

Managing YARN Docker Containers

Date published: 2020-02-11

Date modified: 2023-08-29



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

| | |
|---|---------------|
| Configuring YARN Docker Containers Support..... | 4 |
| Prerequisites for installing Docker..... | 4 |
| Recommendations for managing Docker containers on YARN..... | 4 |
| Installing Docker..... | 6 |
| Configuring Docker..... | 6 |
| Configuring YARN for managing Docker containers..... | 7 |
| Docker on YARN configuration properties..... | 7 |
| Running Dockerized Applications on YARN..... | 10 |
| Docker on YARN example: MapReduce job..... | 10 |
| Docker on YARN example: DistributedShell..... | 11 |
| Docker on YARN example: Spark-on-Docker-on-YARN..... | 12 |

Configuring YARN Docker Containers Support

You can configure YARN to run Docker containers.

Docker containerization provides isolation and enables you to run multiple versions of the same applications side-by-side. For example, you can have a stable production version of an application while you evaluate a test version.

On the other hand, using Docker containers introduces a new layer of virtualization, thus creates some overhead compared to regular containers.

Running Docker containers on YARN works very similar to running regular containers. Containers have access to files that are localized for the container as well as logging.

To facilitate the use of YARN features, the following rules need to be followed:

- The processes in the containers must run as the user submitting the application (or the local-user in insecure mode).
- The mount allowlist must include the `yarn.local.dirs` so that the files needed for the application are available in the container. This is ensured by Cloudera Manager.

Prerequisites for installing Docker

Docker's container management capabilities are highly dependent on the Linux OS kernel version and capabilities. As a result, Docker only supports systems with modern kernels. It is recommended to check with your operating system vendor for supported system configurations for use with Docker.

See the Support Matrix for the operating system requirements.

Recommendations for managing Docker containers on YARN

YARN expects that Docker is already installed on all NodeManager hosts where Docker containers will run. Consider these recommendations before installing and configuring Docker for use with YARN.

Docker Version

The minimum recommended version is Docker 1.12.5. Docker is rapidly evolving and shipping multiple releases per year. Not all versions of Docker have been tested. Docker versioning changed in 2017, and is now known as Docker CE. Cloudera recommends running a recent version of Docker CE.

Note that recent versions of Docker CE use the overlay2 storage driver which probably does not work for all workloads.

RHEL/CentOs provides a version of Docker that can be installed through yum.

Storage Driver

The storage driver you choose depends on OS kernel, workload, and Docker version. Cloudera recommends administrators to read the documentation, consult with their operating system vendor, and test the desired workload before making a decision.

Tests showed that device mapper using LVM is generally stable. Under high write load to the container's root filesystem, device mapper exhibited panics. SSDs for the Docker graph storage are recommended in this case, but care still needs to be taken. Overlay and overlay2 perform significantly better than device mapper and are recommended if the OS kernel and workload support it.

CGroup Support

YARN provides isolation through the use of cgroups. Docker also has cgroup management built in. If isolation through cgroups is desired, Cloudera recommends to use the cgroup management of YARN. YARN creates the cgroup hierarchy and set the the `--cgroup-parent` flag when launching the container.

The cgroupdriver must be set to cgroupfs. Ensure that Docker is running using the `--exec-opt native.cgroupdriver=cgroupfs` docker daemon option.



Note:

The Docker version included with RHEL/CentOS 7.2+ sets the cgroupdriver to systemd. You must change this, typically in the `docker.service` systemd unit file.

```
vi /usr/lib/systemd/system/docker.service
```

Find and add the following line to the cgroupdriver:

```
--exec-opt native.cgroupdriver=cgroupfs \
```

Also, this version of Docker may include oci-hooks that expect to use the systemd cgroupdriver. Search for `oci` on your system and remove these files. For example:

```
rm -f /usr/libexec/oci/hooks.d/oci-systemd-hook
rm -f /usr/libexec/oci/hooks.d/oci-register-machine
```

Networking

YARN has support for running Docker containers on a user specified network, however, it does not manage the Docker networks. Administrators have to create the networks before running the containers. Node labels can be used to isolate particular networks. It is crucial to read and understand the Docker networking documentation. Cloudera does not recommend swarm based options. Overlay networks can be used if the setup uses an external store, such as etcd.

YARN request the networking details, such as IP address and hostname, from the Docker. As a result, all networking types are supported. Set the `YARN_CONTAINER_RUNTIME_DOCKER_CONTAINER_NETWORK` environment variable to specify the network that is used.

Cloudera recommends to use host networking only for testing. If the network where the NodeManagers are running has a sufficient number of IP addresses, the bridge networking with `--fixed-cidr` option works properly. Each NodeManager is allocated with a small portion of the larger IP space, and then the NodeManagers allocate those IP addresses to containers.

If you want to use an administrator defined network, add the network to the `yarn.nodemanager.runtime.linux.dockerr.allowed-container-networks` property using Cloudera Manager.

Image Management

Images can be preloaded on all NodeManager hosts or they can be implicitly pulled at runtime if they are available in a public Docker registry, such as Docker hub. If the image does not exist on the NodeManager host and cannot be pulled, the container fails.

Docker Bind Mounted Volumes



Note:

Take extra care when enabling this feature. Enabling access to directories such as, but not limited to, /, /etc, /run, or /home is not advisable and can result in containers negatively impacting the host or leaking sensitive information.

Files and directories from the host are commonly needed within the Docker containers, which Docker provides through volumes. Examples include localized resources, Apache Hadoop binaries, and sockets. In order to make use of this feature, the following must be configured:

The administrator must define the volume allowlist by setting `docker.allowed.ro-mounts` and `docker.allowed.rw-mounts` to the list of parent directories that are allowed to be mounted.

The application submitter requests the required volumes at application submission time using the `YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS` environment variable.

The administrator supplied allowlist is defined as a comma separated list of directories that are allowed to be mounted into containers. The source directory supplied by the user must either match or be a child of the specified directory.

The user supplied mount list is defined as a comma separated list in the form `source:destination:mode`. This format is ensured when you use Cloudera Manager, as a validation error is displayed, if a mount does not satisfy this format. The source is the file or directory on the host. The destination is the path within the container where the source will be bind mounted. The mode defines the mode the user expects for the mount, which can be “ro” or “rw” representing read-only and read-write mode respectively.

Related Information

[Enable Cgroups](#)

Installing Docker

Identify the version of Docker provided by your operating system vendor and install it.

About this task

Cloudera recommends to install the version of Docker that is provided by your Operating System vendor. The Docker package is known by several names: `docker-engine`, `docker`, and `docker-ce`.

Procedure

- Install the package by entering the following command for the respective operating system:

| Operating System | Command for Installation |
|--|-------------------------------------|
| RHEL/CentOS/Oracle Linux 6 RHEL/CentOS/Oracle Linux 7 | <code>yum install docker</code> |
| SLES 12 SLES 11 | <code>zypper install docker</code> |
| Ubuntu 14 Ubuntu 16 | <code>apt-get install docker</code> |
| Debian 7 | <code>apt-get install docker</code> |

Configuring Docker

After Docker is installed, edit the Docker's daemon configuration (`daemon.json`) file to add the recommended configuration.

About this task

Note that configuring the storage driver is not included. That has to be added after reviewing the storage driver options.

Procedure

- Add the following options to the `/etc/docker/daemon.json` file:

```
{
    "live-restore" : true,
    "debug" : true
}
```

- Ensure that Docker is using the `cgroupfs cgroupdriver` option when YARN cgroups support is enabled.

`vi /usr/lib/systemd/system/docker.service`

Find and add the following line to the `cgroupdriver`:

`--exec-opt native.cgroupdriver=cgroupfs \`

Configuring YARN for managing Docker containers

Use Cloudera Manager to configure YARN for managing Docker containers.

Before you begin

- Install Docker binary to all hosts.
- If cgroups are enabled, use the `cgroupfs cgroupdriver` for cgroups support.
- Ensure that all host has running docker daemon

Procedure

1. In Cloudera Manager, select the YARN service.
2. Select the Configuration tab.
3. Find the Always Use Linux Container Executor property and select it. That enables the YARN service to use the Linux Container Executor.
4. Search for Allowed Linux Runtimes and add docker to the list.
5. Find the Enable Docker Containers property and select the role configuration groups on which you want to enable Docker containers.
6. If you installed the Docker binaries to a different folder than the default, set the Docker Binary Path property to the applicable folder.
7. Use the other properties under the Docker on YARN filter to fine-grain the configuration. For more information, see *Docker on YARN configuration properties*.
8. Click Save Changes.

Related Information

[Docker on YARN configuration properties](#)

Docker on YARN configuration properties

You can use the properties under the Docker on YARN filter in Cloudera Manager to configure the Docker on YARN feature.

| Name | Property | Default value | Description |
|--|---|---|--|
| Enable Docker Containers | | false (unselected) | Specifies if Docker containers in YARN are enabled. |
| Allowed Linux Runtimes | yarn.nodemanager.runtime.linux.allowed-runtimes | default | Specifies the runtimes that are allowed when LinuxContainerExecutor is used. |
| Sleep Delay Before SIGKILL | yarn.nodemanager.sleep-delay-before-sigkill.ms | 10 (ms) | Specifies the time in milliseconds between sending a SIGTERM and a SIGKILL signal to a running container. |
| Allowed Docker Container Networks | yarn.nodemanager.runtime.linux.docker.allowed-container-networks | host, none, bridge | Specifies the networks that are allowed for Docker containers. Valid values are determined by Docker networks available from the docker network ls command. |
| Default Docker Container Network | yarn.nodemanager.runtime.linux.docker.default-container-network | host | Specifies which allowed network is used when launching Docker containers but no network is specified in the request. This network must be added to yarn.nodemanager.runtime.linux.docker.allowed-container-network. |
| Allow Using Host PID Namespace for Docker Containers | yarn.nodemanager.runtime.linux.docker.host-pid-namespace.allowed | false (unselected) | Specifies if Docker containers are allowed to use the host PID namespace. |
| Allow Privileged Docker Containers | yarn.nodemanager.runtime.linux.docker.privileged-containers.allowed | false (unselected) | Specifies if applications are allowed to run in privileged containers. Privileged containers are granted the complete set of capabilities and are not subject to the limitations imposed by the device cgroup controller. Use with extreme care! |
| Allow Delayed Removal of Docker Containers | yarn.nodemanager.runtime.linux.docker.delayed-removal.allowed | false (unselected) | Specifies if Debug Deletion Delay is used for Docker containers. Debug Deletion Delay is useful for troubleshooting Docker container related launch failures. For more information, see yarn.nodemanager.delete.debug-delay-sec. |
| ACL for Privileged Docker Containers | yarn.nodemanager.runtime.linux.docker.privileged-containers.acl | Empty | A comma-separated list that specifies the users who can request privileged Docker containers if privileged Docker containers are allowed. |
| Docker Capabilities | yarn.nodemanager.runtime.linux.docker.capabilities | CHOWN,DAC_OVERRIDE,FSETID,FOWNER,MKNOD,NET_RAW,SETGID,SETUID,SETFCAP,SETPCAP,NET_BIND_SERVICE,SYS_CHROOT,KILL,AUDIT_WRITE | Specifies the capabilities assigned to Docker containers when they are launched. The values may not be case-sensitive from a docker perspective, but Cloudera recommends to keep them uppercase. |
| Enable User Remapping for Docker Containers | yarn.nodemanager.runtime.linux.docker.enable-userremapping.allowed | true | Specifies if Docker containers can run with the UID (User Identifier) and GID (Group Identifier) of the calling user. |

| Name | Property | Default value | Description |
|--|---|---|--|
| User Remapping UID Threshold for Docker Containers | yarn.nodemanager.runtime.linux.docker.userremapping-uid-threshold | 1 | Specifies the minimum UID (User Identifier) for a remapped user. Users with UIDs lower than this minimum value are not allowed to launch Docker containers when user remapping (yarn.nodemanager.runtime.linux.docker.enable-userremapping.allowed) is enabled. |
| User Remapping GID Threshold for Docker Containers | yarn.nodemanager.runtime.linux.docker.userremapping-gid-threshold | 1 | Specifies the minimum GID (Group Identifier) for a remapped user. Users with GIDs lower than this minimum value are not allowed to launch Docker containers when user remapping (yarn.nodemanager.runtime.linux.docker.enable-userremapping.allowed) is enabled. |
| Default Read-Only Mounts for Docker Containers | yarn.nodemanager.runtime.linux.docker.default-ro-mounts | Although, it is not visible on the UI, the NodeManager Local Directories and Cgroups root are always added to this list. | A list that specifies the default read-only mounts to be bind-mounted into all Docker containers that use DockerContainerRuntime. NodeManager Local Directories and Cgroups root are always added to this list. Ensure that any additional default read-only mounts are also added to the Allowed Read-Only Mounts list. |
| Default Read-Write Mounts for Docker Containers | yarn.nodemanager.runtime.linux.docker.default-rw-mounts | Although, it is not visible on the UI, the NodeManager Local Directories and NodeManager Container Log Directories are always added to this list. | A list that specifies the default read-write mounts to be bind-mounted into all Docker containers that use DockerContainerRuntime. NodeManager Local Directories and NodeManager Container Log Directories are always added to this list. Ensure that any additional default read-write mounts are also added to the Allowed Read-Write Mounts list. |
| Allowed Read-Only Mounts for Docker Containers | docker.allowed-ro-mounts | Although, it is not visible on the UI, the NodeManager Local Directories and Cgroups root are always added to this list. | Specifies the directories that Docker containers are allowed to mount in read-only mode. NodeManager Local Directories and Cgroups root are always added to this list. Ensure that any additional default read-only mounts are also added here. |
| Allowed Read-Write Mounts for Docker Containers | docker.allowed-rw-mounts | Although, it is not visible on the UI, the NodeManager Local Directories and NodeManager Container Log Directories are always added to this list. | Specifies the directories that Docker containers are allowed to mount in read-write mode. NodeManager Local Directories and NodeManager Container Log Directories are always added to this list. Ensure that any additional default read-write mounts are also added here. |

| Name | Property | Default value | Description |
|---|--|--|---|
| Default Tempfs Mounts for Docker Containers | yarn.nodemanager.runtime.linux.docker.default-tmpfs-mounts | Empty - no directories are allowed in tmpfs to be mounted. | Specifies the directories in tmpfs that Docker containers are allowed to mount. By default, no directories are allowed in tmpfs to be mounted. |
| Docker Binary Path | docker.binary | /usr/bin/docker | Specifies the path of the binary in the hosts that is used to launch Docker containers. |
| Allowed Devices for Docker Containers | docker.allowed.devices | Empty - no devices are allowed to be mounted. | Specifies the devices that Docker containers are allowed to mount. |
| Allowed Volume Drivers for Docker Containers | docker.allowed.volume-drivers | Empty - no volume drivers are allowed. | Specifies the volume drivers which are allowed to be used with Docker. |
| Enable No-new-privileges Flag for Docker Containers | docker.no-new-privileges.enabled | false (unselected) | Specifies if the no-new-privileges flag for docker run is enabled. The no-new-privileges flag ensures that the process or its children processes do not gain any additional privileges. |
| Trusted Registries for Docker Containers | docker.trusted-registries | Empty - no registries are defined. | A list that specifies the trusted docker registries for running trusted Docker containers. |

Related Information

[Configuring YARN for managing Docker containers](#)

Running Dockerized Applications on YARN

Usage examples for Docker on YARN.

Before using Docker on YARN ensure that you have enabled the Docker on YARN feature using Cloudera Manager. For more information, see [Configure YARN for managing Docker container](#).

If you encounter any problems, check [Troubleshooting Docker on YARN](#).

Docker on YARN example: MapReduce job

Learn how to run the Pi MapReduce example job in a Docker image. In a clean cluster where no fine-grained permissions are set, issues can occur.

Procedure

1. Prepare a UNIX-based Docker image with Java, preferably JDK8. For example, [ibmjava:8](#).
2. In Cloudera Manager, select the YARN service.
3. Click the Configuration tab.
4. Search for docker.trusted.registries and find the Trusted Registries for Docker Containers property.
5. Add library to the list of trusted registries to allow the ibmjava.
6. Search for map.output.
7. Find the Compression Codec of MapReduce Map Output property.
8. Change its value to org.apache.hadoop.io.compress.DefaultCodec.

9. Click Save Changes.
10. Restart the YARN service using Cloudera Manager.
11. Search for the hadoop-mapreduce-example jar in a Cloudera Manager manager host.
12. Set the YARN_JAR environment variable to the path of the hadoop-mapreduce-example jar.

For example, using the default value:

YARN_JAR=/opt/cloudera/parcels/CDH/jars/hadoop-mapreduce-examples-<jar version number>.jar

13. Start the Pi MapReduce job with the following command:

```
yarn jar $YARN_JAR pi \ -Dmapreduce.map.env="YARN_CONTAINER_RUNTIME_TYPE
=docker,YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=library/ibmjava:8" \
-Dmapreduce.reduce.env="YARN_CONTAINER_RUNTIME_TYPE=docker,YARN_CONTAIN
ER_RUNTIME_DOCKER_IMAGE=library/ibmjava:8" \
1 40000
```

Related Information

[Trouleshooting Docker on YARN](#)

Docker on YARN example: DistributedShell

Learn how to run arbitrary shell command through a DistributedShell YARN application.

Procedure

1. Prepare a UNIX-based Docker image. For example, [ubuntu:18.04](#).
2. In Cloudera Manager, select the YARN service.
3. Click the Configuration tab.
4. Search for docker.trusted.registries and find the Trusted Registries for Docker Containers property.
5. Add library to the list of trusted registries to allow ubuntu:18.04.
6. Click Save Changes.
7. Restart the YARN service using Cloudera Manager.
8. Search for the hadoop-yarn-applications-distributedshell jar in a Cloudera Manager manager host.
9. Set the YARN_JAR environment variable to the path of the hadoop-yarn-applications-distributedshell jar.

For example, using the default value:

YARN_JAR=/opt/cloudera/parcels/CDH/jars/hadoop-yarn-applications-distributedshell-<jar version number>.jar

10. Choose an arbitrary shell command.

For example “cat /etc/*-release” which displays OS-related information in UNIX-based systems.

11. Run the DistributedShell job providing the shell command in the -shell_command option:

```
sudo -u hdfs hadoop org.apache.hadoop.yarn.applications.distributedshell
.Client \
-jar $YARN_JAR \
-shell_command "cat /etc/*-release" \
-shell_env YARN_CONTAINER_RUNTIME_TYPE=docker \
-shell_env YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=library/ubuntu:18.04
```

12. Check the output of the command using yarn log command line tool:

```
sudo -u yarn yarn logs -applicationId <id of the DistributedShell application> -log_files stdout
```

The output should look like the following in case of the ubuntu image:

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.3 LTS"
NAME="Ubuntu"
VERSION="18.04.3 LTS (Bionic Beaver)"
...
```

Docker on YARN example: Spark-on-Docker-on-YARN

Learn how to submit a Spark application to run in Docker containers on YARN.

Procedure

1. Prepare a UNIX-based Docker image with Java and Python installed. For example, use any arbitrary docker image satisfying this condition or the one built from the following Dockerfile:

```
FROM centos
RUN yum -y install python36
RUN ln -s /usr/bin/python3.6 /usr/local/bin/python
RUN yum -y install java-1.8.0-openjdk
ENV JAVA_HOME /usr/lib/jvm/jre
```

2. In Cloudera Manager, select the YARN service.
3. Click the Configuration tab.
4. Use the Docker on YARN filter.
5. Find the Trusted Registries for Docker Containers property.
6. Add the registry of the docker image to the list of trusted registries.
7. Find the Allowed Read-Only Mounts for Docker Containers property.
8. Add /opt/cloudera/parcels, /etc/hadoop and /etc/passwd to the list of allowed read-only mounts.
9. Click Save Changes.
10. Restart the YARN service using Cloudera Manager.
11. Select an arbitrary python Spark application.

For example an application that initializes the SparkContext object and then prints the python version:

```
import sys
from pyspark import SparkConf, SparkContext
conf = SparkConf().setAppName("Version app").setMaster("yarn")
sc = SparkContext(conf=conf)
if sys.version_info[0] == 3:
    print("Python 3")
elif sys.version_info[0] == 2:
    print("Python 2")
```

12. Submit the python script to the cluster by typing the following command:

```
spark-submit \
  --master yarn \
  --deploy-mode cluster \
```

```
--conf spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_TYPE=docker \  
--conf spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=registry/image:tag \  
--conf spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/passwd:/etc/passwd:ro,/opt/cloudera/parcels:/opt/cloudera/parcels:ro,/etc/krb5.conf:/etc/krb5.conf:ro \  
--conf spark.executorEnv.YARN_CONTAINER_RUNTIME_TYPE=docker \  
--conf spark.executorEnv.YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=registry/image:tag \  
--conf spark.executorEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS="/etc/passwd:/etc/passwd:ro,/opt/cloudera/parcels:/opt/cloudera/parcels:ro,/etc/krb5.conf:/etc/krb5.conf:ro" \  
<path to python script>
```

13. Check the output of the script.

- a) Open the Spark history Server web UI.
- b) Search for the just submitted job.