing

Check Data Distribution:

```
-- Check data distribution across shards
SELECT
    shardNum() as shard,
    replicaNum() as replica,
    count(*) as rows,
    hostName() as node
FROM replicated_table
GROUP BY shardNum(), replicaNum(), hostName()
ORDER BY shard, replica;
```

Manual Data Rebalancing (if needed):

```
-- Force replication (if replication lag exists)
ALTER TABLE replicated_table SYNC REPLICA ON CLUSTER
cluster_1shard_2replicas;

-- Check replication queue
SELECT * FROM system.replication_queue WHERE database = 'cluster_db' AND
table = 'replicated_table';
```

Web Interfaces and Monitoring

ClickHouse Web Interface

ClickHouse provides a web interface for query execution:

- **Primary Node**: http://clickhouse-primary:8123
- **Secondary Node**: http://clickhouse-secondary:8123

Using the Web Interface:

1. Open browser to http://clickhouse-primary:8123
2. Enter SQL queries in the text area
3. Click "Execute" to run queries
4. Results are displayed in table format

Example queries for web interface:

```
SELECT version();
SELECT now() as current_time;
SELECT * FROM system.clusters;
```

Monitoring and Metrics

ClickHouse provides extensive monitoring capabilities:

System Tables for Monitoring:

```
-- General cluster status
SELECT * FROM system.clusters;

-- Replication status
SELECT * FROM system.replicas;

-- Query performance
SELECT * FROM system.query_log ORDER BY event_time DESC LIMIT 10;

-- Memory usage
SELECT
    formatReadableSize(memory_usage) as memory_used,
    formatReadableSize(memory_usage * 100 / max_memory_usage) as
memory_percent
FROM system.metrics
WHERE metric LIKE '%memory%';

-- Disk usage
SELECT
    formatReadableSize(sum(bytes)) as total_size,
    sum(rows) as total_rows
FROM system.parts
WHERE active = 1;
```

Performance Monitoring

Query Performance Analysis:

```
-- Enable query logging
SET log_queries = 1;
SET log_queries_min_type = 'QUERY_FINISH';

-- Analyze recent queries
SELECT
    query,
    query_duration_ms / 1000 as duration_seconds,
    memory_usage,
    result_rows,
    result_bytes
FROM system.query_log
WHERE event_time > now() - INTERVAL 1 HOUR
  AND type = 'QueryFinish'
ORDER BY query_duration_ms DESC
LIMIT 10;
```

Advanced Cluster Configuration

Table Replication Strategies

ReplicatedMergeTree for High Availability:

```
-- Create highly replicated table
CREATE TABLE critical_data ON CLUSTER cluster_1shard_2replicas
(
    id UInt64,
    event_time DateTime,
    event_type String,
    data String,
    INDEX idx_type event_type TYPE tokenbf_v1(256, 2, 0) GRANULARITY 4
)
ENGINE =
ReplicatedMergeTree('/clickhouse/tables/{shard}/cluster_db/critical_data',
'{replica}')
ORDER BY (event_time, id)
PARTITION BY toYYYYMM(event_time)
TTL event_time + INTERVAL 30 DAY DELETE;
```

Distributed Tables for Query Distribution:

```
-- Create distributed table for even distribution
CREATE TABLE distributed_critical_data AS critical_data
ENGINE = Distributed(cluster_2shards_1replicas, cluster_db, critical_data,
sipHash64(id));
```

Backup and Recovery

Creating Backups:

```
-- Backup using ClickHouse backup tools
BACKUP TABLE replicated_table ON CLUSTER cluster_1shard_2replicas TO
Disk('backups', 'replicated_table_backup');

-- Backup entire database
BACKUP DATABASE cluster_db ON CLUSTER cluster_1shard_2replicas TO
Disk('backups', 'cluster_db_backup');
```

Restoring from Backup:

```sql
-- Restore table
RESTORE TABLE replicated_table ON CLUSTER cluster_1shard_2replicas FROM
Disk('backups', 'replicated_table_backup');

-- Restore database
RESTORE DATABASE cluster_db ON CLUSTER cluster_1shard_2replicas FROM
Disk('backups', 'cluster_db_backup');
```

Performance Optimization

Memory Configuration:

```xml
<!-- Add to /etc/clickhouse-server/config.d/performance.xml -->
<?xml version="1.0"?>
<clickhouse>
    <max_memory_usage>10000000000</max_memory_usage>
    <max_memory_usage_for_user>8000000000</max_memory_usage_for_user>

<max_bytes_before_external_group_by>2000000000</max_bytes_before_external_group_by>

<max_bytes_before_external_sort>2000000000</max_bytes_before_external_sort>
</clickhouse>
```

Merge Optimization:

```xml
<!-- Merge tree performance -->
<merge_tree>
    <max_suspicious_broken_parts>5</max_suspicious_broken_parts>

<max_bytes_to_merge_at_max_space_in_pool>10485760000</max_bytes_to_merge_at_max_space_in_pool>
    <min_bytes_for_wide_part>10485760</min_bytes_for_wide_part>
    <min_rows_for_wide_part>1048576</min_rows_for_wide_part>
</merge_tree>
```

Troubleshooting

Common Cluster Issues

1. ZooKeeper Connection Issues

Symptoms: ClickHouse fails to start with ZooKeeper connection errors

Solutions:

```
# Check ZooKeeper status on both nodes
sudo systemctl status zookeeper

# Test ZooKeeper connectivity
/opt/zookeeper/bin/zkCli.sh -server clickhouse-primary:2181
/opt/zookeeper/bin/zkCli.sh -server clickhouse-secondary:2181

# Check ZooKeeper logs
sudo tail -f /opt/zookeeper/logs/zookeeper.log

# Restart services
sudo systemctl restart zookeeper
sudo systemctl restart clickhouse-server
```

2. Replication Lag

Symptoms: Data not appearing on all replicas

Solutions:

```sql
-- Check replication queue
SELECT * FROM system.replication_queue WHERE database = 'cluster_db';

-- Force replication
ALTER TABLE replicated_table SYNC REPLICA;

-- Check ZooKeeper lag
SELECT absolute_delay, queue_size
FROM system.replicas
WHERE database = 'cluster_db' AND table = 'replicated_table';
```

3. Network Connectivity Issues

Symptoms: Nodes cannot communicate or queries fail

Solutions:

```
# Test connectivity between nodes
nc -zv clickhouse-primary 9001
nc -zv clickhouse-secondary 9001

# Check firewall settings
sudo ufw status
sudo ufw allow 9000/tcp
sudo ufw allow 8123/tcp
sudo ufw allow 9009/tcp
```

```
# Check ClickHouse network configuration
sudo nano /etc/clickhouse-server/config.d/network.xml
```

4. Memory Issues

Symptoms: Queries fail with memory errors or OOM

Solutions:

```
-- Monitor memory usage
SELECT metric, value
FROM system.metrics
WHERE metric LIKE '%memory%';

-- Adjust memory limits
SET max_memory_usage = 20000000000;

-- Check query memory usage
SELECT
    query,
    memory_usage,
    formatReadableSize(memory_usage) as memory_formatted
FROM system.query_log
WHERE type = 'QueryFinish'
ORDER BY memory_usage DESC
LIMIT 10;
```

Log Locations

ClickHouse Server Logs:

- **Primary Node**: `/var/log/clickhouse-server/clickhouse-server.log`
- **Secondary Node**: `/var/log/clickhouse-server/clickhouse-server.log`

ZooKeeper Logs:

- **Both Nodes**: `/opt/zookeeper/logs/zookeeper.log`

System Logs:

- **ClickHouse Service**: `journalctl -u clickhouse-server`
- **ZooKeeper Service**: `journalctl -u zookeeper`

Diagnostic Commands

Cluster Health Check:

```
# ClickHouse cluster status
clickhouse-client --host clickhouse-primary --query "SELECT * FROM system.clusters"
clickhouse-client --host clickhouse-secondary --query "SELECT * FROM system.replicas"

# ZooKeeper ensemble status
/opt/zookeeper/bin/zkServer.sh status

# Network connectivity
nc -zv clickhouse-primary 8123
nc -zv clickhouse-secondary 8123

# Process status
ps aux | grep clickhouse
ps aux | grep zookeeper
```

Security Configuration

Network Security

Configure Firewall:

```
# On both nodes
sudo ufw allow 22/tcp          # SSH
sudo ufw allow 8123/tcp        # ClickHouse HTTP
sudo ufw allow 9001/tcp        # ClickHouse TCP
sudo ufw allow 9009/tcp        # ClickHouse interserver
sudo ufw allow 2181/tcp        # ZooKeeper
sudo ufw allow 2888/tcp        # ZooKeeper
sudo ufw allow 3888/tcp        # ZooKeeper
sudo ufw enable
```

ClickHouse Authentication

Enable User Authentication:

```
<!-- In /etc/clickhouse-server/users.d/security.xml -->
<?xml version="1.0"?>
<clickhouse>
    <!-- IP restrictions for default user -->
    <users>
        <default>
            <password>secure_password</password>
            <networks>
                <ip>127.0.0.1</ip>
                <ip>192.168.1.0/24</ip>
            </networks>
            <profile>readonly</profile>
            <quota>default</quota>
        </default>
    </users>
</clickhouse>
```

SSL/TLS Configuration

Enable SSL (Advanced):

```
<!-- In /etc/clickhouse-server/config.d/ssl.xml -->
<?xml version="1.0"?>
<clickhouse>
    <https_port>8443</https_port>
    <tcp_port_secure>9440</tcp_port_secure>

    <openSSL>
        <server>
            <certificateFile>/etc/clickhouse-server/certs/server.crt</certificateFile>
            <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
            <caConfig>/etc/clickhouse-server/certs/ca.crt</caConfig>
            <verificationMode>relaxed</verificationMode>
            <cacheSessions>true</cacheSessions>
            <disableProtocols>sslv2,sslv3</disableProtocols>
            <preferServerCiphers>true</preferServerCiphers>
        </server>
    </openSSL>
</clickhouse>
```

Scaling Beyond 2 Nodes

Adding Additional Nodes

To expand your ClickHouse cluster beyond 2 nodes:

1. **Prepare New Node**:

   ```
   # Install Ubuntu and required packages
   # Configure network and hostnames
   # Set up SSH access from primary node
   ```

2. **Install ClickHouse and ZooKeeper**:

   ```
   # Copy configuration from existing nodes
   # Update replica macros and server IDs
   ```

```
# Add node to cluster configuration
```

3. **Update Cluster Configuration**:

```
<!-- Update /etc/clickhouse-server/config.d/cluster.xml on all nodes -->
<remote_servers>
    <cluster_3shards_2replicas>
        <!-- Add new shard/replica configurations -->
    </cluster_3shards_2replicas>
</remote_servers>
```

4. **Restart Services**:

```
# Restart ZooKeeper ensemble
# Restart ClickHouse on all nodes
```

High Availability Architecture

For production clusters, consider:

- **Multiple ZooKeeper nodes** (3, 5, or 7 for quorum)
- **Geographically distributed replicas**
- **Load balancers** for client connections
- **Monitoring and alerting systems**
- **Automated backup and recovery procedures**

Step 10: Final Verification with Both Systems Running

1. Verify Both Clusters Are Working

From Master Node (hadoop-master/clickhouse-primary):

```
# Test Hadoop is still working
jps
hdfs dfsadmin -report
yarn node -list

# Test ClickHouse is working
clickhouse-client --port 9001 --query "SELECT version()"
clickhouse-client --host clickhouse-secondary --port 9001 --query "SELECT version()"

# Test Hadoop web interfaces
# Hadoop NameNode: http://hadoop-master:9870
# Hadoop ResourceManager: http://hadoop-master:8088
# Test ClickHouse web interfaces
# ClickHouse Primary: http://clickhouse-primary:8123
# ClickHouse Secondary: http://clickhouse-secondary:8123
```

From Worker Node (hadoop-worker1/clickhouse-secondary):

```
# Test Hadoop is still working
jps
hdfs dfs -ls /

# Test ClickHouse is working
clickhouse-client --port 9001 --query "SELECT version()"
clickhouse-client --host clickhouse-primary --port 9001 --query "SELECT version()"
```

2. Verify Port Compatibility

On both nodes, check all services are running:

```
# Check Hadoop services
netstat -tulpn | grep -E '(8020|9000|9870|8088|19888)'

# Check ClickHouse services
netstat -tulpn | grep -E '(8123|9001|9004|9005|9009|9010)'

# Check ZooKeeper services
netstat -tulpn | grep -E '(2181|2888|3888)'
```

3. Test Resource Usage

```
# Check system resource usage
free -h
df -h
top

# Check Hadoop processes
ps aux | grep -E '(NameNode|DataNode|ResourceManager|NodeManager)'

# Check ClickHouse processes
ps aux | grep clickhouse

# Check ZooKeeper processes
ps aux | grep zookeeper
```

Why perform this final verification?

- Ensures both systems are running without conflicts
- Confirms resource allocation is adequate for both clusters
- Verifies network connectivity for all services
- Provides baseline for monitoring and troubleshooting

4. Performance Considerations

- **Memory**: Monitor total memory usage to ensure both systems have adequate resources
- **Disk Space**: Plan storage allocation between HDFS and ClickHouse data directories
- **CPU Load**: Monitor CPU usage during concurrent Hadoop and ClickHouse operations
- **Network**: Ensure network bandwidth can handle both Hadoop and ClickHouse traffic

References

- ClickHouse Official Documentation
- ClickHouse Cluster Setup Guide
- ZooKeeper Administrator's Guide
- ClickHouse Performance Tuning
- ClickHouse Replication Guide

License

This guide is provided as-is for educational purposes. ClickHouse is licensed under the Apache License 2.0.