Cloudera Runtime 7.1.9

Managing YARN Docker Containers

Date published: 2020-02-11
Date modified: 2023-08-29

https://docs.cloudera.com/

Legal Notice

Contents

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 containerConfigure YARN for managing Docker container.

If you encounter any problems, check Troubleshooting Docker on YARNTroubleshooting 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 numbe r>.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 applic
ation> -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=re
gistry/image:tag \
  --conf spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/et
c/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/p
asswd:/etc/passwd:ro,/opt/cloudera/parcels/:/opt/cloudera/parcels/:ro,/e
tc/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.