(/)

OFFICIAL REPOSITORY

cassandra (/r/_/cassandra/) ☆

Last pushed: 6 days ago

Repo Info (/_/cassandra/)

Short Description

Apache Cassandra is an open-source distributed storage system.

Full Description

Supported tags and respective `Dockerfile` links

- `2.1.20`, `2.1` (*2.1/Dockerfile*) (https://github.com/docker-library/cassandra/blob/5c58a3353bb4d533f78bf809f1e55ea0b67b4ba5/2.1/Dockerfile)
- `2.2.13`, `2.2`, `2` (*2.2/Dockerfile*) (https://github.com/docker-library/cassandra/blob/5c58a3353bb4d533f78bf809f1e55ea0b67b4ba5/2.2/Dockerfile)
- `3.0.17`, `3.0` (*3.0/Dockerfile*) (https://github.com/docker-library/cassandra/blob/5c58a3353bb4d533f78bf809f1e55ea0b67b4ba5/3.0/Dockerfile)
- `3.11.3`, `3.11`, `3`, `latest` (*3.11/Dockerfile*) (https://github.com/docker-library/cassandra/blob/5c58a3353bb4d533f78bf809f1e55ea0b67b4ba5/3.11/Dockerfile)

Quick reference

- **Where to get help**:
  the Docker Community Forums (https://forums.docker.com/), the Docker Community Slack (https://blog.docker.com/2016/11/introducing-docker-community-directory-docker-community-slack/), or Stack Overflow (https://stackoverflow.com/search?tab=newest&q=docker)

- **Where to file issues**:
  https://github.com/docker-library/cassandra/issues (https://github.com/docker-library/cassandra/issues)

- **Maintained by**:
  the Docker Community (https://github.com/docker-library/cassandra)

- **Supported architectures**: (more info (https://github.com/docker-library/official-images#architectures-other-than-amd64))
  `amd64` (https://hub.docker.com/r/amd64/cassandra/), `arm64v8` (https://hub.docker.com/r/arm64v8/cassandra/), `i386` (https://hub.docker.com/r/i386/cassandra/), `ppc64le` (https://hub.docker.com/r/ppc64le/cassandra/)

- **Published image artifact details**:
  repo-info repo's `repos/cassandra/` directory (https://github.com/docker-library/repo-info/blob/master/repos/cassandra) (history (https://github.com/docker-library/repo-info/commits/master/repos/cassandra))
  (image metadata, transfer size, etc)

- **Image updates**:
  official-images PRs with label `library/cassandra` (https://github.com/docker-library/official-images/pulls?q=label%3Alibrary%2Fcassandra)
  official-images repo's `library/cassandra` file (https://github.com/docker-library/official-images/blob/master/library/cassandra) (history (https://github.com/docker-library/official-images/commits/master/library/cassandra))

- **Source of this description**:
  docs repo's `cassandra/` directory (https://github.com/docker-library/docs/tree/master/cassandra) (history (https://github.com/docker-library/docs/commits/master/cassandra))

- **Supported Docker versions**:
  the latest release (https://github.com/docker/docker-ce/releases/latest) (down to 1.6 on a best-effort basis)

What is Cassandra?

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients.

> wikipedia.org/wiki/Apache_Cassandra
> (https://en.wikipedia.org/wiki/Apache_Cassandra)

How to use this image

Start a `cassandra` server instance

Starting a Cassandra instance is simple:

```
$ docker run --name some-cassandra -d cassandra:tag
```

... where `some-cassandra` is the name you want to assign to your container and `tag` is the tag specifying the Cassandra version you want. See the list above for relevant tags.

Connect to Cassandra from an application in another Docker container

This image exposes the standard Cassandra ports (see the Cassandra FAQ (https://wiki.apache.org/cassandra/FAQ#ports)), so container linking makes the Cassandra instance available to other application containers. Start your application container like this in order to link it to the Cassandra container:

```
$ docker run --name some-app --link some-cassandra:cassandra -d app
```

Make a cluster

Using the environment variables documented below, there are two cluster scenarios: instances on the same machine and instances on separate machines. For the same machine, start the instance as described above. To start other instances, just tell each new node where the first is.

```
$ docker run --name some-cassandra2 -d -e CASSANDRA_SEEDS="$(docker
```

... where `some-cassandra` is the name of your original Cassandra Server container, taking advantage of `docker inspect` to get the IP address of the other container.

Or you may use the docker run --link option to tell the new node where the first is:

```
$ docker run --name some-cassandra2 -d --link some-cassandra:cassan
```

For separate machines (ie, two VMs on a cloud provider), you need to tell Cassandra what IP address to advertise to the other nodes (since the address of the container is behind the docker bridge).

Assuming the first machine's IP address is `10.42.42.42` and the second's is `10.43.43.43`, start the first with exposed gossip port:

```
$ docker run --name some-cassandra -d -e CASSANDRA_BROADCAST_ADDRESS
```

Then start a Cassandra container on the second machine, with the exposed gossip port and seed pointing to the first machine:

```
$ docker run --name some-cassandra -d -e CASSANDRA_BROADCAST_ADDRESS
```

Connect to Cassandra from `cqlsh`

The following command starts another Cassandra container instance and runs `cqlsh` (Cassandra Query Language Shell) against your original Cassandra container, allowing you to execute CQL statements against your database instance:

```
$ docker run -it --link some-cassandra:cassandra --rm cassandra sh
```

... or (simplified to take advantage of the `/etc/hosts` entry Docker adds for linked containers):

```
$ docker run -it --link some-cassandra:cassandra --rm cassandra cqls
```

... where `some-cassandra` is the name of your original Cassandra Server container.

More information about the CQL can be found in the Cassandra documentation (https://cassandra.apache.org/doc/latest/cql/index.html).

Container shell access and viewing Cassandra logs

The `docker exec` command allows you to run commands inside a Docker container. The following command line will give you a bash shell inside your `cassandra` container:

```
$ docker exec -it some-cassandra bash
```

The Cassandra Server log is available through Docker's container log:

```
$ docker logs some-cassandra
```

Configuring Cassandra

The best way to provide configuration to the `cassandra` image is to provide a custom `/etc/cassandra/cassandra.yaml` file. There are many ways to provide this file to the container (via short `Dockerfile` with `FROM + COPY`, via Docker Configs (https://docs.docker.com/engine/swarm/configs/), via runtime bind-mount, etc), the details of which are left as an exercise for the reader.

To use a different file name (for example, to avoid all image-provided configuration behavior), use `-Dcassandra.config=/path/to/cassandra.yaml` as an argument to the image (as in, `docker run ... cassandra -`

`Dcassandra.config=/path/to/cassandra.yaml`).

There are a small number of environment variables supported by the image which will modify `/etc/cassandra/cassandra.yaml` in some way (but the script is modifying YAML, so is naturally fragile):

- `CASSANDRA_LISTEN_ADDRESS` : This variable is for controlling which IP address to listen for incoming connections on. The default value is `auto` , which will set the `listen address` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__listen_address) option in `cassandra.yaml` to the IP address of the container as it starts. This default should work in most use cases.

- `CASSANDRA_BROADCAST_ADDRESS` : This variable is for controlling which IP address to advertise to other nodes. The default value is the value of `CASSANDRA_LISTEN_ADDRESS` . It will set the `broadcast address` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__broadcast_address) and `broadcast rpc address` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__broadcast_rpc_address) options in `cassandra.yaml` .

- `CASSANDRA_RPC_ADDRESS` : This variable is for controlling which address to bind the thrift rpc server to. If you do not specify an address, the wildcard address (`0.0.0.0`) will be used. It will set the `rpc address` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__rpc_address) option in `cassandra.yaml` .

- `CASSANDRA_START_RPC` : This variable is for controlling if the thrift rpc server is started. It will set the `start rpc` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__start_rpc) option in `cassandra.yaml` .

- `CASSANDRA_SEEDS` : This variable is the comma-separated list of IP addresses used by gossip for bootstrapping new nodes joining a cluster. It will set the `seeds` value of the `seed provider` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__seed_provider) option in `cassandra.yaml` . The `CASSANDRA_BROADCAST_ADDRESS` will be added the the seeds passed in so that the sever will talk to itself as well.

- `CASSANDRA_CLUSTER_NAME` : This variable sets the name of the cluster and must be the same for all nodes in the cluster. It will set the `cluster name` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?

scroll=configCassandra_yaml__cluster_name) option of `cassandra.yaml`.

- `CASSANDRA_NUM_TOKENS`: This variable sets number of tokens for this node. It will set the `num_tokens` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__num_tokens) option of `cassandra.yaml`.

- `CASSANDRA_DC`: This variable sets the datacenter name of this node. It will set the `dc` (http://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archsnitchGossipPF.html) option of `cassandra-rackdc.properties`. You must set `CASSANDRA_ENDPOINT_SNITCH` to use the "GossipingPropertyFileSnitch" (https://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archsnitchGossipPF.html) in order for Cassandra to apply `cassandra-rackdc.properties`, otherwise this variable will have no effect.

- `CASSANDRA_RACK`: This variable sets the rack name of this node. It will set the `rack` (http://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archsnitchGossipPF.html) option of `cassandra-rackdc.properties`. You must set `CASSANDRA_ENDPOINT_SNITCH` to use the "GossipingPropertyFileSnitch" (https://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archsnitchGossipPF.html) in order for Cassandra to apply `cassandra-rackdc.properties`, otherwise this variable will have no effect.

- `CASSANDRA_ENDPOINT_SNITCH`: This variable sets the snitch implementation this node will use. It will set the `endpoint snitch` (http://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/configCassandra_yaml.html?scroll=configCassandra_yaml__endpoint_snitch) option of `cassandra.yml`.

Caveats

Where to Store Data

Important note: There are several ways to store data used by applications that run in Docker containers. We encourage users of the `cassandra` images to familiarize themselves with the options available, including:

- Let Docker manage the storage of your database data by writing the database files to disk on the host system using its own internal volume management (https://docs.docker.com/engine/tutorials/dockervolumes/#adding-a-data-volume). This is the default and is easy and fairly transparent to the user. The downside is that the files may be hard to locate for tools and applications that run directly on the host system, i.e. outside containers.
- Create a data directory on the host system (outside the container) and mount this to a directory visible from inside the container

(https://docs.docker.com/engine/tutorials/dockervolumes/#mount-a-host-directory-as-a-data-volume). This places the database files in a known location on the host system, and makes it easy for tools and applications on the host system to access the files. The downside is that the user needs to make sure that the directory exists, and that e.g. directory permissions and other security mechanisms on the host system are set up correctly.

The Docker documentation is a good starting point for understanding the different storage options and variations, and there are multiple blogs and forum postings that discuss and give advice in this area. We will simply show the basic procedure here for the latter option above:

1. Create a data directory on a suitable volume on your host system, e.g. `/my/own/datadir`.
2. Start your `cassandra` container like this:

```
$ docker run --name some-cassandra -v /my/own/datadir:/var/lib/ca
```

The `-v /my/own/datadir:/var/lib/cassandra` part of the command mounts the `/my/own/datadir` directory from the underlying host system as `/var/lib/cassandra` inside the container, where Cassandra by default will write its data files.

No connections until Cassandra init completes

If there is no database initialized when the container starts, then a default database will be created. While this is the expected behavior, this means that it will not accept incoming connections until such initialization completes. This may cause issues when using automation tools, such as `docker-compose`, which start several containers simultaneously.

License

View license information (https://git-wip-us.apache.org/repos/asf?p=cassandra.git;a=blob;f=LICENSE.txt;hb=cassandra-3.11.1) for the software contained in this image.

As with all Docker images, these likely also contain other software which may be under other licenses (such as Bash, etc from the base distribution, along with any direct or indirect dependencies of the primary software being contained).

Some additional license information which was able to be auto-detected might be found in the `repo-info` repository's `cassandra/` directory (https://github.com/docker-library/repo-info/tree/master/repos/cassandra).

As for any pre-built image usage, it is the image user's responsibility to ensure that any use of this image complies with any relevant licenses for all software contained within.

Docker Pull Command

```
docker pull cassandra
```