Installing and Updating an Extension

ClickHouse supports **custom extensions** via [User Defined Functions (UDFs)], external dictionaries, and shared libraries that extend its core capabilities with custom logic, formats, or integrations. These behave similarly to modules or plugins in other systems and must be configured at server startup. Common examples include integration with geospatial libraries, custom UDFs, or external dictionary sources like MySQL or HTTP.

In Elestio-hosted ClickHouse instances or any Docker Compose-based setup, extensions can be added by mounting external libraries or configuration files and referencing them in config.xml or users.xml. This guide walks through how to install, load, and manage ClickHouse extensions using Docker Compose along with best practices and common troubleshooting steps.

Installing and Enabling ClickHouse Extensions

ClickHouse extensions are typically compiled as shared objects (.so) files or defined as configuration files for dictionaries or formats. These files must be mounted into the container and referenced explicitly in the server's configuration files.

Example: Load Custom Shared Library UDF

Suppose you have a compiled UDF called `libexample_udf.so`. To include it in a Docker Compose setup:

Update docker-compose.yml

Mount the shared library into the container:

```
services:
  clickhouse:
    image: clickhouse/clickhouse-server:latest
    volumes:
      - ./modules/libexample_udf.so:/usr/lib/clickhouse/user_defined/libexample_udf.so
```

```
      - ./configs/config.xml:/etc/clickhouse-server/config.xml
    ports:
      - "8123:8123"
      - "9000:9000"
```

- `./modules/libexample_udf.so`: local path to the shared library on the host.
- `/usr/lib/clickhouse/user_defined/`: default directory for user libraries inside the container.

Make sure the file exists before running Docker Compose.

Configure config.xml to Load the UDF

In your custom config.xml:

```
<user_defined>
    <function>
        <name>example_udf</name>
        <type>udf</type>
        <library>libexample_udf.so</library>
    </function>
</user_defined>
```

> The library path must match the volume mount location.

Restart the ClickHouse Service

After updating the Compose and configuration files, restart the service:

```
docker-compose down
docker-compose up -d
```

This will reload ClickHouse with the specified UDF.

Verify the Extension is Loaded

Connect using the ClickHouse CLI or HTTP interface and run:

```
SELECT example_udf('test input');
```

If successful, the function will return expected results from the loaded library. You can also confirm the server loaded your shared library by inspecting logs:

```
docker-compose logs clickhouse
```

Look for lines that indicate the library was found and loaded.

Managing External Dictionaries

ClickHouse supports loading external data sources (like MySQL, HTTP APIs, or files) as dictionaries

Mount Dictionary Configuration

In docker-compose.yml:

```
volumes:
  - ./configs/dictionaries/:/etc/clickhouse-server/dictionaries/
```

Reference in config.xml

```
<dictionaries_config>/etc/clickhouse-server/dictionaries/*.xml</dictionaries_config>
```

Example dictionary file (mysql_dictionary.xml):

```
<dictionary>
  <name>mysql_dict</name>
  <source>
    <mysql>
      <host>mysql-host</host>
      <user>root</user>
      <password>password</password>
      <db>test</db>
      <table>cities</table>
    </mysql>
  </source>
  <layout><flat /></layout>
  <structure>
    <id>id</id>
    <attribute>
      <name>name</name>
      <type>String</type>
    </attribute>
```

```
  </structure>
</dictionary>
```

Use the dictionary in queries:

```
SELECT dictGetString('mysql_dict', 'name', toUInt64(42));
```

Updating or Removing Extensions

ClickHouse doesn’t support unloading UDFs or dictionaries at runtime. To modify or remove an extension:

1. **Stop the container**:

```
docker-compose down
```

2. Edit config files:

- Replace or remove the <function> entry in config.xml or dictionary config.
- Replace or remove the .so file if applicable.

3. Restart the container:

```
docker-compose up -d
```

> Always test changes in staging before deploying to production.

Troubleshooting Common Extension Issues

Issue	Cause	Resolution
ClickHouse fails to start	Invalid config or missing .so file	Run docker-compose logs clickhouse and fix missing files or XML syntax

Issue	Cause	Resolution
UDF not recognized	Wrong library path or missing permissions	Ensure volume mount is correct and file is executable inside container
Dictionary not available	Config file not found or misconfigured XML	Double-check dictionaries_config and validate with SHOW DICTIONARIES
Segmentation fault	Invalid shared library or ABI mismatch	Recompile UDF for correct platform, verify against installed ClickHouse version
Query fails silently	Dictionary or UDF not fully loaded	Recheck server logs for errors during startup

Security Considerations

ClickHouse extensions especially shared libraries run with the same privileges as the ClickHouse server. Be cautious:

- Only load trusted .so files from verified sources.
- Ensure clickhouse user has restricted permissions inside the container.
- Never expose dictionary or UDF paths to writable directories from external systems.

Avoid using custom UDFs or dictionaries from unknown sources in production environments without a thorough code review.

Revision #1
Created 2025-06-11 06:13:08 UTC
Updated 2025-06-11 06:19:40 UTC