# Getting Started with the DAOS Hadoop Filesystem

Here, we describe the steps required to build and deploy the DAOS Hadoop filesystem, and the configurations to access DAOS in Spark and Hadoop. We assume that the DAOS servers and agents have already been deployed in the environment. Otherwise, they can be deployed by following the [DAOS Installation Guide](https://daos-stack.github.io/admin/installation/).

!!! note DAOS support for Spark and Hadoop is not available in DAOS 1.0. It is targeted for the DAOS 1.2 release.

## Build DAOS Hadoop Filesystem

Below are the steps to build the Java jar files for the DAOS Java and DAOS Hadoop filesystem. These jar files are required when running Spark and Hadoop. You can ignore this section if you already have the pre-built jars.

$ git clone https://github.com/daos-stack/daos.git  
$ cd daos  
$ git checkout <desired branch or commit>  
## assume DAOS is built and installed to <daos\_install> directory  
$ cd src/client/java  
$ mvn clean package -DskipITs -Ddaos.install.path=<daos\_install>

After build, the package daos-java-<version>-assemble.tgz will be available under distribution/target.

## Deploy DAOS Hadoop Filesystem

After unzipping daos-java-<version>-assemble.tgz, you will get the following files.

### daos-java-<version>.jar and hadoop-daos-<version>.jar

They need to be deployed on every compute node that runs Spark or Hadoop. Place them in a directory, e.g., $SPARK\_HOME/jars for Spark and $HADOOP\_HOME/share/hadoop/common/lib for Hadoop, which is accessible to all the nodes or copy them to every node.

### daos-site-example.xml

You have two choices, with or without UNS path, to construct DAOS URI. If you choose the second choice, you have to copy the file to your application config directory, e.g., $SPARK\_HOME/conf for Spark and $HADOOP\_HOME/etc/hadoop for Hadoop. Then do some proper configuration and rename it to daos-site.xml. For the second choice, daos-site.xml is optional. See next section for details.

## Configure DAOS Hadoop FileSystem

### DAOS Environment Variable

Export all DAOS related env variables and the following env variable in your application, e.g., spark-env.sh for Spark and hadoop-env.sh for Hadoop. The following env enables signal chaining in JVM to better interoperate with DAOS native code that installs its own signal handlers. It ensures that signal calls are intercepted so that they do not actually replace the JVM’s signal handlers if the handlers conflict with those already installed by the JVM. Instead, these calls save the new signal handlers, or “chain” them behind the JVM-installed handlers. Later, when any of these signals are raised and found not to be targeted at the JVM, the DAOS’s handlers are invoked.

$ export LD\_PRELOAD=<YOUR JDK HOME>/jre/lib/amd64/libjsig.so

### DAOS URIs

DAOS FileSystem binds to schema “daos”. DAOS URIs are in the format of “daos://[authority]//[path]”. Both authority and path are optional. There are two types of DAOS URIs, with and without DAOS UNS path depending on where you want the DAOS Filesystem to get initialized and configured.

#### With DAOS UNS Path

The simple form of URI is “daos:///<your uns path>[/sub path]”. “<your path>” is your OS file path created with the daos command or Java DAOS UNS method, DaosUns.create(). The “[sub path]” is optional. You can create the UNS path with below command.

$ daos cont create --pool <pool UUID> -path <your path> --type=POSIX

Or

$ java -Dpath="your path" -Dpool\_id="your pool uuid" -cp ./daos-java-1.1.0-shaded.jar io.daos.dfs.DaosUns create

After creation, you can use below command to see what DAOS properties set to the path.

$ getfattr -d -m - <your path>

#### Without DAOS UNS Path

The simple form of URI is “daos:///[sub path]”. Please check description of “fs.defaultFS” in [daos-site-example.xml](hadoop-daos/src/main/resources/daos-site-example.xml) for how to configure filesystem. In this way, preferred configurations are in daos-site.xml which should be put in right place, e.g., Java classpath, and loadable by Hadoop DAOS FileSystem.

If the DAOS pool and container have not been created, we can use the following command to create them and get the pool UUID, container UUID, and service replicas.

$ dmg pool create --scm-size=<scm size> --nvme-size=<nvme size>  
$ daos cont create --pool <pool UUID> --type POSIX

After that, configure daos-site.xml with the pool and container created.

<configuration>  
...  
 <property>  
 <name>fs.daos.pool.uuid</name>  
 <value>your pool UUID</value>  
 <description>UUID of DAOS pool</description>  
 </property>  
 <property>  
 <name>fs.daos.container.uuid</name>  
 <value>your container UUID</value>  
 <description>UUID of DAOS container created with "--type posix"</description>  
 </property>  
 <property>  
 <name>fs.daos.pool.svc</name>  
 <value>your pool service replicas</value>  
 <description>service list separated by ":" if more than one service</description>  
 </property>  
...  
</configuration>

You may want to connect to two DAOS servers or two DFS instances mounted to different containers in one DAOS server from same JVM. Then, you need to add authority to your URI to make it unique since Hadoop caches filesystem instance keyed by “schema + authority” in global (JVM). It applies to the both types of URIs described above.

### Tune More Configurations

If your DAOS URI is the mapped UUIDs, you can follow descriptions of each config item in [daos-site-example.xml](hadoop-daos/src/main/resources/daos-site-example.xml) to set your own values in loadable daos-site.xml.

If your DAOS URI is the UNS path, your configurations, except those set by DAOS UNS creation, in daos-site.xml can still be effective. To make configuration source consistent, an alternative to the configuration file daos-site.xml is to set all configurations to the UNS path. You put the configs to the same UNS path with below command.

# install attr package if get "command not found" error  
$ setfattr -n user.daos.hadoop -v "fs.daos.server.group=daos\_server:fs.daos.pool.svc=0" <your path>

Or

$ java -Dpath="your path" -Dattr=user.daos.hadoop -Dvalue="fs.daos.server.group=daos\_server:fs.daos.pool.svc=0"  
 -cp ./daos-java-1.1.0-shaded.jar io.daos.dfs.DaosUns setappinfo

For the “value” property, you need to follow pattern, key1=value1:key2=value2.. .. And key\* should be from [daos-site-example.xml](hadoop-daos/src/main/resources/daos-site-example.xml). If value\* contains characters of ‘=’ or ‘:’, you need to escape the value with below command.

$ java -Dop=escape-app-value -Dinput="daos\_server:1=2" -cp ./daos-java-1.1.0-shaded.jar io.daos.dfs.DaosUns util

You’ll get escaped value, “daos\_server”, for “daos\_server:1=2”.

If you configure the same property in both daos-site.xml and UNS path, the value in daos-sitem.xml takes priority. If user set Hadoop configuration before initializing Hadoop DAOS FileSystem, the user’s configuration takes priority.

### Configure Spark to Use DAOS

To access DAOS Hadoop filesystem in Spark, add the jar files to the classpath of the Spark executor and driver. This can be configured in Spark’s configuration file spark-defaults.conf.

spark.executor.extraClassPath /path/to/daos-java-<version>.jar:/path/to/hadoop-daos-<version>.jar  
spark.driver.extraClassPath /path/to/daos-java-<version>.jar:/path/to/hadoop-daos-<version>.jar

#### Access DAOS in Spark

All Spark APIs that work with the Hadoop filesystem will work with DAOS. We use the daos:// URI to access files stored in DAOS. For example, to read the people.json file from the root directory of DAOS filesystem, we can use the following pySpark code:

df = spark.read.json("daos://default:1/people.json")

### Configure Hadoop to Use DAOS

Edit $HADOOP\_HOME/etc/hadoop/core-site.xml to change fs.defaultFS to daos://default:1 or “daos://uns/<your path>”. Then append below configuration to this file and $HADOOP\_HOME/etc/hadoop/yarn-site.xml.

<property>  
 <name>fs.AbstractFileSystem.daos.impl</name>  
 <value>io.daos.fs.hadoop.DaosAbsFsImpl</value>  
</property>

DAOS has no data locality since it is remote storage. You need to add below configuration to the scheduler configuration file, like capacity-scheduler.xml in yarn.

<property>  
 <name>yarn.scheduler.capacity.node-locality-delay</name>  
 <value>-1</value>  
</property>

Then replicate daos-site.xml, core-site.xml, yarn-site.xml and capacity-scheduler.xml to other nodes.

#### Access DAOS in Hadoop

If everything goes well, you should see “/user” directory being listed after issuing below command.

$ hadoop fs -ls /

You can also play around with other Hadoop commands, like -copyFromLocal and -copyToLocal. You can also start Yarn and run some mapreduce jobs on Yarn. Just make sure you have DAOS URI, daos://default:1/, set correctly in your job.

#### Known Issues

If you use Omni-path PSM2 provider in DAOS, you’ll get connection issue in Yarn container due to PSM2 resource not being released properly in time.