

# Using Java virtual machine (JVM) debugger with Apache Spark jobs in Cloudera Data Engineering (Preview)

Date published: 2022-11-23

Date modified: 2022-11-23

# Legal Notice

© Cloudera Inc. 2022. 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

<b>Using Java virtual machine (JVM) debugger with Apache Spark jobs in Cloudera Data Engineering (Preview)</b>	<b>1</b>
<b>Legal Notice</b>	<b>2</b>
<b>Contents</b>	<b>3</b>
Before you begin	4
Creating Cloudera CDP v2 credentials	4
Note the Role ARN	4
Setting up AWS accounts to run kubectl commands	4
Setting up Azure Virtual Clusters	5
Debugging with JVM	5
Flags to use with CDE CLI Job Run / CDE Spark Submit to Perform Debugging	5
Run a job in CDE	5
(Optional 1) CDE CLI - Job Run	5
(Optional 2) CDE CLI - Spark Submit	6
Starting the debugging session	6
Required flags for AWS jobs	6
Optional Flags:	6
Example Command For AWS	6
Example Command for Azure	7

Learn how to connect a Java virtual machine (JVM) debugger remotely to Spark pods (driver/executor) in Cloudera Data Engineering (CDE). You can use this JVM debugger to identify bugs or performance issues for Java/Scala Spark jobs where a JAR file is uploaded.

**Important:** This feature is not yet supported for Azure Private AKS and AWS Private EKS users.

## Before you begin

The user who performs the debugging must be a CDE Admin or must be given credential from a CDE Admin. There are two tasks that you need to complete before using the debugger:

1. Creating Cloudera CDP v2 credentials in the CDE CLI
2. Create an ARN role.
3. Setting up AWS accounts to run kubectl commands in the AWS console

Azure setup is not required.

## Creating Cloudera CDP v2 credentials

As a CDE Admin, set up the CDE CLI using API access keys before connecting the JVM debugger. See [Cloudera Data Engineering CLI authentication](#) for instructions on how to do this.

## Note the Role ARN

You'll need the Role ARN to set up the AWS accounts to run kubectl commands.

1. Go to the Cloudera **Management Console**.
2. Click **Environment**
3. Click the environment where your CDE Service is deployed.
4. Click **Summary**.
5. Note the **Role ARN** under **Credentials**.

## Setting up AWS accounts to run kubectl commands

If your environment has AWS EKS Virtual Clusters, as a system administrator, you must set up AWS accounts to run a Kubernetes port forward session before you use the JVM debugger. A system administrator must set up AWS accounts to run kubectl commands. These steps are to be performed once. After the setup is complete, IAM user credentials can be shared with those who want to use the JVM debugger. The credentials file must be stored at **\$HOME/.aws/credentials**.

1. In the AWS console, create an IAM user (for example, kubectl-user) with Programmatic access (you need not grant any permissions).
2. Write down the User ARN to use in step 5, and copy the Access key ID and Secret access key and set up an AWS profile as follows:

```
[kubectl-user]
```

```
aws_access_key_id = <Access Key ID>
```

```
aws_secret_access_key = <Secret access key>
```

3. Navigate to IAM Roles and edit the cross-account IAM role (note the Role ARN) that was created as part of the CDP prerequisites. You can note the Role ARN in the previous set of steps under Note the Role ARN.
4. Navigate to **Trust relationships > Edit trust relationships**.
5. Add the following to the policy document, then click **Update trust policy**.

```
"Effect": "Allow",  
"Principal": {  
  "AWS": "User ARN from step 2"  
},  
"Action": "sts:AssumeRole"  
  
},
```

6. Note the IAM User (that is, the kubectl-user) and Amazon Resource Names (ARNs) for all accounts because the user will need to provide the user names during debugging.
7. Install the **aws-iam-authenticator** CLI tool. See amazon docs: <https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html>

## Setting up Azure Virtual Clusters

No setup is needed.

## Debugging with JVM

### Flags to use with CDE CLI Job Run / CDE Spark Submit to Perform Debugging

Note the flags below to use in your debugging process:

- debug-driver** - To debug driver
- debug-executor** - To debug executor

### Run a job in CDE

There are two ways to run a job:

- CDE CLI - Job Run
- CDE CLI- Spark Submit

(Option 1) CDE CLI - Job Run

Run the job with debug flags from the CDE CLI:

```
$> cde job run --name java-spark --debug-driver
```

## (Option 2) CDE CLI - Spark Submit

Run the job using the spark submit command from the CDE CLI:

```
$> cde spark submit
learning-spark-with-java/target/learning-spark-with-java-1.0-SNAPSHOT
.jar debug-driver --class dataframe.LongSleep --conf
spark.executor.memory=2g --debug-driver=true
```

The job run process starts. If a debugger is not attached, then progress on the job run will not be made.

## Starting the debugging session

Run the following using the run id that resulted from running a job in option 1 or 2 above:

```
$> cde run debug -id run_id_of_job
```

## Required flags for AWS jobs

Use these required flags for jobs using AWS EKS Virtual Clusters:

```
--aws-profile (string eg: kubectl-user)
  * If not provided, the "default" profile will be chosen.

--aws-role-arn (string eg: cross_account_role_from_aws_prerequisites)
  * It is mandatory that this is provided for aws running jobs.
  An assume-role operation will be performed on this role using
  the IAM User credentials in $HOME/.aws/credentials (Check
  Special Setup in case of AWS VCs).
```

## Optional Flags:

If ports are not provided, a default port is assigned for both driver and executor. Run the following to assign a different port for each:

- --driver-port (int)
- --executor-port (int)

## Example Command For AWS

```
$> cde run debug --aws-role-arn
arn:aws:iam::302225162231:role/dex-priv-default-aws-cross-account-rol
e --aws-profile kubectl-user --id 20
```

## Example Command for Azure

```
$> cde run debug -id 20
```

The CLI command waits for Spark pods to display, and then a debug session to those pods opens in another window. Once a session is established, it will print localhost:local\_port for the debugger to connect to.