Guia de Configuração ODBC: ClickHouse → SQL Server

📋 Pré-requisitos

Certifique-se de ter os seguintes arquivos na raiz do seu projeto:

- `Dockerfile-clickhouse-odbc`
- `odbc.ini`
- `odbcinst.ini`
- `docker-compose.yml`

🚀 Passo a Passo

1. Criar a Estrutura de Diretórios

```bash
mkdir -p clickhouse/config
```

2. Copiar Arquivo de Configuração (Opcional)

Se quiser customizar o ClickHouse, crie o arquivo de configuração:

```bash
cp clickhouse-odbc-config.xml clickhouse/config/odbc.xml
```

3. Atualizar o docker-compose.yml

Atualize a seção do ClickHouse no seu `docker-compose.yml`:

```yaml
```

```
clickhouse:
  build:
    context: .
    dockerfile: Dockerfile-clickhouse-odbc
  container_name: clickhouse
  restart: unless-stopped
  ports:
    - "8123:8123"
    - "9000:9000"
  volumes:
    - clickhouse_data:/var/lib/clickhouse
    - ./clickhouse/config:/etc/clickhouse-server/config.d
  environment:
    CLICKHOUSE_USER: default
    CLICKHOUSE_PASSWORD: admin123
    CLICKHOUSE_DB: default
    TZ: America/Sao_Paulo
  ulimits:
    nofile:
      soft: 262144
      hard: 262144
  networks:
    - bigdata-net
  depends_on:
    - sqlserver
```

4. Verificar Senha no odbc.ini

IMPORTANTE: Certifique-se de que a senha no `odbc.ini` corresponde à senha do SQL Server no `docker-compose.yml`:

- SQL Server: `@Admin123`
- odbc.ini: `@Admin123` ✅

5. Rebuild e Start dos Containers

```bash
```

```bash
# Parar containers existentes
docker-compose down

# Rebuild da imagem do ClickHouse
docker-compose build clickhouse

# Iniciar todos os serviços
docker-compose up -d

# Verificar logs do ClickHouse
docker-compose logs -f clickhouse
```

🧪 Testando a Conexão

Teste 1: Verificar Instalação ODBC

```bash
docker exec -it clickhouse bash

# Verificar configuração ODBC
odbcinst -j

# Listar DSNs disponíveis
odbcinst -q -s

# Testar conexão ODBC diretamente
isql -v sqlserver_asprod
```

Teste 2: Testar no ClickHouse Client

```bash
# Conectar ao ClickHouse
docker exec -it clickhouse clickhouse-client --user=default --password=admin123

# Executar comandos SQL de teste
```

Use os comandos do arquivo `test-odbc-connection.sql` para testar.

Teste 3: Consulta Simples via ODBC

```sql
```

```
-- Criar tabela temporária para testar
SELECT * FROM odbc('DSN=sqlserver_asprod', 'SELECT @@VERSION as version');

-- Listar databases do SQL Server
SELECT * FROM odbc('DSN=sqlserver_asprod', 'SELECT name FROM sys.databases');

-- Listar tabelas
SELECT * FROM odbc('DSN=sqlserver_asprod', 'SELECT name FROM sys.tables');
```

🔧 Resolução de Problemas

Erro: "Cannot open shared object file"

```bash
# Verificar se o driver está instalado
docker exec -it clickhouse ls -la /opt/microsoft/msodbcsql18/lib64/

# Verificar arquivo correto
docker exec -it clickhouse find /opt -name "libmsodbcsql*"
```

Erro: "Connection refused"

1. Verificar se o SQL Server está rodando:

```bash
docker-compose ps sqlserver
```

2. Testar conexão de rede:

```bash
docker exec -it clickhouse ping sqlserver
docker exec -it clickhouse telnet sqlserver 1433
```

3. Verificar logs do SQL Server:

```bash
docker-compose logs sqlserver
```

Erro: "Login failed for user 'sa'"

1. Confirmar senha no docker-compose.yml
2. Aguardar o SQL Server inicializar completamente (pode levar 1-2 minutos)

3. Resetar senha se necessário:

```bash
docker exec -it sqlserver /opt/mssql-tools/bin/sqlcmd -S localhost -U sa -P '@Admin123' -Q "ALTER LOGIN sa WITH PASS
```

Debug Geral

```bash
# Ver variáveis de ambiente ODBC
docker exec -it clickhouse env | grep ODBC

# Testar conexão com isql
docker exec -it clickhouse isql -v sqlserver_asprod

# Verificar logs do ClickHouse
docker-compose logs clickhouse | grep -i odbc

# Habilitar trace ODBC (editar odbc.ini)
# Trace=Yes
# TraceFile=/tmp/odbc.log
docker exec -it clickhouse cat /tmp/odbc.log
```

📊 Exemplo Prático: ETL do SQL Server para ClickHouse

```sql
```

```sql
-- 1. Criar tabela de origem no ClickHouse apontando para SQL Server
CREATE TABLE staging.fonte_sqlserver
ENGINE = ODBC('DSN=sqlserver_asprod', 'MeuBanco', 'MinhaTabela');

-- 2. Criar tabela de destino otimizada no ClickHouse
CREATE TABLE analytics.minha_tabela_analise
(
    id UInt64,
    nome String,
    valor Decimal(18,2),
    data_criacao DateTime,
    data_atualizacao DateTime DEFAULT now()
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(data_criacao)
ORDER BY (id, data_criacao);

-- 3. Importar dados
INSERT INTO analytics.minha_tabela_analise
SELECT
    id,
    nome,
    valor,
    data_criacao
FROM staging.fonte_sqlserver
WHERE data_criacao >= today() - INTERVAL 30 DAY;

-- 4. Verificar importação
SELECT
    count() as total_registros,
    min(data_criacao) as data_inicio,
    max(data_criacao) as data_fim
FROM analytics.minha_tabela_analise;
```

🎯 Próximos Passos

1. **Configurar CDC** com Kafka Connect para streaming em tempo real
2. **Criar Materialized Views** para agregações automáticas
3. **Configurar Scheduled Queries** para sincronização periódica
4. **Implementar Particionamento** para otimizar queries grandes
5. **Configurar Replicação** se precisar de alta disponibilidade

📝 Notas Importantes

- O ClickHouse ODBC tem melhor performance com queries que retornam muitos dados
- Para pequenas queries frequentes, considere usar CDC + Kafka
- Sempre use filtros WHERE para limitar a quantidade de dados transferidos
- Monitore o uso de memória durante importações grandes
- Configure timeout adequado para queries longas

🔗 Referências

- ClickHouse ODBC Documentation
- Microsoft ODBC Driver for SQL Server
- UnixODBC Configuration