Cachalot DB - version 2.5

Administration Guide

Introduction

In this document you will find:

- A description of the distribution package content
- Description of the configuration file
- Explanation on the inner working of the server and the related files
- Presentation of the new monitoring application
- Considerations on backup/restore and data migration procedures.

The distribution package.

Each release contains three zip files.

- **Platform dependent package**: it works on any compatible system if the dotnet runtime is installed.
- **Windows64 package**: It works on any recent windows 64bits system as a standalone application. Dotnet runtime is not required.
- **Linux64 package**: It works on Linux 64bits distributions as a standalone application. Before any release, it is extensively evaluated on the last LTS distribution on Ubuntu and Red Hat.

As it is an open-source project (MIT license), feel free to compile it for other platforms.

Inside each package there are three folders:

- **Doc**: containing three pdf files: **UserGuide**, **SQLandLINQguide**, **AdministrationGuide** (this one)
- **Server**: containing all the binaries required to install a server, the default configuration file, and a CSV import tool
- **Monitoring**: the new graphical monitoring tool

When deploying a Cachalot server (usually as a node in a multi-server cluster), copy the content of the “Server” folder from the distribution package to your installation folder.

On Windows, three applications are available: **Server.exe**, **WindowsService.exe** and **CSVImport.exe**. Server and WindowsService contain the same code packaged as a console application and a windows service. The console application can be started directly, the service must be installed using the provided **install.cmd** script. Administrative privileges are required to install the service. Once installed, it may be configured and started using the "Services" control panel.

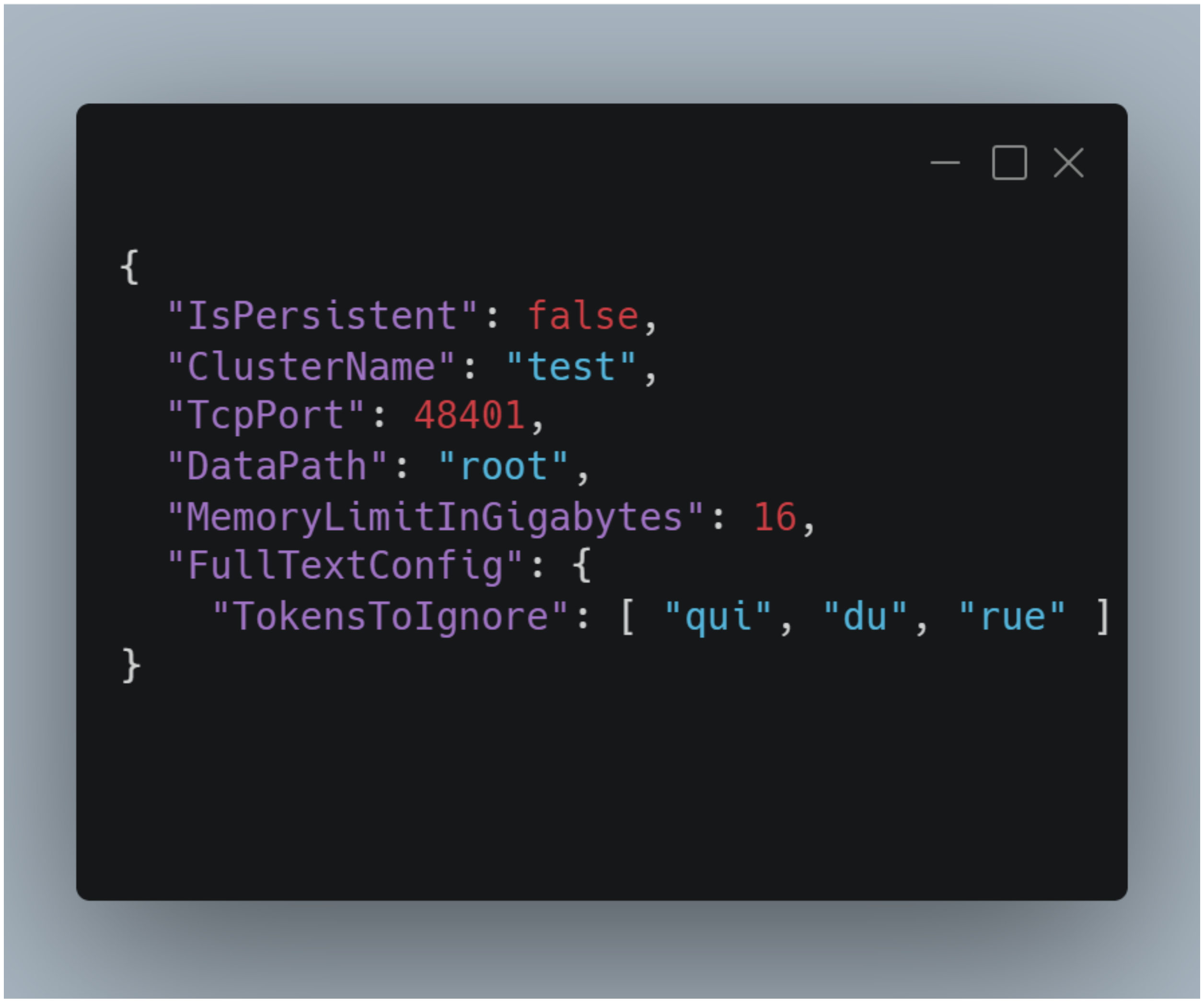
On Linux, the application binaries are **Server** and **CSVImport**. The server can be either run directly or registered as a service with the "systemd."

After decompressing the zip package on Linux, use chmod to add the execution rights on these two files.

The configuration file.

Running **Server.exe** without a command line parameter will read its configuration from the file **node_config.json**. When we specify a parameter, it will be interpreted as a suffix to the name of the configuration file.

For example, “**Server 01**” will use the config file **node_config_01.json**.



```
{
  "IsPersistent": false,
  "ClusterName": "test",
  "TcpPort": 48401,
  "DataPath": "root",
  "MemoryLimitInGigabytes": 16,
  "FullTextConfig": {
    "TokensToIgnore": [ "qui", "du", "rue" ]
}
```

Figure 1 A typical configuration file

Is Persistent:

When true, it works in database mode, otherwise as a distributed cache.

Cluster Name:

Used for monitoring only, and it needs to be the same for all the cluster nodes.

TCP port:

The port. It needs to be unique on a machine.

Data Path:

If not present, the directory is automatically created when server starts. It contains two sub-directories: “data” and “logs.”

Full Text Config:

It is an optional list of tokens to ignore to speed up the full-text search. See the USERGUIDE for details.

Memory Limit In Gigabytes:

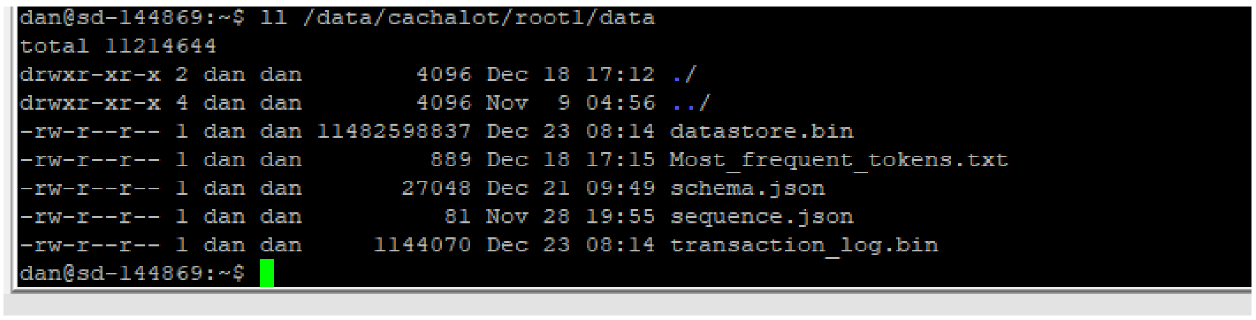
Mostly used for monitoring. All nodes in a cluster should have the same value.

The files in the “data” directory

In cache mode (non-persistent), the data directory contains only the **schema.json** file which defines the schema for each collection. See the USERGUIDE for an explanation on the schema.

In persistent mode, “data” directory contains all the persistent data:

- The append-only transaction log
    - All changes are **synchronously** appended to this file before being applied to the memory.
- The permanent storage
    - All the events from the transaction log are **asynchronously** applied to this file by a background thread.
- The “sequence” file containing the last values for the unique key generators.



```
dan@sd-144869:~$ ll /data/cachalot/root1/data
total 11214644
drwxr-xr-x 2 dan dan        4096 Dec 18 17:12 ./
drwxr-xr-x 4 dan dan        4096 Nov  9 04:56 ../
-rw-r--r-- 1 dan dan 11482598837 Dec 23 08:14 datastore.bin
-rw-r--r-- 1 dan dan         889 Dec 18 17:15 Most_frequent_tokens.txt
-rw-r--r-- 1 dan dan       27048 Dec 21 09:49 schema.json
-rw-r--r-- 1 dan dan          81 Nov 28 19:55 sequence.json
-rw-r--r-- 1 dan dan     1144070 Dec 23 08:14 transaction_log.bin
dan@sd-144869:~$
```

Figure 2 The content of the data folder

When the server starts on a non-empty database, it does some cleanup operations before accepting user connections:

- It applies all the pending transactions to the permanent storage.
- Then it compactifies the transaction log.
- Then it cleans up the datastore by removing deleted and dirty records.
    - Dirty record = a record in the permanent storage that is not used anymore as the object it contained was updated, its size has grown, and it moved to the end of the file

Simply restarting a server can release lots of disk space.

The monitoring application

This is a new component of the ecosystem available since the 2.5 version.

It consists of a self-contained web application. It includes a web server, the backend service, and the UI.

You can have a look without installing it at https://cachalot-db.com/. This is a showcase version that only allows the connection to predetermined list of clusters.

It can be run locally, deployed as-is or behind a reverse proxy (nginx for example).

The executable file is **CachalotMonitor.exe** on Windows and **CachalotMonitor** on Linux.

Comprehensive tooltips are available on each control, so we will not go into every detail here.

For security reasons, by default you connect as “guest.” No action that modifies data or schema is available. You can only examine the collections and their layout, schemas, and query data.

When installing it from the package, the “admin” password is not set. The first one you use will be accepted and will become the admin password that is checked for the new connections.

To reset the admin password, delete the file **user_hashes.txt** and restart the monitoring application.

Brief description of the pages in the monitoring application

1) The “connection” tab:
    a. Allows to connect to a cluster by providing the explicit list of servers and ports.
    b. Or connect to a cluster you previously connected to, by a single click.
2) The “collections” tab:
    a. Visualize the collections. For each one you cans see:
        i. The layout
        ii. The number of items in the collection
        iii. The eviction type (applies only to non-persistent collections)
        iv. If full-text search is available for the given collection
3) The “schema” tab:
    a. The list of query able properties on each collection and the type on index
    b. In admin mode indexes can be upgraded (from None to Dictionary or from Dictionary to Ordered)
4) The “data” tab:
    a. Query your data.
    b. Visualize your data as table (only query able properties) or as Json (full document).
    c. Examine the execution plan.
    d. Export or import data as Json.
    e. Delete specific data.
5) The “admin” tab: common administrative tasks as
    a. Backup/Restore/Recreate (more on these in the next chapter)
    b. Drop database.
    c. Switch on/off the read-only mode.
    d. Truncate or drop a collection.

Backup / restore and migration procedure.

Backup and restore are designed to be as fast as possible. To achieve this, every node in a cluster saves and restores data in parallel. The client only triggers the process.

As we want all the node backups to be stored in the same place (normally on a different server than the cluster), the backup directory must be a shared network drive mapped under the same path for all the servers in the cluster.

The cluster that produces the backup and the one that restores it must have the same number of nodes.

To transfer data between two clusters with different configurations, another function is available: “Recreate.” In this case, the client reads the backup and feeds all data to the cluster. It is slower than “Restore”; data will be redistributed between the nodes of the target cluster.

“Recreate“ does not remove data from the target database. It can be used to merge data from different clusters.

It you need to reinitialize an existing cluster drop the database before.

When restoring a backup, every node in the cluster performs a “soft restart.” The process is not terminated but all data (and schema information) is deleted and the server goes through the same initialization process as a normal start.

The cluster is not available during restore.

While running a backup the cluster is still available for read-only operations. All write actions will be queued and executed after.

If the database is updated frequently switching it to read-only mode before starting the backup is a guarantee to produce a consistent backup for the whole cluster.