How-to: Use the ShareLib in Apache Oozie (CDH 5)

May 12, 2014 (https://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/) | By Robert Kanter (https://blog.cloudera.com/blog/author/rkanter/) | 3 Comments (https://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/#comments)
Categories: How-to (https://blog.cloudera.com/blog/category/how-to/) Oozie (https://blog.cloudera.com/blog/category/oozie/)

The internals of Oozie's ShareLib have changed recently (reflected in CDH 5.0.0). Here's what you need to know.

In a previous blog post (http://blog.cloudera.com/blog/2012/12/how-to-use-the-sharelib-in-apache-oozie/) about one year ago, I explained how to use the Apache Oozie ShareLib in CDH 4. Since that time, things have changed about the ShareLib in CDH 5 (particularly directory structure), so some of the previous information is now obsolete. (These changes went upstream under OOZIE-1619 (https://issues.apache.org/jira/browse/OOZIE-1619).)

In this post I'll explain those changes. I recommend that you read the previous post for context first, because the reasoning behind the ShareLib (and other related information) is still relevant.

Directory Structure Changes

In CDH 4.x, the directory structure of the ShareLib looks like this:

```
/user/oozie/share/lib/distcp/*.jar
                     /hive/*.jar
                     /...
```

In CDH 5.x, there's now an additional level, which contains a timestamp:

```
/user/oozie/share/lib/lib_20140311155426/distcp/*.jar
                                        /hive/*.jar
                                        /...
```

The location of the ShareLib is still specified by the `oozie.service.WorkflowAppService.system.libpath` configuration property as before; the `lib_<timestamp>` directories will be created under that, as seen in the above example.

As you may have guessed, there can actually be multiple iterations of the these `lib_<timestamp>` directories; as you may have been able to tell from the above example, the timestamp format is `yyyymmddhhmmss`. The main reason for that is to address the following scenario.

Suppose you have some jobs running that are using the ShareLib. You decide that you want to update the ShareLib, but you don't want to wait for those jobs to finish. Previously, doing that could cause those jobs to fail because the distributed cache would get "confused". With the new directory structure, that won't happen — the already running jobs will continue to use the old ShareLib and any new jobs will use the latest ShareLib. On startup, Oozie will look for the newest `lib_<timestamp>` directory and use that.

Another benefit of the new ShareLib directory structure is that you can actually update the ShareLib without restarting Oozie! We'll see more about that later.

For reference, here's what the (MRv2) ShareLib looks like in CDH 5.0.0. Note that other than version numbers and the `lib_<timestamp>` directory, it looks similar to the CDH 4.1.2 ShareLib from the previous blog post.

```
drwxr-xr-x share/lib/lib_20140403151601/distcp
-rw-r--r-- share/lib/lib_20140403151601/distcp/hadoop-distcp-2.3.0-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/distcp/oozie-sharelib-distcp-4.0.0-cdh5.0.0.jar
drwxr-xr-x share/lib/lib_20140403151601/hcatalog
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/ST4-4.0.4.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/ant-1.8.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/ant-launcher-1.8.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/avro-1.7.5-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/avro-ipc-1.7.5-cdh5.0.0-tests.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/avro-ipc-1.7.5-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/avro-mapred-1.7.5-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/bonecp-0.7.1.RELEASE.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/commons-compress-1.4.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/commons-httpclient-3.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/commons-io-2.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/datanucleus-api-jdo-3.2.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/datanucleus-core-3.2.2.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/datanucleus-rdbms-3.2.1.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/groovy-all-2.1.6.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-ant-0.12.0-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-common-0.12.0-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-exec-0.12.0-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-hcatalog-core-0.12.0-cdh5.0.0.jar
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-hcatalog-pig-adapter-0.12.0-cdh5.0.0
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-hcatalog-server-extensions-0.12.0-cdh
-rw-r--r-- share/lib/lib_20140403151601/hcatalog/hive-metastore-0.12.0-cdh5.0.0.jar
```

Oozie will automatically clean up old ShareLib `lib_<timestamp>` directories based on the following rules:

1. After `ShareLibService.temp.sharelib.retention.days` days (default: 7)
2. Will always keep the latest 2

Currently, Oozie checks for stale directories only at startup, which means that it won't do any cleanup unless you restart Oozie. (We plan to improving this approach in a future release via OOZIE-1783 (https://issues.apache.org/jira/browse/OOZIE-1783).) In the meantime, you can safely delete older `lib_<timestamp>` directories from HDFS, as long as no jobs are currently using them. For most CDH users, this shouldn't be a huge problem because the ShareLib is typically only upgraded when upgrading CDH.

Installation Changes

In CDH 4.x, the installation directions said to essentially un-tar the ShareLib tarball and upload it to the `/user/oozie/share/lib` directory in HDFS. In the conclusion of the previous blog post, I had mentioned that we're always making improvements to the ShareLib, and specifically explained that OOZIE-1054 (https://issues.apache.org/jira/browse/OOZIE-1054) would provide a script that installs the ShareLib for you. While this feature has been supported since CDH 4.3, it wasn't readily available.

In CDH 5.0.0, we've now made the script available and it is the recommended way to install the ShareLib. It handles un-tarring the ShareLib, creating the `lib_<timestamp>` directory, and uploading it for you. **This is very important:** Simply uploading the ShareLib to `/user/oozie/share/lib` will no longer work because Oozie won't find the jars!

To use the script, you simply run this command:

```
oozie-setup sharelib create -fs FS_URI [-locallib SHARED_LIBRARY]
```

where `FS_URI` is the HDFS URI of the filesystem that the ShareLib should be installed (for example, `hdfs://<HOST<<PORT>`) and the optional `SHARED_LIBRARY` is the ShareLib tarball. In many cases the script will be able to find the ShareLib tarball for you, but in case it can't or if you want to use a different one (say the MRv1 version), you can specify it here. (There's also an `oozie-setup sharelib upgrade` command with the same arguments, but it's deprecated and currently does exactly the same thing as the `create` command.)

More details on the installation procedure can be found in the documentation here (http://blog.cloudera.com//content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/cdh5ig_oozie_configure.html). When upgrading from CDH 4.x to CDH 5.x, we recommend that you manually delete your ShareLib first so the CDH 4.x jars are not in the way.

Helpful Commands and Info

We've also added some additional ways to get information about the ShareLib.

The `admin -shareliblist` command can be used to list the ShareLib contents without having to go into HDFS and figure out which ShareLib Oozie is currently using. For example:

```
$ oozie admin -shareliblist
[Available ShareLib]
oozie
hive
distcp
hcatalog
sqoop
mapreduce-streaming
hive2
pig
```

And you can get a list of the jars in each of the ShareLibs like this:

```
$ oozie admin -shareliblist pig
[Available ShareLib]
pig
hdfs://rkanter-has-1.ent.cloudera.com:8020/user/oozie/share/lib/lib_20140403151601/pig/ant-
hdfs://rkanter-has-1.ent.cloudera.com:8020/user/oozie/share/lib/lib_20140403151601/pig/antl
hdfs://rkanter-has-1.ent.cloudera.com:8020/user/oozie/share/lib/lib_20140403151601/pig/antl
...
```

I mentioned earlier that you can now actually update the ShareLib while Oozie is running. This can be done with the `admin -sharelibupdate` command:

```
$ oozie admin -sharelibupdate
[ShareLib update status]
     host = rkanter-has-1.ent.cloudera.com:11000
     status = Successful
     sharelibDirOld = hdfs://rkanter-has-1.ent.cloudera.com:8020/user/oozie/share/lib/lib_2
     sharelibDirNew = hdfs://rkanter-has-1.ent.cloudera.com:8020/user/oozie/share/lib/lib_2
```

As you can see, this caused Oozie to switch to the latest `lib_<timestamp>` directory. This command also works with Oozie HA; if you run it once, each of the Oozie servers will look for the latest ShareLib.

You can also find info about the current ShareLib Oozie is using from the instrumentation log (which is also available on the Web UI, Hue, and REST API). The relevant properties are all under the "libs" group. You'll mostly be interested in these:

- `sharelib.keys`: ShareLibs loaded (e.g. "oozie", "hive", "pig", etc)
- `sharelib.source`: Indicates if the `system.libpath` or the mapping file is being used. (We'll look at the mapping file later.)
- `sharelib.system.libpath`: Path to the currently loaded sharelib

Overriding the ShareLib

I mentioned this in the old blog post, but I wanted to point it out again: If you want to change which ShareLib Oozie is using, or include multiple ShareLibs, for a particular action or action type, you can do so. You may have noticed that there is now an "hcatalog" sharelib. There isn't a new HCat action; this is a convenience for other actions that want to talk to HCat, and therefore need the HCatalog jars. A common use case is having Pig talk to HCatalog. You can override the ShareLib for an action type with the `oozie.action.sharelib.for.#ACTIONTYPE#` property. It can go in an action's `<configuration>` section, `job.properties`, or `oozie-site.xml` and with priority in that order as well. For example, if you want all Pig actions in one of your Workflows to include the HCatalog ShareLib, you would add `oozie.action.sharelib.for.pig=pig,hcatalog` to your `job.properties`.

ShareLib Mapping File

Another new feature is the ability to specify a "mapping file" instead of using the location at `oozie.service.WorkflowAppService.system.libpath` for the ShareLib. The location of the mapping file can be specified by `oozie.service.ShareLibService.mapping.file` in `oozie-site.xml`, which should contain a list of `=` entries. For example:

```
oozie.pig_10=hdfs:///share/lib/pig/pig-0.10.1/lib/
oozie.pig=hdfs:///share/lib/pig/pig-0.11.1/lib/
oozie.distcp=hdfs:///share/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-distcp-2.2.0.jar
```

Note that this is an advanced feature and completely optional. Most users should use the previously discussed ShareLib directories instead of the mapping file.

One Last Thing

I've seen a lot of confusion about how to include additional jars with your workflow and I'd like to use this opportunity to clarify. Below are the various ways to include a jar with your workflow:

1. Set `oozie.libpath=/path/to/jars,another/path/to/jars` in `job.properties`.
- This is useful if you have many workflows that all need the same jar; you can put it in one place in HDFS and use it with many workflows. The jars will be available to all actions in that workflow.
- There is no need to ever point this at the ShareLib location. (I see that in a lot of workflows.) Oozie knows where the ShareLib is and will include it automatically if you set `oozie.use.system.libpath=true` in `job.properties`.
2. Create a directory named "lib" next to your `workflow.xml` in HDFS and put jars in there.
- This is useful if you have some jars that you only need for one workflow. Oozie will automatically make those jars available to all actions in that workflow.
3. Specify the `<archive>` tag in an action with the path to a single jar; you can have multiple `<archive>` tags.
- This is useful if you want some jars only for a specific action and not all actions in a workflow.
- The downside is that you have to specify them in your workflow.xml, so if you ever need to add/remove some jars, you have to change your `workflow.xml.`
4. Add jars to the ShareLib (e.g. `/user/oozie/share/lib/lib_<timestamp>/pig`)
- While this will work, it's not recommended for two reasons:
- The additional jars will be included with every workflow using that ShareLib, which may be unexpected to those workflows and users.
- When upgrading the ShareLib, you'll have to recopy the additional jars to the new ShareLib.

Conclusion

At first, these changes may seem complicated and overwhelming. But just remember that, in a nutshell, all we did was add an extra level with a timestamp (the `lib_<timestamp>` directory). The ShareLib still works the same way as before and you don't have to update any of your workflows to continue using it. Other than the installation changes (which Cloudera Manager can handle for you), everything else is optional or provided to make things easier.

Further Reading

- CDH 5 Configuring Oozie Documentation (http://blog.cloudera.com//content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/cdh5ig_oozie_configure.html)
- Oozie Documentation on the new ShareLib admin commands (CLI) (http://archive.cloudera.com/cdh5/cdh/5/oozie/DG_CommandLineTool.html#Getting_list_of_available_sharelib)
- Oozie Documentation on the new ShareLib admin commands (REST) (http://archive.cloudera.com/cdh5/cdh/5/oozie/WebServicesAPI.html#List_available_sharelib)

Robert Kanter is a Software Engineer at Cloudera, and an Oozie Committer/PMC Member.

apache (https://blog.cloudera.com/blog/tag/apache/) Apache Oozie (https://blog.cloudera.com/blog/tag/apache-oozie/) Avro (https://blog.cloudera.com/blog/tag/avro/) CDH (https://blog.cloudera.com/blog/tag/cdh/) cloudera (https://blog.cloudera.com/blog/tag/cloudera/) Cloudera Manager (https://blog.cloudera.com/blog/tag/cloudera-manager/) configuration (https://blog.cloudera.com/blog/tag/configuration/) Hadoop (https://blog.cloudera.com/blog/tag/hadoop/) HBase (https://blog.cloudera.com/blog/tag/hbase/) HDFS (https://blog.cloudera.com/blog/tag/hdfs/) Hive (https://blog.cloudera.com/blog/tag/hive/) Hue (https://blog.cloudera.com/blog/tag/hue/) installation (https://blog.cloudera.com/blog/tag/installation/) java (https://blog.cloudera.com/blog/tag/java/) log (https://blog.cloudera.com/blog/tag/log/) MapReduce (https://blog.cloudera.com/blog/tag/mapreduce/) Oozie (https://blog.cloudera.com/blog/tag/oozie/) Pig (https://blog.cloudera.com/blog/tag/pig/) R (https://blog.cloudera.com/blog/tag/r/) release (https://blog.cloudera.com/blog/tag/release/) REST (https://blog.cloudera.com/blog/tag/rest/) Sqoop (https://blog.cloudera.com/blog/tag/sqoop/) ui (https://blog.cloudera.com/blog/tag/ui/)

3 responses on “How-to: Use the ShareLib in Apache Oozie (CDH 5)”

Nisha

August 29, 2014 at 12:35 pm

(https://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/#comment-46130)

Hi, thank you for the post.
I am seeing this exception in CDH5 VM , Could not locate oozie sharelib exception, so I was trying to execute the command on your blog, oozie-setup sharelib create -fs FS_URI [-locallib SHARED_LIBRARY]. But it is asking for the password. I tried default “cloudera” password it did not work. pls advise. Thank you!!

Justin Kestelyn (@kestelyn)

August 29, 2014 at 1:09 pm

(https://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/#comment-46131)

For fastest response, please post your question in the Oozie area at community.cloudera.com.

Hafizur Rahman

September 12, 2016 at 7:07 pm

(https://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/#comment-77146)

`sudo oozie-setup sharelib create -fs FS_URI [-locallib SHARED_LIBRARY]` should work

‹ How Apache Hadoop YARN HA Works (https://blog.cloudera.com/blog/2014/05/how-apache-hadoop-yarn-ha-works/)

How-to: Process Time-Series Data Using Apache Crunch › (https://blog.cloudera.com/blog/2014/05/how-to-process-time-series-data-using-apache-crunch/)