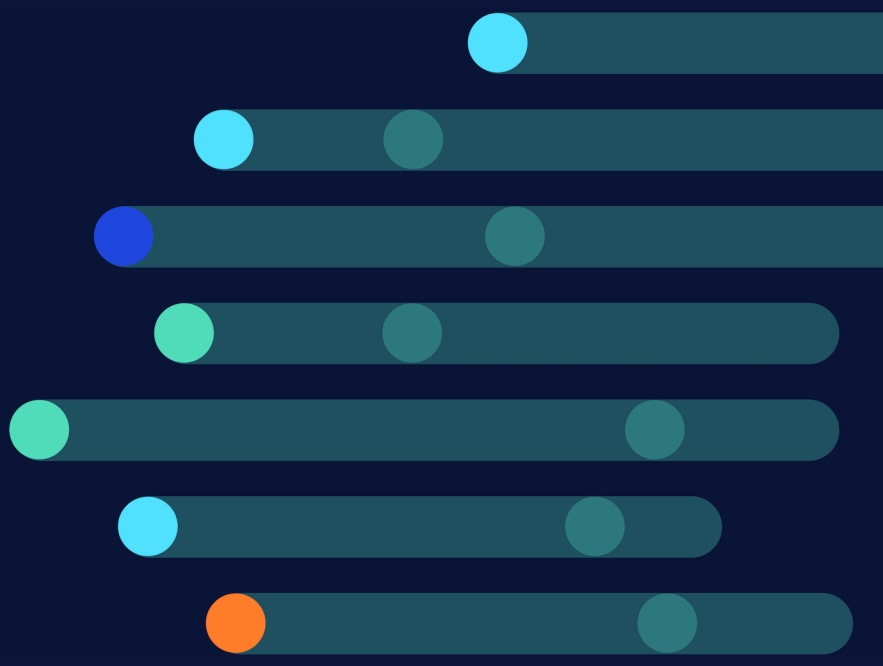
DataStax Docker Docs



DATASTAX Documentation

2021-04-17-07:00

Contents

1. DataStax Docker Quickstart

Use DataStax Docker images to create containers for production and non-production environments.

Prerequisites

1. Download and install Docker from the Docker website.
2. Download the DataStax Docker images from Docker Hub.

Quick start examples

Create a DDAC database container

```
$ docker run -e DS_LICENSE=accept --name my-ddac -d datastax/ddac
```

Create a DSE database container

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server
```

Create a DSE container with Analytics, Search, and Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -k
-s -g
```

Create a DSE OpsCenter container

```
$ docker run -e DS_LICENSE=accept -p 8888:8888 --name my-opscenter -d
datastax/dse-opscenter
```

Create a DataStax Studio container

```
$ docker run -e DS_LICENSE=accept --name my-studio -p 9091:9091 -d
datastax/dse-studio --link my-dse
```

Learn more

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

2. DataStax Enterprise recommended settings for Docker

To ensure your success when using Docker, follow the recommended guidance and settings for using DataStax Enterprise (DSE) with Docker.

Important: Although DataStax provides the following guidance, adaptations of these instructions might be required depending on the deployment. It is highly recommended to rigorously test the use cases under consideration before deploying a DataStax installation on Docker in production environments.

General guidance

DSE achieves resilience and high availability through a cluster of nodes that replicate data across the cluster. This replication ensures that if any individual node fails, access to data is not lost and performance is maintained. However, in a containerized environment, running multiple DSE nodes on the same physical hardware will introduce a single point of failure.

Important: To avoid a single point of failure, run only a single DataStax container on a DSE cluster per Docker host. If running multiple DataStax containers on a single Docker host, ensure that the containers are in different DSE clusters.

Software versions

DataStax Agent versions

The official DataStax images include the latest DataStax Agent version at the time of official image build. If you require a version of the DataStax Agent that differs from the one included with the official image, you must build an image that includes the required versions.

Hardware settings

Docker container resource requirements

For minimum container resource requirements, follow the capacity planning guidance for selecting hardware for production environments:

- DSE

Optimizing SSDs

The default SSD configurations on most Linux distributions are not optimal. To ensure the best settings, see the recommended production settings to optimize SSDs:

- DSE 6.7 | DSE 6.0 | DSE 5.1

- DDAC

Optimizing settings for RAID on SSD

The optimum `readahead` setting for RAID on SSDs (in Amazon EC2) is 8 KB, the same as it is for non-RAID SSDs. For details, see Optimizing SSDs.

Optimizing RAID settings for spinning disks on the host

Typically, a `readahead` of 128 is recommended.

Check to ensure `setra` is not set to 65536:

```
$ sudo blockdev --report /dev/spinning_disk
```

To set setra:

```
$ sudo blockdev --setra 128 /dev/spinning_disk
```

System settings

Synchronizing clocks

Because time is not namespaced in the Linux kernel, containers share the clock with the Docker host machine. Ensure that clocks are synchronized on the host machines and containers by configuring NTP or other methods on the host machines.

Disabling swap

Swapping must be disabled for performance and node stability. Run the following command on the Docker host to disable swap. The Docker host passes this setting to the container.

See *Disabling swap* for DSE 6.7 | DSE 6.0 | DSE 5.1 | DDAC.

```
$ sudo swapoff --all
```

- To disable swap per container, see Preventing a container from using SWAP in the Docker documentation.
- To make this change permanent, remove all swap file entries from `/etc/fstab`.

Disabling CPU frequency sequencing on the Docker host

To ensure optimal performance, do not use governors that lower the CPU frequency. Instead, reconfigure all CPUs to use the `performance` governor on the Docker hosts.

See *Disable CPU frequency scaling* for DSE 6.7 | DSE 6.0 | DSE 5.1 | DDAC.

```
for CPUFREQ in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
do
    [ -f $CPUFREQ ] || continue
    echo -n performance > $CPUFREQ
```

```
done
```

Disabling THP on the Docker host

THP can cause performance issues in DSE or DDAC when it defragments 4 K chunks into 2 MB chunks. To disable `defrag`, run the following command on the Docker host:

```
$ echo never | sudo tee /sys/kernel/mm/transparent_hugepage/defrag
```

See *Check Java Hugepages settings* for DSE 6.7 | DSE 6.0 | DSE 5.1 | DDAC.

Increasing user resource limits

All containers by default inherit user limits from the Docker daemon. In production environments, DSE expects the following changes to `ulimit`:

```
ulimit -n 100000 # nofile: max number of open files
```

```
ulimit -l unlimited # memlock: maximum locked-in-memory address space
```

1. Run the following command to check the Docker daemon defaults for `ulimits`:

   ```
   $ docker run --rm ubuntu /bin/bash -c 'ulimit -a'
   ```

2. To set `ulimit` for Docker containers, run the `docker run` command with the following `ulimit` options:

   ```
   $ --ulimit nofile=100000:100000 --ulimit nproc=32768 --ulimit
   memlock=-1:-1
   ```

DSE tries to lock memory using `mlock`. When running in Docker, that capability is disabled. To enable `mlock`, add the following option to the `docker run` command:

```
$ --cap-add=IPC_LOCK
```

On the Docker host, check the value of `vm.max_map_count`, which should be set to 1048575.

```
$ cat /proc/sys/vm/max_map_count
```

To set the value of `vm.max_map_count`, add the following line to `/etc/sysctl.conf`, and then run `sysctl -p` to propagate the changes.

```
$ vi /etc/sysctl.conf
```

```
$ vm.max_map_count = 1048575
```

```
$ sudo sysctl -p
```

See *Set user resource limits* for DSE 6.7 | DSE 6.0 | DSE 5.1 | DDAC.

Configuring heap settings

For each container in production environments, explicitly set the JVM heap size using the `JVM_EXTRA_OPTS` environment variable with the `docker run` command.

For example, to use 16 GB for the JVM heap, run the `docker run` command with the following option:

```
$ docker run -e JVM_EXTRA_OPTS="-Xms16g -Xmx16g"
```

Storage and resource requirements

Mounting configuration volumes

For advanced configuration management, DataStax provides a mechanism for modifying configurations without replacing or customizing DataStax Docker containers. When any of the approved configuration files are mounted to a host volume, the files are mapped automatically within the container. See Using the DSE configuration volume.

Mapping node data to a local folder on the host

The DSE Docker container writes all node-specific data in the directories under `/var/lib/cassandra/` by default. To persist this data, map the data directories inside the container to a directory on the host file system using the -v option with the `docker run` command, or by using a volume driver.

For example, to mount the DSE data volume to the `/dse/data` directory on the Docker host, run the `docker run` command with the following option:

```
$ docker run -v /dse/data:/var/lib/cassandra
```

Hosting the `/var/lib/cassandra` directory outside the container with the -v option allows the container to be deleted and recreated without losing data. See Persisting data.

Configuring storage drivers

If using the Docker `devicemapper` storage driver, do not use the default `loop-lvm` mode, which is only appropriate for testing. Instead, configure `docker-engine` to use direct-lvm mode, which is suitable for production environments.

Resources allocated to Linux VM in Docker for Windows

See the DataStax Developer Blog Running DSE on Microsoft Windows Using Docker. When running Docker for Windows, the default resources allocated to the Linux VM running docker are 2 GB RAM and 2 CPUs. Adjust these resources as appropriate to meet the requirements for your containers. See Getting Started in *Docker Desktop for Windows*.

Network considerations

Configuring network settings

Because the default network settings in Docker (via Linux bridge) slows networking considerably, do not use these network settings in production environments. Instead, use docker host networking by adding the `--network host` option to the `docker run` command, or use a plugin that can manage IP ranges across clusters of hosts. The host networking limits the number of nodes per Docker host to one, which is the recommended configuration to use in production.

```
$ docker run -d --network host --name container_name
```

Configuring ports

Communication occurs on many different ports. Account for required communication and security for these ports when binding ports to the Docker host:

- DSE 6.7 | DSE 6.0 | DSE 5.1

- DDAC

3. Docker Guide for DataStax 6.8

Use DataStax Docker images to create DataStax Enterprise (DSE) 6.8 server, DSE OpsCenter 6.8, and DataStax Studio 6.8 containers in production and non-production environments.

Getting started with DataStax and Docker

Use DataStax Docker images to create containers in production and non-production environments for development, learn DataStax Enterprise (DSE), DataStax OpsCenter, and DataStax Studio, try new ideas, and test and demonstrate an application. The following images are available:

- DDAC: DataStax Distribution of Apache Cassandra™ (DDAC) is a certified version of open source Apache Cassandra™ for development and production.
- DataStax Enterprise: The best distribution of Apache Cassandra™ with integrated Search, Analytics, and Graph, and Advanced Security capabilities.
- DataStax Studio: An interactive developer's tool for DataStax Enterprise which is designed to help your DSE database, Cassandra Query Language (CQL), DSE Graph, and Gremlin Query Language development.
- DSE OpsCenter: The web-based visual management and monitoring solution for DSE.

To get started, clone the repository and change for your environment.

Prerequisites

To use the Docker images, ensure the following requirements are met:

- Docker CE/EE 17.03 or later. Supported platforms:
  - Linux
  - Windows (See Running DSE on Microsoft Windows Using Docker.)
  - Mac
- Basic understanding of Docker images and containers.
- Docker installed on your local system.

Creating and starting Docker containers

Use the following information to create DataStax Enterprise (DSE) server, DSE OpsCenter, and DataStax Studio containers in production and non-production environments.

Creating a DataStax Enterprise container

Create a DataStax Enterprise (DSE) server container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Enabling advanced functionality

By default, the DSE server image is configured as a transactional (database) node. To configure the node with DSE advanced functionality, add the corresponding option that enables the intended feature to the end of the `docker run` command.

Combine startup options to run more than one feature.

Option	Description
`-g`	Enable and start DSE Graph.
`-k`	Enable and start DSE Analytics.
`-s`	Enable and start DSE Search.

Examples

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Create a DSE database container

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server
```

Create a DSE container with Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -g
```

Create a DSE container with Analytics (Spark) enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -k
```

Create a DSE container with Search enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
```

Create a DSE container with Search, Analytics, and Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
-k -g
```

Creating an OpsCenter container

Create a DSE OpsCenter container and a connected DSE server container on the same Docker host. For a list of the most commonly used options, see Docker run options.

Note: DSE 6.8 requires OpsCenter 6.8.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create an OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ -p 8888:8888 \ --name my-
opscenter \ -d datastax/dse-opscenter
```

2. Create a DSE server container that is linked to the OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ --link my-opscenter:opscenter \
--name my-dse \ -d datastax/dse-server
```

3. Get the DSE container IP address.

```
$ docker exec -it my-dse nodetool status
```

4. Open a browser and navigate to http://*dse_container_ip*:8888, where *dse_container_ip* is the IP address of the OpsCenter container.
    a. Click **Manage existing cluster**.
    b. Enter the DSE container IP address in the **host name** field.
    c. Click **Install agents manually**. The agent is already installed on the DSE image, so no installation is required.

Results
OpsCenter is ready to use with DSE.

What's next:

What's next
See the DSE OpsCenter User Guide for detailed usage and configuration instructions.

Creating a Studio container

Create a DataStax Studio container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create a Studio container, using `my-dse` as the hostname.

```
$ docker run -e DS_LICENSE=accept \ -p 9091:9091 \ --link my-dse \ --
name my-studio \ -d datastax/dse-studio
```

2. Open a browser and navigate to http://*studio_container_ip*:9091, where *container_IP* is the IP address of the container.

Results
Studio is ready to use with DSE.

What's next:

What's next
See the DataStax Studio documentation for detailed usage and configuration instructions.

Docker run options

The following options are the most commonly used when creating a DataStax container.

Option	Description
`-e`	Sets environment variables to accept the licensing agreement and change the initial configuration. **Required**. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`-d`	Starts the container in the background. Recommended.
`-p`	Publish container ports on the host to allow remote access to DSE, OpsCenter, and Studio. See exposing public ports for more information.
`-v`	Bind mount a directory on the local host to a DSE Volume to manage configuration or preserve data. See using exposed volumes for more information.
`--link`	Link a DSE container to OpsCenter, or Studio to DSE. For example, `--link my-opscenter:opscenter` or `--link my-dse`.
`--name`	Assign a name to the container.

Managing the configuration

Manage the DataStax Enterprise (DSE) configuration using one of the following options:

- The DSE configuration volume to get configuration files from a mounted host directory without replacing or customizing configuration file in the container.
- Environment variables to change the configuration at runtime.
- Docker file or directory volume mounts.
- Docker overlay file system.

Note: DSE and DDAC use the default values defined for the environment variables unless explicitly set at runtime. Custom configuration files override the default or explicitly set environment variables.

DataStax uses a common base image for all products. To customize the operating system or install additional packages, modify the `base/Dockerfile`. The DataStax base images use OpenJDK due to the end of public updates for Oracle JDK. All DataStax repositories on Docker Hub include OpenJDK.

Using the DSE configuration volume

Docker images provided by DataStax include a startup script that swaps DataStax Enterprise (DSE) configuration files found in the `/config` volume directory with the configuration file in the default location on the container.

Procedure

1. Create a directory on your local host to store the configuration files.
2. Add the configuration files to replace in the container. The file name must match a corresponding configuration file in the image and include all required values. For example `cassandra.yaml`, `dse.yaml`, `opscenterd.conf`.

   See the GitHub pages for a full list of configuration files.

   - DSE
   - OpsCenter
   - Studio
3. Mount the local directory to the exposed `/config` directory during startup.

   For example:

   ```
   $ docker run -v /dse/conf:/config
   ```
4. Start the container.

For example, to start a transactional node:

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/config:/
config datastax/dse-server \ -d datastax/dse-server
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

5. After you make changes or add config files to the `/config` volume, restart the container to propagate the changes to the database.

```
$ docker restart container_name
```

Restarting the container restarts DSE and DDAC.

Using environment variables

Configure the DataStax Enterprise (DSE) Docker image by setting environment variables when the container is created. Use the docker run command `-e` option.

Table 1. Environment variables

Variable	Setting	Description
`DS_LICENSE`	accept	To show and acknowledge the license, set the variable `DS_LICENSE` to the value `accept`. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`LISTEN_ADDRESS`	IP_address	IP address to listen for connections from other nodes. Defaults to the container IP address.
`BROADCAST_ADDRESS`	IP_address	IP address to advertise to other nodes. Defaults to the same value as `LISTEN_ADDRESS`.
`NATIVE_TRANSPORT_ADDRESS`	IP_address	IP address to list for client and driver connections. Default: `0.0.0.0`.
`NATIVE_TRANSPORT_BROADCAST_ADDRESS`	IP_address	IP address to advertise to clients and drivers. Defaults to the same value as `BROADCAST_ADDRESS`.

Table 1. Environment variables (continued)

Variable	Setting	Description
`SEEDS`	IP_address	Comma-delimited list of seed nodes for the cluster. Defaults to the node `BROADCAST_ADDRESS`.
`START_NATIVE_TRANSPORT`	true | false	Determines whether to start the Thrift RPC server. If not set, the default value is preserved in the `cassandra.yaml` file.
`CLUSTER_NAME`	string	Name of the cluster. Default: `Test Cluster`.
`NUM_TOKENS`	int	Number of tokens randomly assigned to the node. Default: `8`.
`DC`	string	Datacenter name. Default: `Cassandra`.
`RACK`	string	Rack name. Default: `rack1`.
`OPSCENTER_IP`	IP_address | string	Address of the OpsCenter instance to use for DSE management. The value can be specified by linking the OpsCenter container using `opscenter` as the name.
`JVM_EXTRA_OPTS`	string	Sets a custom value for the JVM heap using `-Xmx` and `-Xms`.
`LANG`	string	Sets a custom locale.
`SNITCH`	string	Sets the snitch implementation this node will use. The value is set in the `endpoint_snitch` parameter in `cassandra.yaml`.

Persisting data

Persisting data allows the container to be deleted and recreated without losing data. To persist data, create directories on the local host and map the directory to the corresponding volume using the docker run command with the `-v` option. For example:

```
$ docker run -v local_directory:container_volume
```

Warning: If the volumes are not mounted from the local host, all data is lost when the container is removed.

DataStax exposes data volumes to preserve data. See volumes and data directories for a list of exposed volumes.

Procedure

1. Create a directory on the Docker host.
2. Bind mount the local directory to the configuration file that will be persisted by starting the container with the `-v` option.

   Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Example

Mounting a DSE data volume

Mount the DSE data volume to the `/dse/data` directory on the Docker host to ensure that the `/data`, `/commit_logs`, and `/saved_caches` directories will be available. Hosting the `/var/lib/cassandra` directory outside the container with the `-v` option allows the Docker container to be deleted and recreated without losing data.

```
$ docker run -e DS_LICENSE=accept --name my-dse -v /dse/data:/var/lib/
cassandra
```

Mounting a DSE configuration volume

Mount the host directory `/dse/config` to the DSE volume `/config` to manage configuration files.

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/conf:/config
datastax/dse-server \ -d datastax/dse-server
```

Mounting an OpsCenter configuration volume

Mount the local directory to the exposed volume `/var/lib/opscenter` by starting the container with the `-v` option.

```
$ docker run -e DS_LICENSE=accept \ -v /dse/data/opscenter:/var/lib/
opscenter \ --name my-opscenter \ -d datastax/dse-opscenter
```

Exposing public ports

To allow remote hosts to access a DataStax Enterprise (DSE) or DataStax Distribution of Apache Cassandra™ (DDAC) node, DSE OpsCenter, or DataStax Studio, map the DSE public port to a host port using the docker run command with the `-p` option.

For a complete list of ports see Securing DataStax Enterprise ports.

Note: When mapping a container port to a local host port, ensure the host port is not already in use by another container or the host.

Example

To allow access to DDAC from a browser on a remote host, open port `8888` as shown in the following example.

```
$ docker run -e DS_LICENSE=accept \ --name my-opscenter \ -p 8888:8888 \ -d
datastax/dse-opscenter
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Volumes and data directories

DataStax exposes the following volumes so that underlying directories can be mounted. If these volumes are not mounted, then sub-directories will be unavailable. For example, by mounting the `/var/lib/cassandra` directory, the `/data`, `/commit_logs`, `/saved_caches` directories will be available.

Tip: To persist data, create the directories on the local host, and then map the local directory to the corresponding volume using the `docker run -v` flag:

```
$ docker run -v local_directory:container_volume
```

See using volumes in the Docker documentation.

DataStax Enterprise (DSE)

Directory	Description
`/var/lib/cassandra`	Database data
`/var/lib/spark`	DSE Analytics with Spark data
`/var/lib/dsefs`	DSEFS data
`/var/log/cassandra`	Database logs
`/var/log/spark`	Spark logs
`/config`	Custom configuration files

Studio

Directory	Description
`/var/lib/datastax-studio`	DataStax Studio data

Attaching to a container

Use the `docker exec -it container_name` command to attach to a container and run DataStax Enterprise (DSE) tools and other operations.

Opening an interactive bash shell

If the container is running in the background (using the `-d` option), use the following command to open an interactive bash shell:

```
$ docker exec -it container_name bash
```

To exit the shell without stopping the container, type `exit`.

Opening an interactive CQL shell (cqlsh)

Use the following command to open the cqlsh prompt.

```
$ docker exec -it container_name cqlsh
```

To exit the shell without stopping the container, use `Ctrl + P + Q`.

Viewing logs

View DSE logs using the docker log command.

```
$ docker logs my-dse
```

Using DSE tools

Use the `docker exec` command to run other tools. For example:

```
$ docker exec -it my-dse nodetool status
```

See the DSE documentation for further information.

Using Docker compose for automated provisioning

Use Docker Compose to automate bootstrapping a multi-node cluster with DataStax Enterprise (DSE), DSE OpsCenter, and DataStax Studio. Use the following links to get sample `compose.yml` files for different tools and services:

- DSE
- DSE OpsCenter
- DataStax Studio

Three node configuration

When creating multiple nodes, use the `node` parameter to bootstrap one node at a time. For example, the first node is `node=0`, the second node is `node=1`, and the third node is `node=2`.

Wait for each node to finish bootstrapping before running `docker-compose` for the next node.

```
$ docker-compose -f docker-compose.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ up
-d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter and Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ -f
docker-compose.studio.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Single node configuration with Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.studio.yml \ up -d
--scale node=0
```

Single node configuration

To bootstrap a single-node cluster, use the `docker run` command and specify the version of DSE to install, plus any additional options.

```
$ docker run -e DS_LICENSE=accept -d datastax/dse-server:version \ options
```

Building custom Docker images

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

DataStax provides a code repository on Github for building custom Docker images.

By default, Gradle downloads the DataStax tarballs from the DataStax Downloads page.

Important: End User License Agreement (EULA). By downloading this DataStax product, you agree to the terms of the EULA.

To build all images, run the following command, entering DataStax Academy credentials for *username* and *password*.

```
$ ./gradlew buildImages
```

To build a product image for a specific version, invoke a Grade task that follows this pattern:

```
$ ./gradlew build<product><version>Image
```

For example:

```
$ ./gradlew buildServer6.0.6Image
```

To build more than one image with specific versions:

```
$ ./gradlew buildServer6.0.6Image buildOpscenter6.5.0Image
```

To get the list of all available tasks, run:

```
$ ./gradlew tasks
```

Multiple product versions

To support multiple product versions without duplicating files, Docker build contexts are generated from source folders that contain FreeMarker templates (files with `.ftl` extensions). The following conventions are used:

- Docker build contexts are generated from self-describing product folders. For example, `server`, `opscenter`, and `studio`.
- All files without the `.ftl` extension are copied to the build context.
- Files with `.ftl` extensions are processed as FreeMarker templates:
  - Template directives are written using angle bracket syntax.
  - Square bracket syntax is used for interpolations.
  - The processed files are copied to the build context without `.ftl` extension. For example, `Dockerfile.ftl` is copied as `Dockerfile`.
- FreeMarker templates use the `version` variable:
  - `version.major` returns product version major number
  - `version.minor` returns product version minor number
  - `version.bugfix` returns product version bugfix number
  - The following version functions are available:

  - `version.lowerThan('x.y.z')` returns `true` if `version` is semantically lower than `x.y.z`
  - `version.greaterEqualThan('x.y.z')` returns `true` if `version` is semantically greater than or equal to `x.y.z`.

To customize the products or to use multiple product versions, modify the templates in their corresponding product folder.

Getting help with Docker

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

Docker known issues

The following issues are recognized.

- Cassandra File System (CFS) is not supported.
- Lifecycle Manager (LCM) is not supported.
- Changing any file not included in the list of approved configuration files will require an additional host volume or customization of the image. An example is SSL key management.
- The JVM heap size must be set for DataStax Enterprise (DSE) running inside the container using the `JVM_EXTRA_OPTS` variable or custom `cassandra-env.sh`. If not set, Java does not honor resource limits set for the container, and will peer through the container to use resources (memory and CPU) of the host. See the `JVM_EXTRA_OPTS` variable in Using environment variables for more information.

Licensing

Review the licensing terms for each of the following products and services:

- DataStax License Terms

- DSE OpsCenter License Terms
- DataStax Studio License Terms
- DDAC License Terms

4. Docker Guide for DataStax 6.7

Use DataStax Docker images to create DataStax Enterprise (DSE) 6.7 server, DSE OpsCenter 6.7, and DataStax Studio 6.7 containers in production and non-production environments.

Getting started with DataStax and Docker

Use DataStax Docker images to create containers in production and non-production environments for development, learn DataStax Enterprise (DSE), DataStax OpsCenter, and DataStax Studio, try new ideas, and test and demonstrate an application. The following images are available:

- DDAC: DataStax Distribution of Apache Cassandra™ (DDAC) is a certified version of open source Apache Cassandra™ for development and production.
- DataStax Enterprise: The best distribution of Apache Cassandra™ with integrated Search, Analytics, and Graph, and Advanced Security capabilities.
- DataStax Studio: An interactive developer's tool for DataStax Enterprise which is designed to help your DSE database, Cassandra Query Language (CQL), DSE Graph, and Gremlin Query Language development.
- DSE OpsCenter: The web-based visual management and monitoring solution for DSE.

To get started, clone the repository and change for your environment.

Prerequisites

To use the Docker images, ensure the following requirements are met:

- Docker CE/EE 17.03 or later. Supported platforms:
  - Linux
  - Windows (See Running DSE on Microsoft Windows Using Docker.)
  - Mac
- Basic understanding of Docker images and containers.
- Docker installed on your local system.

Creating and starting Docker containers

Use the following information to create DataStax Enterprise (DSE) server, DSE OpsCenter, and DataStax Studio containers in production and non-production environments.

Creating a DataStax Enterprise container

Create a DataStax Enterprise (DSE) server container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Enabling advanced functionality

By default, the DSE server image is configured as a transactional (database) node. To configure the node with DSE advanced functionality, add the corresponding option that enables the intended feature to the end of the `docker run` command.

Combine startup options to run more than one feature.

Option	Description
`-g`	Enable and start DSE Graph.
`-k`	Enable and start DSE Analytics.
`-s`	Enable and start DSE Search.

Examples

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Create a DSE database container

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server
```

Create a DSE container with Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -g
```

Create a DSE container with Analytics (Spark) enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -k
```

Create a DSE container with Search enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
```

Create a DSE container with Search, Analytics, and Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
-k -g
```

Creating an OpsCenter container

Create a DSE OpsCenter container and a connected DSE server container on the same Docker host. For a list of the most commonly used options, see Docker run options.

Note: DSE 6.7 requires OpsCenter 6.7.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create an OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ -p 8888:8888 \ --name my-
opscenter \ -d datastax/dse-opscenter
```

2. Create a DSE server container that is linked to the OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ --link my-opscenter:opscenter \
--name my-dse \ -d datastax/dse-server
```

3. Get the DSE container IP address.

```
$ docker exec -it my-dse nodetool status
```

4. Open a browser and navigate to http://*dse_container_ip*:8888, where *dse_container_ip* is the IP address of the OpsCenter container.
    a. Click **Manage existing cluster**.
    b. Enter the DSE container IP address in the **host name** field.
    c. Click **Install agents manually**. The agent is already installed on the DSE image, so no installation is required.

Results
OpsCenter is ready to use with DSE.

What's next:

What's next
See the DSE OpsCenter User Guide for detailed usage and configuration instructions.

Creating a Studio container

Create a DataStax Studio container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create a Studio container, using `my-dse` as the hostname.

```
$ docker run -e DS_LICENSE=accept \ -p 9091:9091 \ --link my-dse \ --
name my-studio \ -d datastax/dse-studio
```

2. Open a browser and navigate to http://*studio_container_ip*:9091, where *container_IP* is the IP address of the container.

Results
Studio is ready to use with DSE.

What's next:

What's next
See the DataStax Studio documentation for detailed usage and configuration instructions.

Docker run options

The following options are the most commonly used when creating a DataStax container.

Option	Description
`-e`	Sets environment variables to accept the licensing agreement and change the initial configuration. **Required**. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`-d`	Starts the container in the background. Recommended.
`-p`	Publish container ports on the host to allow remote access to DSE, OpsCenter, and Studio. See exposing public ports for more information.
`-v`	Bind mount a directory on the local host to a DSE Volume to manage configuration or preserve data. See using exposed volumes for more information.
`--link`	Link a DSE container to OpsCenter, or Studio to DSE. For example, `--link my-opscenter:opscenter` or `--link my-dse`.
`--name`	Assign a name to the container.

Managing the configuration

Manage the DataStax Enterprise (DSE) configuration using one of the following options:

- The DSE configuration volume to get configuration files from a mounted host directory without replacing or customizing configuration file in the container.
- Environment variables to change the configuration at runtime.
- Docker file or directory volume mounts.
- Docker overlay file system.

Note: DSE and DDAC use the default values defined for the environment variables unless explicitly set at runtime. Custom configuration files override the default or explicitly set environment variables.

DataStax uses a common base image for all products. To customize the operating system or install additional packages, modify the `base/Dockerfile`. The DataStax base images use OpenJDK due to the end of public updates for Oracle JDK. All DataStax repositories on Docker Hub include OpenJDK.

Using the DSE configuration volume

Docker images provided by DataStax include a startup script that swaps DataStax Enterprise (DSE) configuration files found in the `/config` volume directory with the configuration file in the default location on the container.

Procedure

1. Create a directory on your local host to store the configuration files.
2. Add the configuration files to replace in the container. The file name must match a corresponding configuration file in the image and include all required values. For example `cassandra.yaml`, `dse.yaml`, `opscenterd.conf`.

   See the GitHub pages for a full list of configuration files.

   - DSE
   - OpsCenter
   - Studio
3. Mount the local directory to the exposed `/config` directory during startup.

   For example:

   ```
   $ docker run -v /dse/conf:/config
   ```
4. Start the container.

For example, to start a transactional node:

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/config:/
config datastax/dse-server \ -d datastax/dse-server
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

5. After you make changes or add config files to the `/config` volume, restart the container to propagate the changes to the database.

```
$ docker restart container_name
```

Restarting the container restarts DSE and DDAC.

Using environment variables

Configure the DataStax Enterprise (DSE) Docker image by setting environment variables when the container is created. Use the docker run command `-e` option.

Table 2. Environment variables

Variable	Setting	Description
`DS_LICENSE`	accept	To show and acknowledge the license, set the variable `DS_LICENSE` to the value `accept`. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`LISTEN_ADDRESS`	IP_address	IP address to listen for connections from other nodes. Defaults to the container IP address.
`BROADCAST_ADDRESS`	IP_address	IP address to advertise to other nodes. Defaults to the same value as `LISTEN_ADDRESS`.
`NATIVE_TRANSPORT_ADDRESS`	IP_address	IP address to list for client and driver connections. Default: `0.0.0.0`.
`NATIVE_TRANSPORT_BROADCAST_ADDRESS`	IP_address	IP address to advertise to clients and drivers. Defaults to the same value as `BROADCAST_ADDRESS`.

Table 2. Environment variables (continued)

Variable	Setting	Description
`SEEDS`	IP_address	Comma-delimited list of seed nodes for the cluster. Defaults to the node `BROADCAST_ADDRESS`.
`START_NATIVE_TRANSPORT`	true | false	Determines whether to start the Thrift RPC server. If not set, the default value is preserved in the `cassandra.yaml` file.
`CLUSTER_NAME`	string	Name of the cluster. Default: `Test Cluster`.
`NUM_TOKENS`	int	Number of tokens randomly assigned to the node. Default: `8`.
`DC`	string	Datacenter name. Default: `Cassandra`.
`RACK`	string	Rack name. Default: `rack1`.
`OPSCENTER_IP`	IP_address | string	Address of the OpsCenter instance to use for DSE management. The value can be specified by linking the OpsCenter container using `opscenter` as the name.
`JVM_EXTRA_OPTS`	string	Sets a custom value for the JVM heap using `-Xmx` and `-Xms`.
`LANG`	string	Sets a custom locale.
`SNITCH`	string	Sets the snitch implementation this node will use. The value is set in the `endpoint_snitch` parameter in `cassandra.yaml`.

Persisting data

Persisting data allows the container to be deleted and recreated without losing data. To persist data, create directories on the local host and map the directory to the corresponding volume using the docker run command with the `-v` option. For example:

```
$ docker run -v local_directory:container_volume
```

Warning: If the volumes are not mounted from the local host, all data is lost when the container is removed.

DataStax exposes data volumes to preserve data. See volumes and data directories for a list of exposed volumes.

Procedure

1. Create a directory on the Docker host.
2. Bind mount the local directory to the configuration file that will be persisted by starting the container with the `-v` option.

   Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Example

Mounting a DSE data volume

Mount the DSE data volume to the `/dse/data` directory on the Docker host to ensure that the `/data`, `/commit_logs`, and `/saved_caches` directories will be available. Hosting the `/var/lib/cassandra` directory outside the container with the `-v` option allows the Docker container to be deleted and recreated without losing data.

```
$ docker run -e DS_LICENSE=accept --name my-dse -v /dse/data:/var/lib/
cassandra
```

Mounting a DSE configuration volume

Mount the host directory `/dse/config` to the DSE volume `/config` to manage configuration files.

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/conf:/config
datastax/dse-server \ -d datastax/dse-server
```

Mounting an OpsCenter configuration volume

Mount the local directory to the exposed volume `/var/lib/opscenter` by starting the container with the `-v` option.

```
$ docker run -e DS_LICENSE=accept \ -v /dse/data/opscenter:/var/lib/
opscenter \ --name my-opscenter \ -d datastax/dse-opscenter
```

Exposing public ports

To allow remote hosts to access a DataStax Enterprise (DSE) or DataStax Distribution of Apache Cassandra™ (DDAC) node, DSE OpsCenter, or DataStax Studio, map the DSE public port to a host port using the docker run command with the `-p` option.

For a complete list of ports see Securing DataStax Enterprise ports.

Note: When mapping a container port to a local host port, ensure the host port is not already in use by another container or the host.

Example

To allow access to DDAC from a browser on a remote host, open port `8888` as shown in the following example.

```
$ docker run -e DS_LICENSE=accept \ --name my-opscenter \ -p 8888:8888 \ -d
datastax/dse-opscenter
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Volumes and data directories

DataStax exposes the following volumes so that underlying directories can be mounted. If these volumes are not mounted, then sub-directories will be unavailable. For example, by mounting the `/var/lib/cassandra` directory, the `/data`, `/commit_logs`, `/saved_caches` directories will be available.

Tip: To persist data, create the directories on the local host, and then map the local directory to the corresponding volume using the `docker run -v` flag:

```
$ docker run -v local_directory:container_volume
```

See using volumes in the Docker documentation.

DataStax Enterprise (DSE)

Directory	Description
`/var/lib/cassandra`	Database data
`/var/lib/spark`	DSE Analytics with Spark data
`/var/lib/dsefs`	DSEFS data
`/var/log/cassandra`	Database logs
`/var/log/spark`	Spark logs
`/config`	Custom configuration files

Studio

Directory	Description
`/var/lib/datastax-studio`	DataStax Studio data

Attaching to a container

Use the `docker exec -it container_name` command to attach to a container and run DataStax Enterprise (DSE) tools and other operations.

Opening an interactive bash shell

If the container is running in the background (using the `-d` option), use the following command to open an interactive bash shell:

```
$ docker exec -it container_name bash
```

To exit the shell without stopping the container, type `exit`.

Opening an interactive CQL shell (cqlsh)

Use the following command to open the cqlsh prompt.

```
$ docker exec -it container_name cqlsh
```

To exit the shell without stopping the container, use `Ctrl + P + Q`.

Viewing logs

View DSE logs using the docker log command.

```
$ docker logs my-dse
```

Using DSE tools

Use the `docker exec` command to run other tools. For example:

```
$ docker exec -it my-dse nodetool status
```

See the DSE documentation for further information.

Using Docker compose for automated provisioning

Use Docker Compose to automate bootstrapping a multi-node cluster with DataStax Enterprise (DSE), DSE OpsCenter, and DataStax Studio. Use the following links to get sample `compose.yml` files for different tools and services:

- DSE
- DSE OpsCenter
- DataStax Studio

Three node configuration

When creating multiple nodes, use the `node` parameter to bootstrap one node at a time. For example, the first node is `node=0`, the second node is `node=1`, and the third node is `node=2`.

Wait for each node to finish bootstrapping before running `docker-compose` for the next node.

```
$ docker-compose -f docker-compose.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ up
-d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter and Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ -f
docker-compose.studio.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Single node configuration with Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.studio.yml \ up -d
--scale node=0
```

Single node configuration

To bootstrap a single-node cluster, use the `docker run` command and specify the version of DSE to install, plus any additional options.

```
$ docker run -e DS_LICENSE=accept -d datastax/dse-server:version \ options
```

Building custom Docker images

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

DataStax provides a code repository on Github for building custom Docker images.

By default, Gradle downloads the DataStax tarballs from the DataStax Downloads page.

Important: End User License Agreement (EULA). By downloading this DataStax product, you agree to the terms of the EULA.

To build all images, run the following command, entering DataStax Academy credentials for *username* and *password*.

```
$ ./gradlew buildImages
```

To build a product image for a specific version, invoke a Grade task that follows this pattern:

```
$ ./gradlew build<product><version>Image
```

For example:

```
$ ./gradlew buildServer6.0.6Image
```

To build more than one image with specific versions:

```
$ ./gradlew buildServer6.0.6Image buildOpscenter6.5.0Image
```

To get the list of all available tasks, run:

```
$ ./gradlew tasks
```

Multiple product versions

To support multiple product versions without duplicating files, Docker build contexts are generated from source folders that contain FreeMarker templates (files with `.ftl` extensions). The following conventions are used:

- Docker build contexts are generated from self-describing product folders. For example, `server`, `opscenter`, and `studio`.
- All files without the `.ftl` extension are copied to the build context.
- Files with `.ftl` extensions are processed as FreeMarker templates:
  - Template directives are written using angle bracket syntax.
  - Square bracket syntax is used for interpolations.
  - The processed files are copied to the build context without `.ftl` extension. For example, `Dockerfile.ftl` is copied as `Dockerfile`.
- FreeMarker templates use the `version` variable:
  - `version.major` returns product version major number
  - `version.minor` returns product version minor number
  - `version.bugfix` returns product version bugfix number
  - The following version functions are available:

  - `version.lowerThan('x.y.z')` returns `true` if `version` is semantically lower than `x.y.z`
  - `version.greaterEqualThan('x.y.z')` returns `true` if `version` is semantically greater than or equal to `x.y.z`.

To customize the products or to use multiple product versions, modify the templates in their corresponding product folder.

Getting help with Docker

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

Docker known issues

The following issues are recognized.

- Cassandra File System (CFS) is not supported.
- Lifecycle Manager (LCM) is not supported.
- Changing any file not included in the list of approved configuration files will require an additional host volume or customization of the image. An example is SSL key management.
- The JVM heap size must be set for DataStax Enterprise (DSE) running inside the container using the `JVM_EXTRA_OPTS` variable or custom `cassandra-env.sh`. If not set, Java does not honor resource limits set for the container, and will peer through the container to use resources (memory and CPU) of the host. See the `JVM_EXTRA_OPTS` variable in Using environment variables for more information.

Licensing

Review the licensing terms for each of the following products and services:

- DataStax License Terms

- DSE OpsCenter License Terms
- DataStax Studio License Terms
- DDAC License Terms

5. Docker Guide for DataStax 6.0

Use DataStax Docker images to create DataStax Enterprise (DSE) 6.0 server, DSE OpsCenter 6.5, and DataStax Studio 6.0 containers in production and non-production environments.

Getting started with DataStax and Docker

Use DataStax Docker images to create containers in production and non-production environments for development, learn DataStax Enterprise (DSE), DataStax OpsCenter, and DataStax Studio, try new ideas, and test and demonstrate an application. The following images are available:

- DDAC: DataStax Distribution of Apache Cassandra™ (DDAC) is a certified version of open source Apache Cassandra™ for development and production.
- DataStax Enterprise: The best distribution of Apache Cassandra™ with integrated Search, Analytics, and Graph, and Advanced Security capabilities.
- DataStax Studio: An interactive developer's tool for DataStax Enterprise which is designed to help your DSE database, Cassandra Query Language (CQL), DSE Graph, and Gremlin Query Language development.
- DSE OpsCenter: The web-based visual management and monitoring solution for DSE.

To get started, clone the repository and change for your environment.

Prerequisites

To use the Docker images, ensure the following requirements are met:

- Docker CE/EE 17.03 or later. Supported platforms:
  - Linux
  - Windows (See Running DSE on Microsoft Windows Using Docker.)
  - Mac
- Basic understanding of Docker images and containers.
- Docker installed on your local system.

Creating and starting Docker containers

Use the following information to create DataStax Enterprise (DSE) server, DSE OpsCenter, and DataStax Studio containers in production and non-production environments.

Creating a DataStax Enterprise container

Create a DataStax Enterprise (DSE) server container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Enabling advanced functionality

By default, the DSE server image is configured as a transactional (database) node. To configure the node with DSE advanced functionality, add the corresponding option that enables the intended feature to the end of the `docker run` command.

Combine startup options to run more than one feature.

Option	Description
`-g`	Enable and start DSE Graph.
`-k`	Enable and start DSE Analytics.
`-s`	Enable and start DSE Search.

Examples

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Create a DSE database container

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server
```

Create a DSE container with Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -g
```

Create a DSE container with Analytics (Spark) enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -k
```

Create a DSE container with Search enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
```

Create a DSE container with Search, Analytics, and Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
-k -g
```

Creating an OpsCenter container

Create a DSE OpsCenter container and a connected DSE server container on the same Docker host. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create an OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ -p 8888:8888 \ --name my-
opscenter \ -d datastax/dse-opscenter
```

2. Create a DSE server container that is linked to the OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ --link my-opscenter:opscenter \
--name my-dse \ -d datastax/dse-server
```

3. Get the DSE container IP address.

```
$ docker exec -it my-dse nodetool status
```

4. Open a browser and navigate to http://*dse_container_ip*:8888, where *dse_container_ip* is the IP address of the OpsCenter container.
    a. Click **Manage existing cluster**.
    b. Enter the DSE container IP address in the **host name** field.
    c. Click **Install agents manually**. The agent is already installed on the DSE image, so no installation is required.

Results
OpsCenter is ready to use with DSE.

What's next:

What's next
See the DSE OpsCenter User Guide for detailed usage and configuration instructions.

Creating a Studio container

Create a DataStax Studio container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create a Studio container, using `my-dse` as the hostname.

```
$ docker run -e DS_LICENSE=accept \ -p 9091:9091 \ --link my-dse \ --
name my-studio \ -d datastax/dse-studio
```

2. Open a browser and navigate to http://*studio_container_ip*:9091, where *container_IP* is the IP address of the container.

Results
Studio is ready to use with DSE.

What's next:

What's next
See the DataStax Studio documentation for detailed usage and configuration instructions.

Docker run options

The following options are the most commonly used when creating a DataStax container.

Option	Description
`-e`	Sets environment variables to accept the licensing agreement and change the initial configuration. **Required**. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`-d`	Starts the container in the background. Recommended.
`-p`	Publish container ports on the host to allow remote access to DSE, OpsCenter, and Studio. See exposing public ports for more information.
`-v`	Bind mount a directory on the local host to a DSE Volume to manage configuration or preserve data. See using exposed volumes for more information.
`--link`	Link a DSE container to OpsCenter, or Studio to DSE. For example, `--link my-opscenter:opscenter` or `--link my-dse`.
`--name`	Assign a name to the container.

Managing the configuration

Manage the DataStax Enterprise (DSE) configuration using one of the following options:

- The DSE configuration volume to get configuration files from a mounted host directory without replacing or customizing configuration file in the container.
- Environment variables to change the configuration at runtime.
- Docker file or directory volume mounts.
- Docker overlay file system.

Note: DSE and DDAC use the default values defined for the environment variables unless explicitly set at runtime. Custom configuration files override the default or explicitly set environment variables.

DataStax uses a common base image for all products. To customize the operating system or install additional packages, modify the `base/Dockerfile`. The DataStax base images use OpenJDK due to the end of public updates for Oracle JDK. All DataStax repositories on Docker Hub include OpenJDK.

Using the DSE configuration volume

Docker images provided by DataStax include a startup script that swaps DataStax Enterprise (DSE) configuration files found in the `/config` volume directory with the configuration file in the default location on the container.

Procedure

1. Create a directory on your local host to store the configuration files.
2. Add the configuration files to replace in the container. The file name must match a corresponding configuration file in the image and include all required values. For example `cassandra.yaml`, `dse.yaml`, `opscenterd.conf`.

   See the GitHub pages for a full list of configuration files.

   - DSE
   - OpsCenter
   - Studio
3. Mount the local directory to the exposed `/config` directory during startup.

   For example:

   ```
   $ docker run -v /dse/conf:/config
   ```
4. Start the container.

For example, to start a transactional node:

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/config:/
config datastax/dse-server \ -d datastax/dse-server
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

5. After you make changes or add config files to the `/config` volume, restart the container to propagate the changes to the database.

```
$ docker restart container_name
```

Restarting the container restarts DSE and DDAC.

Using environment variables

Configure the DataStax Enterprise (DSE) Docker image by setting environment variables when the container is created. Use the docker run command `-e` option.

Table 3. Environment variables

Variable	Setting	Description
`DS_LICENSE`	accept	To show and acknowledge the license, set the variable `DS_LICENSE` to the value `accept`. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`LISTEN_ADDRESS`	IP_address	IP address to listen for connections from other nodes. Defaults to the container IP address.
`BROADCAST_ADDRESS`	IP_address	IP address to advertise to other nodes. Defaults to the same value as `LISTEN_ADDRESS`.
`NATIVE_TRANSPORT_ADDRESS`	IP_address	IP address to list for client and driver connections. Default: `0.0.0.0`.
`NATIVE_TRANSPORT_BROADCAST_ADDRESS`	IP_address	IP address to advertise to clients and drivers. Defaults to the same value as `BROADCAST_ADDRESS`.

Table 3. Environment variables (continued)

Variable	Setting	Description
`SEEDS`	IP_address	Comma-delimited list of seed nodes for the cluster. Defaults to the node `BROADCAST_ADDRESS`.
`START_NATIVE_TRANSPORT`	true | false	Determines whether to start the Thrift RPC server. If not set, the default value is preserved in the `cassandra.yaml` file.
`CLUSTER_NAME`	string	Name of the cluster. Default: `Test Cluster`.
`NUM_TOKENS`	int	Number of tokens randomly assigned to the node. Default: `8`.
`DC`	string	Datacenter name. Default: `Cassandra`.
`RACK`	string	Rack name. Default: `rack1`.
`OPSCENTER_IP`	IP_address | string	Address of the OpsCenter instance to use for DSE management. The value can be specified by linking the OpsCenter container using `opscenter` as the name.
`JVM_EXTRA_OPTS`	string	Sets a custom value for the JVM heap using `-Xmx` and `-Xms`.
`LANG`	string	Sets a custom locale.
`SNITCH`	string	Sets the snitch implementation this node will use. The value is set in the `endpoint_snitch` parameter in `cassandra.yaml`.

Persisting data

Persisting data allows the container to be deleted and recreated without losing data. To persist data, create directories on the local host and map the directory to the corresponding volume using the docker run command with the `-v` option. For example:

```
$ docker run -v local_directory:container_volume
```

Warning: If the volumes are not mounted from the local host, all data is lost when the container is removed.

DataStax exposes data volumes to preserve data. See volumes and data directories for a list of exposed volumes.

Procedure

1. Create a directory on the Docker host.
2. Bind mount the local directory to the configuration file that will be persisted by starting the container with the `-v` option.

   Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Example

Mounting a DSE data volume

Mount the DSE data volume to the `/dse/data` directory on the Docker host to ensure that the `/data`, `/commit_logs`, and `/saved_caches` directories will be available. Hosting the `/var/lib/cassandra` directory outside the container with the `-v` option allows the Docker container to be deleted and recreated without losing data.

```
$ docker run -e DS_LICENSE=accept --name my-dse -v /dse/data:/var/lib/
cassandra
```

Mounting a DSE configuration volume

Mount the host directory `/dse/config` to the DSE volume `/config` to manage configuration files.

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/conf:/config
datastax/dse-server \ -d datastax/dse-server
```

Mounting an OpsCenter configuration volume

Mount the local directory to the exposed volume `/var/lib/opscenter` by starting the container with the `-v` option.

```
$ docker run -e DS_LICENSE=accept \ -v /dse/data/opscenter:/var/lib/
opscenter \ --name my-opscenter \ -d datastax/dse-opscenter
```

Exposing public ports

To allow remote hosts to access a DataStax Enterprise (DSE) node, DSE OpsCenter, or DataStax Studio, map the DSE public port to a host port using the docker run command with the `-p` option.

For a complete list of ports see Securing DataStax Enterprise ports.

Note: When mapping a container port to a local host port, ensure the host port is not already in use by another container or the host.

Example

To allow access to OpsCenter from a browser on a remote host, open port `8888` as shown in the following example.

```
$ docker run -e DS_LICENSE=accept \ --name my-opscenter \ -p 8888:8888 \ -d
datastax/dse-opscenter
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Volumes and data directories

DataStax exposes the following volumes so that underlying directories can be mounted. If these volumes are not mounted, then sub-directories will be unavailable. For example, by mounting the `/var/lib/cassandra` directory, the `/data`, `/commit_logs`, `/saved_caches` directories will be available.

Tip: To persist data, create the directories on the local host, and then map the local directory to the corresponding volume using the `docker run -v` flag:

```
$ docker run -v local_directory:container_volume
```

See using volumes in the Docker documentation.

DataStax Enterprise (DSE)

Directory	Description
`/var/lib/cassandra`	Database data
`/var/lib/spark`	DSE Analytics with Spark data
`/var/lib/dsefs`	DSEFS data
`/var/log/cassandra`	Database logs
`/var/log/spark`	Spark logs
`/config`	Custom configuration files

Studio

Directory	Description
`/var/lib/datastax-studio`	DataStax Studio data

Attaching to a container

Use the `docker exec -it container_name` command to attach to a container and run DataStax Enterprise (DSE) tools and other operations.

Opening an interactive bash shell

If the container is running in the background (using the `-d` option), use the following command to open an interactive bash shell:

```
$ docker exec -it container_name bash
```

To exit the shell without stopping the container, type `exit`.

Opening an interactive CQL shell (cqlsh)

Use the following command to open the cqlsh prompt.

```
$ docker exec -it container_name cqlsh
```

To exit the shell without stopping the container, use `Ctrl + P + Q`.

Viewing logs

View DSE logs using the docker log command.

```
$ docker logs my-dse
```

Using DSE tools

Use the `docker exec` command to run other tools. For example:

```
$ docker exec -it my-dse nodetool status
```

See the DSE documentation for further information.

Using Docker compose for automated provisioning

Use Docker Compose to automate bootstrapping a multi-node cluster with DataStax Enterprise (DSE), DSE OpsCenter, and DataStax Studio. Use the following links to get sample `compose.yml` files for different tools and services:

- DSE
- DSE OpsCenter
- DataStax Studio

Three node configuration

When creating multiple nodes, use the `node` parameter to bootstrap one node at a time. For example, the first node is `node=0`, the second node is `node=1`, and the third node is `node=2`.

Wait for each node to finish bootstrapping before running `docker-compose` for the next node.

```
$ docker-compose -f docker-compose.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ up
-d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter and Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ -f
docker-compose.studio.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Single node configuration with Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.studio.yml \ up -d
--scale node=0
```

Single node configuration

To bootstrap a single-node cluster, use the `docker run` command and specify the version of DSE to install, plus any additional options.

```
$ docker run -e DS_LICENSE=accept -d datastax/dse-server:version \ options
```

Building custom Docker images

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

DataStax provides a code repository on Github for building custom Docker images.

By default, Gradle downloads the DataStax tarballs from the DataStax Downloads page.

Important: End User License Agreement (EULA). By downloading this DataStax product, you agree to the terms of the EULA.

To build all images, run the following command, entering DataStax Academy credentials for *username* and *password*.

```
$ ./gradlew buildImages
```

To build a product image for a specific version, invoke a Grade task that follows this pattern:

```
$ ./gradlew build<product><version>Image
```

For example:

```
$ ./gradlew buildServer6.0.6Image
```

To build more than one image with specific versions:

```
$ ./gradlew buildServer6.0.6Image buildOpscenter6.5.0Image
```

To get the list of all available tasks, run:

```
$ ./gradlew tasks
```

Multiple product versions

To support multiple product versions without duplicating files, Docker build contexts are generated from source folders that contain FreeMarker templates (files with `.ftl` extensions). The following conventions are used:

- Docker build contexts are generated from self-describing product folders. For example, `server`, `opscenter`, and `studio`.
- All files without the `.ftl` extension are copied to the build context.
- Files with `.ftl` extensions are processed as FreeMarker templates:
  - Template directives are written using angle bracket syntax.
  - Square bracket syntax is used for interpolations.
  - The processed files are copied to the build context without `.ftl` extension. For example, `Dockerfile.ftl` is copied as `Dockerfile`.
- FreeMarker templates use the `version` variable:
  - `version.major` returns product version major number
  - `version.minor` returns product version minor number
  - `version.bugfix` returns product version bugfix number
  - The following version functions are available:

  - `version.lowerThan('x.y.z')` returns `true` if `version` is semantically lower than `x.y.z`
  - `version.greaterEqualThan('x.y.z')` returns `true` if `version` is semantically greater than or equal to `x.y.z`.

To customize the products or to use multiple product versions, modify the templates in their corresponding product folder.

Getting help with Docker

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

Docker known issues

The following issues are recognized.

- Cassandra File System (CFS) is not supported.
- Lifecycle Manager (LCM) is not supported.
- Changing any file not included in the list of approved configuration files will require an additional host volume or customization of the image. An example is SSL key management.
- The JVM heap size must be set for DataStax Enterprise (DSE) running inside the container using the `JVM_EXTRA_OPTS` variable or custom `cassandra-env.sh`. If not set, Java does not honor resource limits set for the container, and will peer through the container to use resources (memory and CPU) of the host. See the `JVM_EXTRA_OPTS` variable in Using environment variables for more information.

Licensing

Review the licensing terms for each of the following products and services:

- DataStax License Terms

- DSE OpsCenter License Terms
- DataStax Studio License Terms

6. Docker Guide for DataStax 5.1

Use DataStax Docker images to create DataStax Enterprise (DSE) 5.1 server, DSE OpsCenter 6.1, and DataStax Studio 2.0 containers in production and non-production environments.

Getting started with DataStax and Docker

Use DataStax Docker images to create containers in production and non-production environments for development, learn DataStax Enterprise (DSE), DataStax OpsCenter, and DataStax Studio, try new ideas, and test and demonstrate an application. The following images are available:

- DDAC: DataStax Distribution of Apache Cassandra™ (DDAC) is a certified version of open source Apache Cassandra™ for development and production.
- DataStax Enterprise: The best distribution of Apache Cassandra™ with integrated Search, Analytics, and Graph, and Advanced Security capabilities.
- DataStax Studio: An interactive developer's tool for DataStax Enterprise which is designed to help your DSE database, Cassandra Query Language (CQL), DSE Graph, and Gremlin Query Language development.
- DSE OpsCenter: The web-based visual management and monitoring solution for DSE.

To get started, clone the repository and change for your environment.

Prerequisites

To use the Docker images, ensure the following requirements are met:

- Docker CE/EE 17.03 or later. Supported platforms:
  - Linux
  - Windows (See Running DSE on Microsoft Windows Using Docker.)
  - Mac
- Basic understanding of Docker images and containers.
- Docker installed on your local system.

Creating and starting Docker containers

Use the following information to create DataStax Enterprise (DSE) server, DSE OpsCenter, and DataStax Studio containers in production and non-production environments.

Creating a DataStax Enterprise container

Create a DataStax Enterprise (DSE) server container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Enabling advanced functionality

By default, the DSE server image is configured as a transactional (database) node. To configure the node with DSE advanced functionality, add the corresponding option that enables the intended feature to the end of the `docker run` command.

Combine startup options to run more than one feature.

Option	Description
`-g`	Enable and start DSE Graph.
`-k`	Enable and start DSE Analytics.
`-s`	Enable and start DSE Search.

Using specific DSE versions

The DSE version with the tag latest changes with each release. To avoid mixing DSE versions, DataStax recommends using a specific DSE version in the `docker run` command.

For example, use the following command to use DSE 6.0.10:

```
$ docker run datastax/dse-server:6.0.10
```

Examples

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Create a DSE database container

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server
```

Create a DSE container with Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -g
```

Create a DSE container with Analytics (Spark) enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -k
```

Create a DSE container with Search enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
```

Create a DSE container with Search, Analytics, and Graph enabled

```
$ docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server -s
-k -g
```

Creating an OpsCenter container

Create a DSE OpsCenter container and a connected DSE server container on the same Docker host. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create an OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ -p 8888:8888 \ --name my-
opscenter \ -d datastax/dse-opscenter
```

2. Create a DSE server container that is linked to the OpsCenter container.

```
$ docker run \ -e DS_LICENSE=accept \ --link my-opscenter:opscenter \
--name my-dse \ -d datastax/dse-server
```

3. Get the DSE container IP address.

```
$ docker exec -it my-dse nodetool status
```

4. Open a browser and navigate to http://*dse_container_ip*:8888, where *dse_container_ip* is the IP address of the OpsCenter container.
    a. Click **Manage existing cluster**.
    b. Enter the DSE container IP address in the **host name** field.
    c. Click **Install agents manually**. The agent is already installed on the DSE image, so no installation is required.

Results
OpsCenter is ready to use with DSE.

What's next:

What's next
See the DSE OpsCenter User Guide for detailed usage and configuration instructions.

Creating a Studio container

Create a DataStax Studio container. For a list of the most commonly used options, see Docker run options.

See the docker run command reference for a full list of options to run a command in a new container.

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Procedure

1. Create a Studio container, using `my-dse` as the hostname.

```
$ docker run -e DS_LICENSE=accept \ -p 9091:9091 \ --link my-dse \ --
name my-studio \ -d datastax/dse-studio
```

2. Open a browser and navigate to http://*studio_container_ip*:9091, where *container_IP* is the IP address of the container.

Results
Studio is ready to use with DSE.

What's next:

What's next
See the DataStax Studio documentation for detailed usage and configuration instructions.

Docker run options

The following options are the most commonly used when creating a DataStax container.

Option	Description
`-e`	Sets environment variables to accept the licensing agreement and change the initial configuration. **Required**. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`-d`	Starts the container in the background. Recommended.
`-p`	Publish container ports on the host to allow remote access to DSE, OpsCenter, and Studio. See exposing public ports for more information.
`-v`	Bind mount a directory on the local host to a DSE Volume to manage configuration or preserve data. See using exposed volumes for more information.
`--link`	Link a DSE container to OpsCenter, or Studio to DSE. For example, `--link my-opscenter:opscenter` or `--link my-dse`.
`--name`	Assign a name to the container.

Managing the configuration

Manage the DataStax Enterprise (DSE) configuration using one of the following options:

- The DSE configuration volume to get configuration files from a mounted host directory without replacing or customizing configuration file in the container.
- Environment variables to change the configuration at runtime.
- Docker file or directory volume mounts.
- Docker overlay file system.

Note: DSE and DDAC use the default values defined for the environment variables unless explicitly set at runtime. Custom configuration files override the default or explicitly set environment variables.

DataStax uses a common base image for all products. To customize the operating system or install additional packages, modify the `base/Dockerfile`. The DataStax base images use OpenJDK due to the end of public updates for Oracle JDK. All DataStax repositories on Docker Hub include OpenJDK.

Using the DSE configuration volume

Docker images provided by DataStax include a startup script that swaps DataStax Enterprise (DSE) configuration files found in the `/config` volume directory with the configuration file in the default location on the container.

Procedure

1. Create a directory on your local host to store the configuration files.
2. Add the configuration files to replace in the container. The file name must match a corresponding configuration file in the image and include all required values. For example `cassandra.yaml`, `dse.yaml`, `opscenterd.conf`.

   See the GitHub pages for a full list of configuration files.

   - DSE
   - OpsCenter
   - Studio
3. Mount the local directory to the exposed `/config` directory during startup.

   For example:

   ```
   $ docker run -v /dse/conf:/config
   ```
4. Start the container.

For example, to start a transactional node:

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/config:/
config datastax/dse-server \ -d datastax/dse-server
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

5. After you make changes or add config files to the `/config` volume, restart the container to propagate the changes to the database.

```
$ docker restart container_name
```

Restarting the container restarts DSE and DDAC.

Using environment variables

Configure the DataStax Enterprise (DSE) Docker image by setting environment variables when the container is created. Use the docker run command `-e` option.

Table 4. Environment variables

Variable	Setting	Description
`DS_LICENSE`	accept	To show and acknowledge the license, set the variable `DS_LICENSE` to the value `accept`. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`LISTEN_ADDRESS`	IP_address	IP address to listen for connections from other nodes. Defaults to the container IP address.
`BROADCAST_ADDRESS`	IP_address	IP address to advertise to other nodes. Defaults to the same value as `LISTEN_ADDRESS`.
`NATIVE_TRANSPORT_ADDRESS`	IP_address	IP address to list for client and driver connections. Default: `0.0.0.0`.
`NATIVE_TRANSPORT_BROADCAST_ADDRESS`	IP_address	IP address to advertise to clients and drivers. Defaults to the same value as `BROADCAST_ADDRESS`.

Table 4. Environment variables (continued)

Variable	Setting	Description
`SEEDS`	IP_address	Comma-delimited list of seed nodes for the cluster. Defaults to the node `BROADCAST_ADDRESS`.
`START_NATIVE_TRANSPORT`	true | false	Determines whether to start the Thrift RPC server. If not set, the default value is preserved in the `cassandra.yaml` file.
`CLUSTER_NAME`	string	Name of the cluster. Default: `Test Cluster`.
`NUM_TOKENS`	int	Number of tokens randomly assigned to the node. Default: `8`.
`DC`	string	Datacenter name. Default: `Cassandra`.
`RACK`	string	Rack name. Default: `rack1`.
`OPSCENTER_IP`	IP_address | string	Address of the OpsCenter instance to use for DSE management. The value can be specified by linking the OpsCenter container using `opscenter` as the name.
`JVM_EXTRA_OPTS`	string	Sets a custom value for the JVM heap using `-Xmx` and `-Xms`.
`LANG`	string	Sets a custom locale.
`SNITCH`	string	Sets the snitch implementation this node will use. The value is set in the `endpoint_snitch` parameter in `cassandra.yaml`.

Persisting data

Persisting data allows the container to be deleted and recreated without losing data. To persist data, create directories on the local host and map the directory to the corresponding volume using the docker run command with the `-v` option. For example:

```
$ docker run -v local_directory:container_volume
```

Warning: If the volumes are not mounted from the local host, all data is lost when the container is removed.

DataStax exposes data volumes to preserve data. See volumes and data directories for a list of exposed volumes.

Procedure

1. Create a directory on the Docker host.
2. Bind mount the local directory to the configuration file that will be persisted by starting the container with the `-v` option.

   Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Example

Mounting a DSE data volume

Mount the DSE data volume to the `/dse/data` directory on the Docker host to ensure that the `/data`, `/commit_logs`, and `/saved_caches` directories will be available. Hosting the `/var/lib/cassandra` directory outside the container with the `-v` option allows the Docker container to be deleted and recreated without losing data.

```
$ docker run -e DS_LICENSE=accept --name my-dse -v /dse/data:/var/lib/
cassandra
```

Mounting a DSE configuration volume

Mount the host directory `/dse/config` to the DSE volume `/config` to manage configuration files.

```
$ docker run -e DS_LICENSE=accept \ --name my-dse \ -v /dse/conf:/config
datastax/dse-server \ -d datastax/dse-server
```

Mounting an OpsCenter configuration volume

Mount the local directory to the exposed volume `/var/lib/opscenter` by starting the container with the `-v` option.

```
$ docker run -e DS_LICENSE=accept \ -v /dse/data/opscenter:/var/lib/
opscenter \ --name my-opscenter \ -d datastax/dse-opscenter
```

Exposing public ports

To allow remote hosts to access a DataStax Enterprise (DSE) node, DSE OpsCenter, or DataStax Studio, map the DSE public port to a host port using the docker run command with the `-p` option.

For a complete list of ports see Securing DataStax Enterprise ports.

Note: When mapping a container port to a local host port, ensure the host port is not already in use by another container or the host.

Example

To allow access to OpsCenter from a browser on a remote host, open port `8888` as shown in the following example.

```
$ docker run -e DS_LICENSE=accept \ --name my-opscenter \ -p 8888:8888 \ -d
datastax/dse-opscenter
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Volumes and data directories

DataStax exposes the following volumes so that underlying directories can be mounted. If these volumes are not mounted, then sub-directories will be unavailable. For example, by mounting the `/var/lib/cassandra` directory, the `/data`, `/commit_logs`, `/saved_caches` directories will be available.

Tip: To persist data, create the directories on the local host, and then map the local directory to the corresponding volume using the `docker run -v` flag:

```
$ docker run -v local_directory:container_volume
```

See using volumes in the Docker documentation.

DataStax Enterprise (DSE)

Directory	Description
`/var/lib/cassandra`	Database data
`/var/lib/spark`	DSE Analytics with Spark data
`/var/lib/dsefs`	DSEFS data
`/var/log/cassandra`	Database logs
`/var/log/spark`	Spark logs
`/config`	Custom configuration files

Studio

Directory	Description
`/var/lib/datastax-studio`	DataStax Studio data

Attaching to a container

Use the `docker exec -it container_name` command to attach to a container and run DataStax Enterprise (DSE) tools and other operations.

Opening an interactive bash shell

If the container is running in the background (using the `-d` option), use the following command to open an interactive bash shell:

```
$ docker exec -it container_name
```

To exit the shell without stopping the container, type `exit`.

Opening an interactive CQL shell (cqlsh)

Use the following command to open the cqlsh prompt.

```
$ docker exec -it container_name cqlsh
```

To exit the shell without stopping the container, use `Ctrl + P + Q`.

Viewing logs

View DSE logs using the docker log command.

```
$ docker logs my-dse
```

Using DSE tools

Use the `docker exec` command to run other tools. For example:

```
$ docker exec -it my-dse nodetool status
```

See the DSE documentation for further information.

Using Docker compose for automated provisioning

Use Docker Compose to automate bootstrapping a multi-node cluster with DataStax Enterprise (DSE), DSE OpsCenter, and DataStax Studio. Use the following links to get sample `compose.yml` files for different tools and services:

- DSE
- DSE OpsCenter
- DataStax Studio

Three node configuration

When creating multiple nodes, use the `node` parameter to bootstrap one node at a time. For example, the first node is `node=0`, the second node is `node=1`, and the third node is `node=2`.

Wait for each node to finish bootstrapping before running `docker-compose` for the next node.

```
$ docker-compose -f docker-compose.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ up
-d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Three node configuration with OpsCenter and Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.opscenter.yml \ -f
docker-compose.studio.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Single node configuration with Studio

```
$ docker-compose -f docker-compose.yml -f docker-compose.studio.yml \ up -d
--scale node=0
```

Single node configuration

To bootstrap a single-node cluster, use the `docker run` command and specify the version of DSE to install, plus any additional options.

```
$ docker run -e DS_LICENSE=accept -d datastax/dse-server:version \ options
```

Building custom Docker images

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

DataStax provides a code repository on Github for building custom Docker images.

By default, Gradle downloads the DataStax tarballs from the DataStax Downloads page.

Important: End User License Agreement (EULA). By downloading this DataStax product, you agree to the terms of the EULA.

To build all images, run the following command, entering DataStax Academy credentials for *username* and *password*.

```
$ ./gradlew buildImages
```

To build a product image for a specific version, invoke a Grade task that follows this pattern:

```
$ ./gradlew build<product><version>Image
```

For example:

```
$ ./gradlew buildServer6.0.6Image
```

To build more than one image with specific versions:

```
$ ./gradlew buildServer6.0.6Image buildOpscenter6.5.0Image
```

To get the list of all available tasks, run:

```
$ ./gradlew tasks
```

Multiple product versions

To support multiple product versions without duplicating files, Docker build contexts are generated from source folders that contain FreeMarker templates (files with `.ftl` extensions). The following conventions are used:

- Docker build contexts are generated from self-describing product folders. For example, `server`, `opscenter`, and `studio`.
- All files without the `.ftl` extension are copied to the build context.
- Files with `.ftl` extensions are processed as FreeMarker templates:
  - Template directives are written using angle bracket syntax.
  - Square bracket syntax is used for interpolations.
  - The processed files are copied to the build context without `.ftl` extension. For example, `Dockerfile.ftl` is copied as `Dockerfile`.
- FreeMarker templates use the `version` variable:
  - `version.major` returns product version major number
  - `version.minor` returns product version minor number
  - `version.bugfix` returns product version bugfix number
  - The following version functions are available:

  - `version.lowerThan('x.y.z')` returns `true` if `version` is semantically lower than `x.y.z`
  - `version.greaterEqualThan('x.y.z')` returns `true` if `version` is semantically greater than or equal to `x.y.z`.

To customize the products or to use multiple product versions, modify the templates in their corresponding product folder.

Getting help with Docker

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

Docker known issues

The following issues are recognized.

- Cassandra File System (CFS) is not supported.
- Lifecycle Manager (LCM) is not supported.
- Changing any file not included in the list of approved configuration files will require an additional host volume or customization of the image. An example is SSL key management.
- The JVM heap size must be set for DataStax Enterprise (DSE) running inside the container using the `JVM_EXTRA_OPTS` variable or custom `cassandra-env.sh`. If not set, Java does not honor resource limits set for the container, and will peer through the container to use resources (memory and CPU) of the host. See the `JVM_EXTRA_OPTS` variable in Using environment variables for more information.

Licensing

Review the licensing terms for each of the following products and services:

- DataStax License Terms

- DSE OpsCenter License Terms
- DataStax Studio License Terms

7. Docker Guide for DataStax Distribution of Apache Cassandra™ (DDAC)

Use DataStax Docker images to create DataStax Distribution of Apache Cassandra™ (DDAC) containers in production and non-production environments.

Getting started with DataStax Distribution of Apache Cassandra and Docker

Use DataStax Docker images to create containers in production and non-production environments for development, learn DataStax Enterprise (DSE), DataStax OpsCenter, and DataStax Studio, try new ideas, and test and demonstrate an application. The following images are available:

- DDAC: DataStax Distribution of Apache Cassandra™ (DDAC) is a certified version of open source Apache Cassandra™ for development and production.
- DataStax Enterprise: The best distribution of Apache Cassandra™ with integrated Search, Analytics, and Graph, and Advanced Security capabilities.
- DataStax Studio: An interactive developer's tool for DataStax Enterprise which is designed to help your DSE database, Cassandra Query Language (CQL), DSE Graph, and Gremlin Query Language development.
- DSE OpsCenter: The web-based visual management and monitoring solution for DSE.

To get started, clone the repository and change for your environment.

Prerequisites

To use the Docker images, ensure the following requirements are met:

- Docker CE/EE 17.03 or later. Supported platforms:
  - Linux
  - Windows (See Running DSE on Microsoft Windows Using Docker.)
  - Mac
- Basic understanding of Docker images and containers.
- Docker installed on your local system.

Creating and starting Docker containers

Create Docker containers for DataStax Distribution of Apache Cassandra™ (DDAC) in production and non-production environments. For a list of the most commonly used options, see Docker run options.

Creating a DataStax Distribution of Apache Cassandra™ (DDAC) container

To create a DataStax Distribution of Apache Cassandra™ (DDAC) database container:

```
$ docker run -e DS_LICENSE=accept --name my-ddac -d datastax/ddac
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

See the docker run command reference for a full list of options to run a command in a new container.

Docker run options

The following options are the most commonly used when creating a DataStax Distribution of Apache Cassandra (DDAC) container.

Option	Description
`-e`	Sets environment variables to accept the licensing agreement and change the initial configuration. **Required**. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`-d`	Starts the container in the background. Recommended.
`-p`	Publish container ports on the host to allow remote access to DDAC. See exposing public ports for more information.
`-v`	Bind mount a directory on the local host to a DSE Volume to manage configuration or preserve data. See using exposed volumes for more information.
`--link`	Link a DDAC container to Studio. For example, `--link my-studio`.
`--name`	Assign a name to the container.

Managing the configuration

Manage the DataStax Distribution of Apache Cassandra™ (DDAC) configuration using one of the following options:

- Environment variables to change the configuration at runtime.
- Docker file or directory volume mounts.
- Docker overlay file system.

Note: DSE and DDAC use the default values defined for the environment variables unless explicitly set at runtime. Custom configuration files override the default or explicitly set environment variables.

DataStax uses a common base image for all products. To customize the operating system or install additional packages, modify the `base/Dockerfile`. The DataStax base images use OpenJDK due to the end of public updates for Oracle JDK. All DataStax repositories on Docker Hub include OpenJDK.

Using environment variables

Configure the DataStax Distribution of Apache Cassandra™ (DDAC) Docker image by setting environment variables when the container is created. Use the docker run command `-e` option.

Table 5. Environment variables

Variable	Setting	Description
`DS_LICENSE`	accept	To show and acknowledge the license, set the variable `DS_LICENSE` to the value `accept`. **Important:** Setting the DS_LICENSE environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.
`LISTEN_ADDRESS`	IP_address	IP address to listen for connections from other nodes. Defaults to the container IP address.
`BROADCAST_ADDRESS`	IP_address	IP address to advertise to other nodes. Defaults to the same value as `LISTEN_ADDRESS`.
`NATIVE_TRANSPORT_ADDRESS`	IP_address	IP address to list for client and driver connections. Default: `0.0.0.0`.
`NATIVE_TRANSPORT_BROADCAST_ADDRESS`	IP_address	IP address to advertise to clients and drivers. Defaults to the same value as `BROADCAST_ADDRESS`.

Table 5. Environment variables (continued)

Variable	Setting	Description
`SEEDS`	IP_address	Comma-delimited list of seed nodes for the cluster. Defaults to the node `BROADCAST_ADDRESS`.
`START_NATIVE_TRANSPORT`	true | false	Determines whether to start the Thrift RPC server. If not set, the default value is preserved in the `cassandra.yaml` file.
`CLUSTER_NAME`	string	Name of the cluster. Default: `Test Cluster`.
`NUM_TOKENS`	int	Number of tokens randomly assigned to the node. Default: `8`.
`DC`	string	Datacenter name. Default: `Cassandra`.
`RACK`	string	Rack name. Default: `rack1`.
`JVM_EXTRA_OPTS`	string	Sets a custom value for the JVM heap using `-Xmx` and `-Xms`.
`LANG`	string	Sets a custom locale.
`SNITCH`	string	Sets the snitch implementation this node will use. The value is set in the `endpoint_snitch` parameter in `cassandra.yaml`.

Persisting data

Persisting data allows the container to be deleted and recreated without losing data. To persist data, create directories on the local host and map the directory to the corresponding volume using the docker run command with the `-v` option. For example:

```
$ docker run -v local_directory:container_volume
```

Warning: If the volumes are not mounted from the local host, all data is lost when the container is removed.

DataStax exposes data volumes to preserve data. See Volumes and data directories for a list of exposed volumes.

Procedure

1. Create a directory on the Docker host.

2. Bind mount the local directory to the configuration file that will be persisted by starting the container with the `-v` option.

   Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Example

Mounting a DDAC data volume

Mount the DataStax Distribution of Apache Cassandra™ (DDAC) data volume to the `/ddac/data` directory on the Docker host to ensure that the `/data`, `/commit_logs`, and `/saved_caches` directories will be available. Hosting the `/var/lib/cassandra` directory outside the container with the `-v` option allows the Docker container to be deleted and recreated without losing data.

```
docker run -e DS_LICENSE=accept --name my-ddac -v /ddac/data:/var/lib/
cassandra
```

Exposing public ports

To allow remote hosts to access a DataStax Enterprise (DSE) node, DSE OpsCenter, or DataStax Studio, map the DSE public port to a host port using the docker run command with the `-p` option.

For a complete list of ports see Securing DataStax Enterprise ports.

Note: When mapping a container port to a local host port, ensure the host port is not already in use by another container or the host.

Example

To allow access to OpsCenter from a browser on a remote host, open port `8888`:

```
$ docker run -e DS_LICENSE=accept \ --name my-opscenter \ -p 8888:8888 \ -d
datastax/dse-opscenter
```

Setting the `DS_LICENSE` environment variable signals your acceptance of the DataStax terms of service and is required for the software to start.

Volumes and data directories

DataStax exposes the following volumes so that underlying directories can be mounted. If these volumes are not mounted, then sub-directories will be unavailable. For example, by mounting the `/var/lib/cassandra` directory, the `/data`, `/commit_logs`, `/saved_caches` directories will be available.

Tip: To persist data, create the directories on the local host, and then map the local directory to the corresponding volume using the `docker run -v` flag:

```
$ docker run -v local_directory:container_volume
```

See Persisting data.

See using volumes in the Docker documentation.

Directory	Description
`/var/lib/cassandra`	Database data
`/var/log/cassandra`	Database logs
`/config`	Custom configuration files

Studio

Directory	Description
`/var/lib/datastax-studio`	DataStax Studio data

Attaching to a container

Use the `docker exec -it <container_name>` command to attach to a container and run DataStax Enterprise (DSE) tools and other operations.

Opening an interactive bash shell

If the container is running in the background (using the `-d` option), use the following command to open an interactive bash shell:

```
$ docker exec -it <container_name> bash
```

To exit the shell without stopping the container, type `exit`.

Opening an interactive CQL shell (cqlsh)

Use the following command to open the cqlsh prompt.

```
$ docker exec -it <container_name> cqlsh
```

To exit the shell without stopping the container, use `Ctrl + P + Q`.

Viewing logs

View DSE logs using the docker log command.

```
$ docker logs my-dse
```

Using DSE tools

Use the `docker exec` command to run other tools. For example:

```
$ docker exec -it my-dse nodetool status
```

See the DSE documentation for further information.

Using Docker compose for automated provisioning

Use Docker Compose to automate bootstrapping a multi-node cluster with DDAC. Get a sample `compose.yml` files for different tools and services. https://github.com/datastax/docker-images/tree/master/example_compose_yamls

Three node configuration

When creating multiple nodes, use the `node` parameter to bootstrap one node at a time. For example, the first node is `node=0`, the second node is `node=1`, and the third node is `node=2`.

Wait for each node to finish bootstrapping before running `docker-compose` for the next node.

```
$ docker-compose -f docker-compose.yml up -d --scale node=0
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=1
```

```
$ docker-compose -f docker-compose.yml up -d --scale node=2
```

Single node configuration

To bootstrap a single-node cluster, use the `docker run` command and specify any additional options.

```
$ docker run -e DS_LICENSE=accept options
```

Building custom Docker images

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

Use DataStax source code to build a custom Docker image for your environment.

To build an image:

- Clone the DataStax Docker Github repository.
- Make any desired configuration changes.
- Run the `./gradlew` command.

DataStax provides a code repository on Github for building custom Docker images.

By default, Gradle downloads the DataStax tarballs from the DataStax Downloads page.

Important: End User License Agreement (EULA). By downloading this DataStax product, you agree to the terms of the EULA.

To build all images, run the following command, entering DataStax Academy credentials for *username* and *password*.

```
$ ./gradlew buildImages
```

To get the list of all available tasks, run:

```
$ ./gradlew tasks
```

Multiple product versions

To support multiple product versions without duplicating files, Docker build contexts are generated from source folders that contain FreeMarker templates (files with `.ftl` extensions). The following conventions are used:

- Docker build contexts are generated from self-describing product folders. For example, `server`, `opscenter`, and `studio`.
- All files without the `.ftl` extension are copied to the build context.
- Files with `.ftl` extensions are processed as FreeMarker templates:
  - Template directives are written using angle bracket syntax.
  - Square bracket syntax is used for interpolations.
  - The processed files are copied to the build context without `.ftl` extension. For example, `Dockerfile.ftl` is copied as `Dockerfile`.

- FreeMarker templates use the `version` variable:
  - `version.major` returns product version major number
  - `version.minor` returns product version minor number
  - `version.bugfix` returns product version bugfix number
  - The following version functions are available:
    - `version.lowerThan('x.y.z')` returns `true` if `version` is semantically lower than `x.y.z`
    - `version.greaterEqualThan('x.y.z')` returns `true` if `version` is semantically greater than or equal to `x.y.z`.

To customize the products or to use multiple product versions, modify the templates in their corresponding product folder.

Getting help with Docker

To get help with DataStax Docker images:

- Ask questions and contribute answers in DataStax Community.
- Report issues on Github.
- View how-to and troubleshooting articles on DataStax Support Knowledge Base.
- Send an email message to techpartner@datastax.com.
- Explore free hands-on courses and role-based learning paths on DataStax Academy.

Docker known issues

The following issues are recognized.

- Cassandra File System (CFS) is not supported.
- Docker for DDAC does not support DataStax Studio or DSE OpsCenter.
- Changing any file not included in the list of approved configuration files will require an additional host volume or customization of the image. An example is SSL key management.
- The JVM heap size must be set for DataStax Enterprise (DSE) running inside the container using the `JVM_EXTRA_OPTS` variable or custom `cassandra-env.sh`. If not set, Java does not honor resource limits set for the container, and will peer through the container to use resources (memory and CPU) of the host. See Tuning

Java resources in the DDAC documentation and see the `JVM_EXTRA_OPTS` variable in Using environment variables for more information.

Licensing

Review the licensing terms for each of the following products and services:

- DataStax License Terms
- DSE OpsCenter License Terms
- DataStax Studio License Terms

8. DataStax Docker Images

Use DataStax Docker images to create containers in production and non-production environments.

Before downloading DataStax Docker images, download and install Docker from the Docker website.