ClickHouse 2-Node Cluster Complete Setup Guide

This comprehensive guide provides step-by-step instructions for setting up a production-ready ClickHouse 2-node cluster with ZooKeeper coordination, replication, and high availability.

Table of Contents

Prerequisites

System Requirements

Both Nodes:

- Ubuntu 20.04 LTS or 24.04 LTS
- Minimum 4GB RAM (8GB+ recommended)
- 20GB+ available disk space for data
- Stable network connection between nodes
- Root or sudo access

Network Requirements

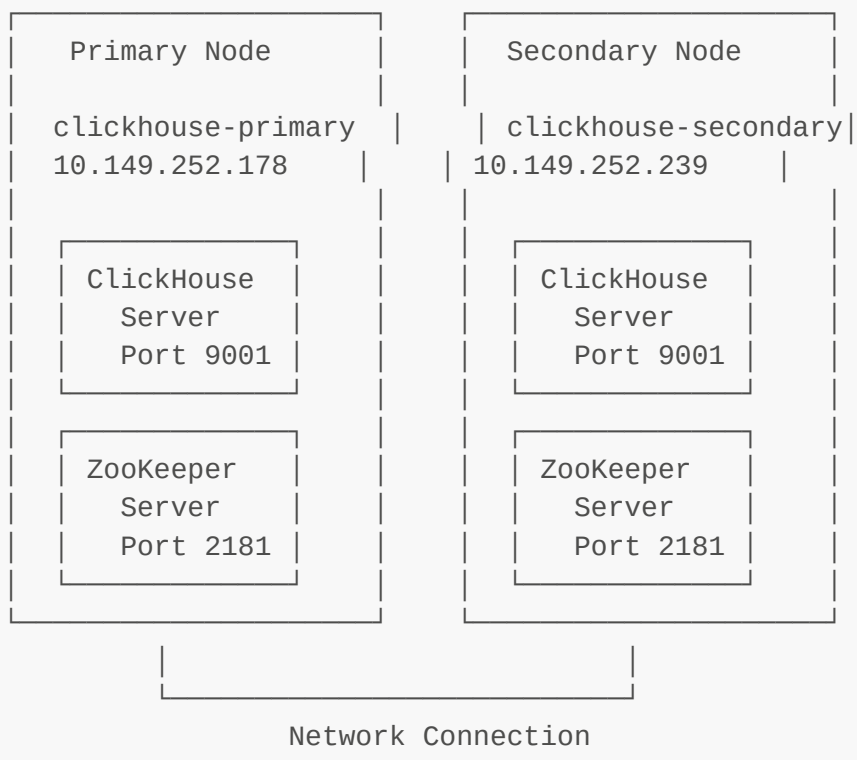
- Both nodes must be able to communicate via network
- Static IP addresses recommended
- Open firewall for required ports

Required Ports

Service	Port	Description
SSH	22	Remote access

Service	Port	Description
ClickHouse HTTP	8123	Web interface and HTTP queries
ClickHouse TCP	9001	Native client connections
ClickHouse Interserver	9009	Data replication
ZooKeeper Client	2181	ZooKeeper client connections
ZooKeeper Peer	2888	ZooKeeper ensemble communication
ZooKeeper Leader Election	3888	ZooKeeper leader election

System Architecture



Step 1: Initial Setup and Verification

1.1 Update System Packages

On both nodes, update the system:

```
sudo apt update && sudo apt upgrade -y
```

1.2 Verify Network Connectivity

Check if nodes can communicate:

```
# From primary node (replace with your actual IPs)
ping -c 3 10.149.252.239

# From secondary node
ping -c 3 10.149.252.178
```

1.3 Check System Information

Verify system details on both nodes:

```
# Check OS version
lsb_release -a

# Check system resources
free -h
df -h

# Check network interfaces
ip addr show
```

Step 2: Configure Hostnames and DNS Resolution

2.1 Set Hostnames

On Primary Node:

```
sudo hostnamectl set-hostname clickhouse-primary
```

On Secondary Node:

```
sudo hostnamectl set-hostname clickhouse-secondary
```

2.2 Configure Hosts File

On both nodes, edit `/etc/hosts` to add hostname resolution:

```
sudo nano /etc/hosts
```

Add the following lines (replace with your actual IP addresses):

```

```

```
# ClickHouse Cluster
10.149.252.178    clickhouse-primary
10.149.252.239    clickhouse-secondary
```

2.3 Verify Hostname Resolution

From Primary Node:

```
# Test hostname resolution
nslookup clickhouse-primary
nslookup clickhouse-secondary

# Test connectivity
ping -c 2 clickhouse-secondary
```

From Secondary Node:

```
# Test hostname resolution
nslookup clickhouse-primary
nslookup clickhouse-secondary

# Test connectivity
ping -c 2 clickhouse-primary
```

Expected Results:

- Both hostnames should resolve to correct IP addresses
- Ping should receive responses from both nodes

Step 3: Setup Passwordless SSH

3.1 Generate SSH Key (if not exists)

On Primary Node:

```
# Check if SSH key already exists
ls -la ~/.ssh/id_rsa.pub

# If not, generate new SSH key
ssh-keygen -t rsa -b 4096 -C "clickhouse-cluster" -f ~/.ssh/id_rsa -N ""
```

3.2 Copy SSH Key to Secondary Node

From Primary Node:

```
# Copy SSH key to secondary node
ssh-copy-id ubuntu@clickhouse-secondary

# Or manually copy if ssh-copy-id not available
cat ~/.ssh/id_rsa.pub | ssh ubuntu@clickhouse-secondary 'mkdir -p ~/.ssh &&
chmod 700 ~/.ssh && cat >> ~/.ssh/authorized_keys && chmod 600
~/.ssh/authorized_keys'
```

3.3 Setup SSH Key from Secondary to Primary

From Secondary Node:

```
# Generate SSH key if not exists
ssh-keygen -t rsa -b 4096 -C "clickhouse-cluster" -f ~/.ssh/id_rsa -N ""

# Copy to primary node
ssh-copy-id ubuntu@clickhouse-primary
```

3.4 Verify Passwordless SSH

From Primary Node:

```
# Test SSH to secondary node
ssh clickhouse-secondary 'hostname && whoami'
```

From Secondary Node:

```
# Test SSH to primary node
ssh clickhouse-primary 'hostname && whoami'
```

Expected Results:

- Both commands should execute without password prompts
- Should return correct hostnames: `clickhouse-primary` and `clickhouse-secondary`

Step 4: Install Java Runtime

ClickHouse and ZooKeeper require Java 11 or later.

4.1 Check Java Installation

On both nodes:

```
java -version
```

4.2 Install Java 11 (if not installed)

On both nodes:

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

4.3 Verify Java Installation

On both nodes:

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

Expected Output:

```
openjdk version "11.0.x" 2024-xx-xx
OpenJDK Runtime Environment (build 11.0.x+xx)
OpenJDK 64-Bit Server VM (build 11.0.x+xx, mixed mode, sharing)
```

Step 5: Install ClickHouse

5.1 Add ClickHouse Repository

On both nodes:

```
# Install prerequisites
sudo apt install -y apt-transport-https ca-certificates curl gnupg

# Add ClickHouse GPG key
curl -fsSL 'https://packages.clickhouse.com/gpg.key' | sudo gpg --dearmor -
o /usr/share/keyrings/clickhouse-keyring.gpg

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

# Update package list
sudo apt update
```

5.2 Install ClickHouse Packages

On both nodes:

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

5.3 Verify ClickHouse Installation

On both nodes:

```
# Check installed packages
dpkg -l | grep clickhouse

# Check ClickHouse version
clickhouse-server --version
clickhouse-client --version

# Check if clickhouse user was created
id clickhouse
```

5.4 Stop ClickHouse (we'll configure first)

On both nodes:

```
sudo systemctl stop clickhouse-server
```

Step 6: Install ZooKeeper

6.1 Download ZooKeeper

From Primary Node (we'll copy to secondary node later):

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

6.2 Extract and Install ZooKeeper

On Primary Node:

```
# 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

# Set ownership
sudo chown -R zookeeper:zookeeper /opt/zookeeper

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

6.3 Install ZooKeeper on Secondary Node

From Primary Node:

```
# Copy ZooKeeper installation to secondary node
ssh clickhouse-secondary 'cd /tmp && wget -q
https://archive.apache.org/dist/zookeeper/zookeeper-3.8.3/apache-zookeeper-
3.8.3-bin.tar.gz && sudo tar -xzf apache-zookeeper-3.8.3-bin.tar.gz -C
/opt/ && sudo mv /opt/apache-zookeeper-3.8.3-bin /opt/zookeeper'

# Setup user and permissions on secondary node
ssh clickhouse-secondary 'sudo useradd -r -s /bin/false zookeeper'
ssh clickhouse-secondary 'sudo chown -R zookeeper:zookeeper /opt/zookeeper'
ssh clickhouse-secondary 'sudo mkdir -p /var/lib/zookeeper'
ssh clickhouse-secondary 'sudo chown zookeeper:zookeeper
/var/lib/zookeeper'
```

Step 7: Configure ZooKeeper Ensemble

7.1 Create ZooKeeper Configuration

On Primary Node:

```
sudo tee /opt/zookeeper/conf/zoo.cfg > /dev/null <<EOF
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/var/lib/zookeeper
clientPort=2181
maxClientCnxns=60
server.1=clickhouse-primary:2888:3888
server.2=clickhouse-secondary:2888:3888
EOF
```

Copy to Secondary Node:

```
sudo tee /opt/zookeeper/conf/zoo.cfg > /dev/null <<EOF
tickTime=2000
```

```
initLimit=10
syncLimit=5
dataDir=/var/lib/zookeeper
clientPort=2181
maxClientCnxns=60
server.1=clickhouse-primary:2888:3888
server.2=clickhouse-secondary:2888:3888
EOF
```

7.2 Set 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
```

7.3 Create ZooKeeper Service

On Primary Node:

```
sudo tee /etc/systemd/system/zookeeper.service > /dev/null <<EOF
[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
RestartSec=5

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

On Secondary Node:

```
sudo tee /etc/systemd/system/zookeeper.service > /dev/null <<EOF
[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
RestartSec=5

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

7.4 Start ZooKeeper Services

On both nodes:

```
# Reload systemd
sudo systemctl daemon-reload

# Start ZooKeeper
sudo systemctl start zookeeper

# Enable auto-start
sudo systemctl enable zookeeper

# Check status
sudo systemctl status zookeeper
```

7.5 Verify ZooKeeper Ensemble

On both nodes:

```
# Check service status
sudo systemctl status zookeeper --no-pager -l

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

# Check if ZooKeeper is listening
sudo netstat -tulpn | grep -E '(2181|2888|3888)'
```

Expected Results:

- One node should show "Mode: leader"
- Other node should show "Mode: follower"
- Port 2181 should be listening on both nodes

Step 8: Configure ClickHouse Cluster

8.1 Create Configuration Directories

On both nodes:

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

8.2 Create Cluster Configuration

On Primary Node:

```
sudo tee /etc/clickhouse-server/config.d/cluster.xml > /dev/null <<'EOF'
<?xml version="1.0"?>
<clickhouse>
    <!-- Remote servers definition -->
    <remote_servers>
        <!-- 2-shard, 1-replica cluster for distributed queries -->
        <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>

        <!-- 1-shard, 2-replica 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>127.0.1.1</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>
EOF
```

On Secondary Node:

```
sudo tee /etc/clickhouse-server/config.d/cluster.xml > /dev/null <<EOF
<?xml version="1.0"?>
<clickhouse>
    <!-- Remote servers definition -->
    <remote_servers>
        <!-- 2-shard, 1-replica cluster for distributed queries -->
        <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>

        <!-- 1-shard, 2-replica 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>127.0.1.1</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>
EOF
```

8.3 Configure Network Settings

On Primary Node:

```
sudo tee /etc/clickhouse-server/config.d/network.xml > /dev/null <<'EOF'
<?xml version="1.0"?>
<clickhouse>
    <!-- Listen on all interfaces -->
    <listen_host>::</listen_host>

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

    <!-- TCP interface (use 9001 to avoid Hadoop conflicts) -->
    <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>
EOF
```

On Secondary Node:

```
sudo tee /etc/clickhouse-server/config.d/network.xml > /dev/null <<EOF
<?xml version="1.0"?>
<clickhouse>
    <!-- Listen on all interfaces -->
    <listen_host>::</listen_host>

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

    <!-- TCP interface (use 9001 to avoid Hadoop conflicts) -->
    <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-secondary</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>
EOF
```

8.4 Configure Data Storage

On both nodes:

```
sudo tee /etc/clickhouse-server/config.d/storage.xml > /dev/null <<'EOF'
<?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>
EOF
```

8.5 Configure User Access

On both nodes:

```
sudo tee /etc/clickhouse-server/users.d/custom_users.xml > /dev/null
<<'EOF'
<?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_g
roup_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>

    <!-- 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>
EOF
```

8.6 Remove Default Password

On both nodes:

```
# Remove password file if it exists
sudo rm -f /etc/clickhouse-server/users.d/default-password.xml
```

Step 9: Start Services and Verify Cluster

9.1 Start ClickHouse Services

On both nodes:

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

# Enable auto-start
sudo systemctl enable clickhouse-server

# Check status
sudo systemctl status clickhouse-server --no-pager -l
```

9.2 Verify Service Status

On both nodes:

```
# Check ClickHouse process
ps aux | grep clickhouse

# Check listening ports
sudo netstat -tulpn | grep -E '(8123|9001|9009)'

# Check ZooKeeper status
sudo systemctl status zookeeper --no-pager -l
```

9.3 Test ClickHouse Connectivity

From Primary Node:

```
# Test local connection
clickhouse-client --port 9001 --query "SELECT version() as
clickhouse_version"

# Test HTTP interface
curl -s "http://localhost:8123/?query=SELECT%20version()"

# Test connection to secondary node
clickhouse-client --host clickhouse-secondary --port 9001 --query "SELECT
version() as clickhouse_version"

# Test secondary node HTTP interface
curl -s "http://clickhouse-secondary:8123/?query=SELECT%20version()"
```

9.4 Check Cluster Configuration

From Primary Node:

```

```

```
# Check cluster configuration
clickhouse-client --port 9001 --query "
SELECT
    cluster,
    shard_num,
    replica_num,
    host_name,
    port,
    user
FROM system.clusters
WHERE cluster LIKE '%cluster%'
ORDER BY cluster, shard_num, replica_num
"
```

Expected Output:

```
cluster_1shard_2replicas    1    1    clickhouse-primary    9001    default
cluster_1shard_2replicas    1    2    clickhouse-secondary  9001    default
cluster_2shards_1replicas   1    1    clickhouse-primary    9001    default
cluster_2shards_1replicas   2    1    clickhouse-secondary  9001    default
```

Step 10: Test Cluster Functionality

10.1 Create Databases

From Primary Node:

```
# Create database on both nodes
clickhouse-client --port 9001 --query "CREATE DATABASE cluster_db"
clickhouse-client --host clickhouse-secondary --port 9001 --query "CREATE
DATABASE cluster_db"

# Verify databases exist
clickhouse-client --port 9001 --query "SHOW DATABASES"
clickhouse-client --host clickhouse-secondary --port 9001 --query "SHOW
DATABASES"
```

10.2 Create Replicated Tables

From Primary Node:

```
# Create a replicated table
clickhouse-client --port 9001 --query "
CREATE TABLE cluster_db.replicated_table
(
    id UInt64,
```

```
    timestamp DateTime,
    message String,
    value Float64
)
ENGINE =
ReplicatedMergeTree('/clickhouse/tables/{shard}/cluster_db/replicated_table
', '{replica}')
ORDER BY id
PARTITION BY toYYYYMMDD(timestamp)
"

# Create a distributed table for queries
clickhouse-client --port 9001 --query "
CREATE TABLE cluster_db.distributed_table AS cluster_db.replicated_table
ENGINE = Distributed(cluster_2shards_1replicas, cluster_db,
replicated_table, rand())
"
```

10.3 Insert Test Data

From Primary Node:

```
# Insert test data
clickhouse-client --port 9001 --query "
INSERT INTO cluster_db.replicated_table VALUES
(1, now(), 'Hello from primary node', 100.5),
(2, now(), 'Cluster setup successful', 200.75),
(3, now(), 'Data replication test', 300.25)
"
```

10.4 Verify Data Replication

From Primary Node:

```
# Check data on primary
clickhouse-client --port 9001 --query "SELECT * FROM
cluster_db.replicated_table ORDER BY id"

# Check replication status
clickhouse-client --port 9001 --query "
SELECT
    database,
    table,
    is_leader,
    is_readonly,
    absolute_delay,
    queue_size
FROM system.replicas
```

```
WHERE database = 'cluster_db' AND table = 'replicated_table'
"
```

From Secondary Node:

```
# Check replicated data
clickhouse-client --host clickhouse-secondary --port 9001 --query "SELECT *
FROM cluster_db.replicated_table ORDER BY id"

# Test distributed queries
clickhouse-client --host clickhouse-secondary --port 9001 --query "SELECT *
FROM cluster_db.distributed_table ORDER BY id"
```

10.5 Test High Availability

Test Primary Node Failure:

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

   ```
   sudo systemctl stop clickhouse-server
   ```

2. **Query Secondary Node Only:**

   ```
   clickhouse-client --host clickhouse-secondary --port 9001 --query
   "SELECT COUNT(*) as total_rows FROM cluster_db.replicated_table"
   ```

3. **Restart Primary Node:**

   ```
   sudo systemctl start clickhouse-server
   ```

10.6 Final Verification

From Primary Node:

```
# Complete cluster health check
echo "=== Cluster Health Check ==="
echo "1. ClickHouse version on both nodes:"
clickhouse-client --port 9001 --query "SELECT 'Primary: ' || version() as
version"
clickhouse-client --host clickhouse-secondary --port 9001 --query "SELECT
'Secondary: ' || version() as version"

echo "2. Cluster configuration:"
```

```
clickhouse-client --port 9001 --query "SELECT COUNT(*) as
total_shard_replica_configs FROM system.clusters WHERE cluster LIKE
'%cluster%'"

echo "3. Data in replicated table:"
clickhouse-client --port 9001 --query "SELECT COUNT(*) as total_rows FROM
cluster_db.replicated_table"

echo "4. HTTP interfaces accessible:"
curl -s "http://clickhouse-primary:8123/?query=SELECT%20%27Primary%20OK%27"
curl -s "http://clickhouse-secondary:8123/?
query=SELECT%20%27Secondary%20OK%27"
```

Cluster Management

Starting the Cluster

```
# Start ZooKeeper first
sudo systemctl start zookeeper
ssh clickhouse-secondary 'sudo systemctl start zookeeper'

# Then start ClickHouse
sudo systemctl start clickhouse-server
ssh clickhouse-secondary 'sudo systemctl start clickhouse-server'
```

Stopping the Cluster

```
# Stop ClickHouse first
sudo systemctl stop clickhouse-server
ssh clickhouse-secondary 'sudo systemctl stop clickhouse-server'

# Then stop ZooKeeper
sudo systemctl stop zookeeper
ssh clickhouse-secondary 'sudo systemctl stop zookeeper'
```

Service Status Check

```
# Check all services
sudo systemctl status clickhouse-server zookeeper
ssh clickhouse-secondary 'sudo systemctl status clickhouse-server
zookeeper'

# Check processes
ps aux | grep -E "(clickhouse|zookeeper)"
```

Port Connectivity Check

```
# Check all required ports
echo "=== Primary Node Ports ==="
sudo netstat -tulpn | grep -E "(22|2181|8123|9001|9009)"

echo "=== Secondary Node Ports ==="
ssh clickhouse-secondary 'sudo netstat -tulpn | grep -E "
(22|2181|8123|9001|9009)"'
```

Troubleshooting

Common Issues and Solutions

1. ClickHouse Fails to Start

Symptoms: Service fails to start or shows errors in logs

Solutions:

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

# Check configuration files
clickhouse-server --config-file=/etc/clickhouse-server/config.xml --dry-run

# Check ZooKeeper connectivity
nc -zv clickhouse-primary 2181
nc -zv clickhouse-secondary 2181

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

2. ZooKeeper Ensemble Issues

Symptoms: ZooKeeper shows connection errors or ensemble not forming

Solutions:

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

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

```
# Verify server IDs
cat /var/lib/zookeeper/myid

# Check network connectivity between nodes
nc -zv clickhouse-primary 2888
nc -zv clickhouse-secondary 2888

# Restart ZooKeeper
sudo systemctl restart zookeeper
```

3. Replication Not Working

Symptoms: Data not replicating between nodes

Solutions:

```
# Check replication status
clickhouse-client --port 9001 --query "
SELECT database, table, is_leader, is_readonly, absolute_delay, queue_size
FROM system.replicas
"

# Check replication queue
clickhouse-client --port 9001 --query "
SELECT * FROM system.replication_queue
WHERE database = 'cluster_db' AND table = 'replicated_table'
"

# Force synchronization
clickhouse-client --port 9001 --query "
ALTER TABLE cluster_db.replicated_table SYNC REPLICA
"

# Check ZooKeeper connectivity from ClickHouse perspective
clickhouse-client --port 9001 --query "
SELECT * FROM system.zookeeper WHERE path = '/'
"
```

4. Network Connectivity Issues

Symptoms: Cannot connect between nodes

Solutions:

```
# Test all connections
nc -zv clickhouse-primary 8123
nc -zv clickhouse-primary 9001
nc -zv clickhouse-primary 9009
nc -zv clickhouse-secondary 8123
```

```
nc -zv clickhouse-secondary 9001
nc -zv clickhouse-secondary 9009

# Check firewall settings
sudo ufw status
sudo ufw allow 22/tcp
sudo ufw allow 8123/tcp
sudo ufw allow 9001/tcp
sudo ufw allow 9009/tcp
sudo ufw allow 2181/tcp
sudo ufw allow 2888/tcp
sudo ufw allow 3888/tcp

# Check hostname resolution
nslookup clickhouse-primary
nslookup clickhouse-secondary
ping -c 2 clickhouse-primary
ping -c 2 clickhouse-secondary
```

5. Memory Issues

Symptoms: Queries fail with memory errors

Solutions:

```
# Check memory usage
free -h
ps aux --sort=-%mem | head -10

# Monitor ClickHouse memory usage
clickhouse-client --port 9001 --query "
SELECT
    metric,
    formatReadableSize(value) as memory_usage
FROM system.metrics
WHERE metric LIKE '%memory%'
"

# Adjust memory limits in configuration
sudo nano /etc/clickhouse-server/config.d/cluster.xml
# Add: <max_memory_usage>20000000000</max_memory_usage>

# Restart ClickHouse after configuration changes
sudo systemctl restart clickhouse-server
```

Log Locations

ClickHouse Logs:

- Server log: `/var/log/clickhouse-server/clickhouse-server.log`

- Error log: `/var/log/clickhouse-server/clickhouse-server.err.log`

ZooKeeper Logs:

- ZooKeeper log: `/opt/zookeeper/logs/zookeeper.log`

System Logs:

- ClickHouse service: `journalctl -u clickhouse-server`
- ZooKeeper service: `journalctl -u zookeeper`

Useful Commands

```
# Real-time log monitoring
sudo tail -f /var/log/clickhouse-server/clickhouse-server.log

# Check recent system errors
journalctl -u clickhouse-server --since "1 hour ago" -p err

# Check cluster health
clickhouse-client --port 9001 --query "
SELECT
    cluster,
    shard_num,
    replica_num,
    host_name,
    port,
    case when error = 0 then 'OK' else 'ERROR' end as status
FROM system.clusters
WHERE cluster LIKE '%cluster%'
"

# Monitor query performance
clickhouse-client --port 9001 --query "
SELECT
    query,
    query_duration_ms/1000 as duration_seconds,
    memory_usage,
    result_rows,
    formatReadableSize(memory_usage) as memory_formatted
FROM system.query_log
WHERE event_time > now() - INTERVAL 1 HOUR
  AND type = 'QueryFinish'
ORDER BY query_duration_ms DESC
LIMIT 10
"
```

Performance Tuning

Memory Optimization

```
<!-- Add to /etc/clickhouse-server/config.d/performance.xml -->
<?xml version="1.0"?>
<clickhouse>
    <max_memory_usage>20000000000</max_memory_usage>
    <max_memory_usage_for_user>10000000000</max_memory_usage_for_user>

<max_bytes_before_external_group_by>4000000000</max_bytes_before_external_g
roup_by>

<max_bytes_before_external_sort>4000000000</max_bytes_before_external_sort>
    <max_insert_threads>0</max_insert_threads>
    <max_threads>8</max_threads>
</clickhouse>
```

Merge Tree Optimization

```
<!-- Add to cluster.xml -->
<merge_tree>
    <max_suspicious_broken_parts>10</max_suspicious_broken_parts>

<max_bytes_to_merge_at_max_space_in_pool>20971520000</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>
    <max_replicated_merges_in_queue>16</max_replicated_merges_in_queue>
</merge_tree>
```

Network Optimization

```
<!-- Add to network.xml -->
<tcp_keepalive_timeout>3</tcp_keepalive_timeout>
<tcp_keepalive_interval>1</tcp_keepalive_interval>
<connect_timeout_with_failover_ms>10000</connect_timeout_with_failover_ms>
<max_connections>4096</max_connections>
```

Monitoring and Maintenance

System Monitoring Commands

```
# Monitor system resources
htop
iotop
nethogs

# Monitor ClickHouse processes
```

```
ps aux | grep clickhouse

# Monitor ZooKeeper processes
ps aux | grep zookeeper

# Check disk space
df -h
du -sh /var/lib/clickhouse/
```

ClickHouse Monitoring Queries

```
-- Cluster status
SELECT
    cluster,
    shard_num,
    replica_num,
    host_name,
    port
FROM system.clusters
WHERE cluster LIKE '%cluster%';

-- Table sizes
SELECT
    database,
    table,
    formatReadableSize(sum(bytes)) as size,
    sum(rows) as total_rows,
    count() as parts_count
FROM system.parts
WHERE active = 1
GROUP BY database, table
ORDER BY sum(bytes) DESC;

-- Replication status
SELECT
    database,
    table,
    is_leader,
    is_readonly,
    absolute_delay,
    queue_size,
    absolute_delay > 60 as has_lag
FROM system.replicas;

-- Query performance
SELECT
    query,
    query_duration_ms/1000 as duration_seconds,
    memory_usage,
    result_rows,
    formatReadableSize(memory_usage) as memory_formatted
```

```
FROM system.query_log
WHERE event_time > now() - INTERVAL 1 HOUR
  AND type = 'QueryFinish'
ORDER BY query_duration_ms DESC
LIMIT 20;
```

Backup Strategies

```
-- Create backup of specific table
BACKUP TABLE cluster_db.replicated_table
TO Disk('backups', 'replicated_table_backup_' + toString(now(),
'%Y%m%d_%H%M%S'));

-- Create backup of entire database
BACKUP DATABASE cluster_db
TO Disk('backups', 'cluster_db_backup_' + toString(now(),
'%Y%m%d_%H%M%S'));

-- Restore from backup
RESTORE TABLE cluster_db.replicated_table
FROM Disk('backups', 'replicated_table_backup_20241027_143000');
```

Maintenance Tasks

```
# Create backup directory
sudo mkdir -p /var/lib/clickhouse/backups
sudo chown clickhouse:clickhouse /var/lib/clickhouse/backups

# Schedule regular backups (add to crontab)
echo "0 2 * * * clickhouse-client --port 9001 --query \"BACKUP DATABASE
cluster_db TO Disk('backups', 'cluster_db_daily_' + toString(now(),
'%Y%m%d'))\"" | sudo crontab -

# Cleanup old backups (keep last 7 days)
echo "0 3 * * * find /var/lib/clickhouse/backups -name '*.backup' -mtime +7
-delete" | sudo crontab -
```

Security Configuration

Firewall Setup

```
# Configure UFW firewall
sudo ufw --force reset
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

```
# Allow SSH
sudo ufw allow 22/tcp

# Allow ClickHouse ports from specific IPs if needed
sudo ufw allow from 10.149.252.0/24 to any port 8123
sudo ufw allow from 10.149.252.0/24 to any port 9001
sudo ufw allow from 10.149.252.0/24 to any port 9009

# Allow ZooKeeper ports within cluster
sudo ufw allow from 10.149.252.0/24 to any port 2181
sudo ufw allow from 10.149.252.0/24 to any port 2888
sudo ufw allow from 10.149.252.0/24 to any port 3888

# Enable firewall
sudo ufw --force enable
```

User Authentication

```
<!-- Add to users.d/security.xml -->
<?xml version="1.0"?>
<clickhouse>
    <users>
        <admin>
            <password>secure_admin_password</password>
            <networks>
                <ip>127.0.0.1</ip>
                <ip>10.149.252.0/24</ip>
            </networks>
            <profile>default</profile>
            <quota>default</quota>
        </admin>

        <readonly_user>
            <password>readonly_password</password>
            <networks>
                <ip>10.149.252.0/24</ip>
            </networks>
            <profile>readonly</profile>
            <quota>default</quota>
        </readonly_user>
    </users>

    <profiles>
        <readonly>
            <max_memory_usage>1000000000</max_memory_usage>
            <max_execution_time>60</max_execution_time>
            <readonly>1</readonly>
        </readonly>
    </profiles>
</clickhouse>
```

Success Criteria

Your ClickHouse 2-node cluster is successfully configured when:

✅ **Services Running:**

- Both ZooKeeper services are running (one leader, one follower)
- Both ClickHouse services are running without errors
- All required ports are listening on both nodes

✅ **Network Connectivity:**

- Passwordless SSH works between nodes
- Hostnames resolve correctly on both nodes
- ClickHouse HTTP interfaces are accessible
- ZooKeeper connectivity works from both nodes

✅ **Cluster Functionality:**

- Cluster configuration appears in `system.clusters`
- Replicated tables can be created and data replicates
- Distributed queries work across both nodes
- High availability is maintained when one node is restarted

✅ **Web Interfaces Accessible:**

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

References and Resources

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