Cloning a Service to Another Provider or Region

Migrating or cloning **ClickHouse** across cloud providers or geographic regions is essential for optimizing performance, meeting compliance requirements, or ensuring high availability. ClickHouse, being a distributed columnar OLAP database, introduces some unique considerations due to its architecture of shards and replicas. A well-planned migration ensures data consistency, system integrity, and minimal downtime.

Pre-Migration Preparation

Before initiating a ClickHouse migration, it is critical to plan for both the data layout and cluster topology:

- **Evaluate the Current Setup:** Document the existing ClickHouse configuration, including cluster layout (shards and replicas), table schemas (especially ReplicatedMergeTree tables), user roles, ZooKeeper (or ClickHouse Keeper) setup, and storage configurations. Identify custom functions, dictionaries, and any external dependencies like Kafka or S3.
- **Define the Migration Target:** Choose the new region or cloud provider. Ensure the target environment supports similar storage and compute characteristics. Plan how the new cluster will be laid out—same shard/replica pattern or adjusted topology. If using cloud-native services (e.g., Elestio), verify feature parity.
- **Provision the Target Environment:** Deploy the ClickHouse nodes with required hardware specs (high IOPS disks, sufficient RAM/CPU). Set up coordination services (ZooKeeper or ClickHouse Keeper) and prepare the cluster topology in configuration files (remote_servers.xml, zookeeper.xml, etc.).
- **Backup the Current Cluster:** Use ClickHouse's built-in backup tools (BACKUP and RESTORE SQL commands, or clickhouse-backup utility) to create consistent snapshots. Ensure backups include both schema and data. Store backups on cloud-agnostic storage (e.g., S3) for ease of access during restoration.

Cloning Execution

To begin cloning ClickHouse, replicate the original cluster's configuration in the new environment, ensuring that the shard and replica layout, coordination services (ZooKeeper or ClickHouse Keeper), and access controls are all set up identically. This includes copying configuration files such as users.xml, remote_servers.xml, and zookeeper.xml, and verifying that all inter-node communication is functional.

For table data, use ClickHouse's native BACKUP and RESTORE SQL commands or the clickhouse-backup utility, ideally in combination with cloud object storage like S3 for efficient parallel upload and download. When restoring, ensure that ReplicatedMergeTree tables use new, unique ZooKeeper paths to avoid replication conflicts with the original cluster. In the case of non-replicated tables, manual data export and import (e.g., using INSERT SELECT or clickhouse-client --query) may be necessary.

After the data and schema have been restored, perform thorough validation by running sample queries, verifying performance against expected baselines, and inspecting logs for errors. Finally, ensure all integrations (e.g., Kafka pipelines, distributed tables, user-defined functions) are functional and fully consistent with the original service before proceeding to production traffic cutover.

Cutover and DNS/Traffic Switch

Once the new ClickHouse cluster has been validated, you can proceed with the traffic cutover. Update your application's client connection strings, service discovery endpoints, or load balancer configurations to direct requests to the new cluster. If you're using DNS-based routing, update A or CNAME records accordingly, taking into account DNS propagation times.

For setups requiring high availability or a gradual transition, consider using weighted DNS records or a load balancer with health checks to route a portion of traffic to the new cluster while monitoring its performance. Ensure that all downstream applications, dashboards, and data pipelines are updated with the new endpoints and credentials. If feasible, maintain the old cluster temporarily as a fallback until the new environment is confirmed stable in production.

Post-Migration Validation and Optimization

- **Validate Application Workflows:** Test analytics dashboards, queries, and data pipelines against the new cluster. Ensure integrations (e.g., Grafana, Kafka consumers, exporters) are fully functional.
- **Monitor Performance:** Use ClickHouse's system.metrics and system.events tables to monitor performance. Validate disk space usage, query latency, and background merges. Adjust settings like max_threads, merge_max_size, or background_pool_size for the new environment.
- **Secure the Environment:** Reapply user and role settings with secure password policies. Enable TLS for inter-node and client communications. Restrict access using firewalls, IP allowlists, and RBAC.
- **Cleanup and Documentation:** Decommission the old cluster only after full confidence in the new setup. Document changes in configuration, node addresses, backup schedules, and operational runbooks.

Benefits of Cloning ClickHouse

Cloning a ClickHouse cluster provides several operational and strategic benefits. It allows teams to test version upgrades, schema changes, and application features on production-like data without impacting live systems. Maintaining a cloned cluster in a separate region or cloud provider also enables robust disaster recovery by providing a ready-to-promote standby.

For organizations with strict compliance or analytics needs, clones can serve as read-only environments for querying and reporting without risking the integrity of live data. Additionally, cloning simplifies cloud migrations by replicating the entire setup schema, configuration, and data into a new environment, thereby minimizing downtime, reducing manual setup, and accelerating cutover with high confidence.

Revision #1
Created 9 June 2025 09:57:35 by kaiwalya
Updated 9 June 2025 10:05:39 by kaiwalya