

Machine Learning Runtimes Release Notes

Date published: 2020-07-16

Date modified: 2024-11-21

CLOUDERA

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

- ML Runtimes What's New..... 4**
 - What's New in ML Runtimes version 2024.10.1..... 4
 - What's New in ML Runtimes version 2024.05.2..... 5
 - What's New in ML Runtimes version 2024.05.1..... 5
 - What's New in ML Runtimes older releases..... 6
 - What's New in ML Runtimes version 2024.02.1..... 6
 - ML Runtimes Version 2023.12.1..... 6
 - ML Runtimes Version 2023.08.2..... 7
 - ML Runtimes Version 2023.08.1..... 7
 - ML Runtimes Version 2023.05.2..... 7
 - ML Runtimes Version 2023.05.1..... 8
 - ML Runtimes Version 2022.11.2..... 9
 - ML Runtimes Version 2022.11..... 9
 - ML Runtimes Version 2022.04..... 10
 - ML Runtimes Version 2021.12..... 10
 - ML Runtimes Version 2021.09.02..... 11
 - ML Runtimes Version 2021.09..... 11
 - ML Runtimes Version 2021.06..... 11
 - ML Runtimes Version 2021.04..... 12
 - ML Runtimes Version 2021.02..... 12
 - ML Runtimes Version 2020.11..... 13
- ML Runtimes Known Issues and Limitations..... 13**
 - Known Issues and Limitations in ML Runtimes version 2024.10.01..... 13
 - Known Issues and Limitations in ML Runtimes version 2024.05.02..... 16
 - Known Issues and Limitations in ML Runtimes version 2024.05.01..... 19
 - Known Issues and Limitations in ML Runtimes older releases..... 22

ML Runtimes What's New

This section lists major features and updates for Machine Learning (ML) Runtimes.

**Note:**

Cloudera recommends upgrading to the latest Cloudera Machine Learning Runtimes available, as the latest version contains possible fixes that are not available in earlier versions. For finding the latest version of Cloudera Machine Learning Runtimes, see section *ML Runtimes Pre-installed Packages*. You can also check the latest version of Cloudera Machine Learning Runtimes in the [Archive Runtimes list](#).

Related Information

[Archive Runtimes](#)

[Adding new ML Runtimes](#)

[ML Runtimes Nvidia GPU Edition](#)

[Upgrading R and Python Packages](#)

[ML Runtimes Environment Variables](#)

What's New in ML Runtimes version 2024.10.1

Review the major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2024.10.1.



Note: New ML Runtimes releases are automatically added to the deployment, if Internet connection is available.

New Feature

Cloudera Copilot is generally available (GA): Cloudera Copilot is a highly configurable AI coding assistant integrated with the JupyterLab editor. The Copilot improves developer productivity by debugging code, answering questions and generating notebooks.

Cloudera Copilot is available in the Runtimes with JupyterLab Editor.

For more information, see [Using Cloudera Copilot](#).

New Runtimes

- Python 3.12 PBJ Workbench
- Python 3.12 PBJ NVIDIA GPU Workbench
- Python 3.12 JupyterLab
- Python 3.12 JupyterLab NVIDIA GPU



Note: Python 3.12 Runtimes are compatible only with Spark version 4.0.

Behavioral Changes

- JupyterLab Real-Time Collaboration (RTC) plug-in is not installed by default.
- On Workbench and Jupyterlab Runtimes, the `await_workers` function in the CDSW R and Python libraries will not time out when `timeout_seconds` is set to 0. Instead, the command will block until the workers are ready. Earlier, if the `timeout_seconds` set to 0 the function call timed out immediately. The new implementation matches the documented behaviour of this function.

Improvements

- Python maintenance versions - Python maintenance versions have been upgraded to 3.7.17, 3.8.19, 3.9.19, 3.10.14, and 3.11.9.
- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).
- Update setup tools in Conda runtime to resolve CVE-2024-6345.

What's New in ML Runtimes version 2024.05.2

Review the major features and updates for Machine Learning Runtimes.

This release is available with Machine Learning Runtimes version 2024.05.2.



Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

Behavioral Changes

- JupyterLab Real-Time Collaboration (RTC) plug-in is not installed by default.

Improvements

- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

What's New in ML Runtimes version 2024.05.1

Review the major features and updates for Machine Learning Runtimes.

This release is available with Machine Learning Runtimes version 2024.05.1.



Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- Cloudera Copilot is an AI-powered coding assistant designed for seamless integration within JupyterLab Machine Learning Runtimes. With its chat interface and comprehensive code completion features, Cloudera Copilot enhances the development experience for machine learning projects. It offers compatibility with both custom models deployed in Cloudera AI Inference Service and Amazon Bedrock models, providing developers with flexibility and efficiency in their workflows.
- JupyterLab is upgraded to 4.1.5 for Python 3.8 and for higher releases.
- Workbench R 4.4 Runtime is available.
- JupyterLab, PBJ Workbench Editor, and Nvidia GPU Runtime variants with Python 3.8 and higher versions are upgraded from CUDA 11.8 and cuDNN 8.7 versions to CUDA 12.3 and cuDNN 9.0 versions. It is recommended to install TensorFlow with the [and-cuda] option, to install TensorFlow's NVIDIA CUDA library dependencies as well, as the current latest TensorFlow version (2.16.1) still has dependencies on cuDNN8.

Due to the known TensorFlow issue <https://github.com/tensorflow/tensorflow/issues/63362> for installing TensorFlow 2.16.1 it is also needed to set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:\$LD_LIBRARY_PATH where python3.X stands for the used Python version, for example python3.9.

Fixed issues

- Python was missing from Scala Runtime, which could lead to warnings or the Scala engine not starting at all. Python interpreter is now added to Scala Runtime.
- With the upgrade of the PY4J library version to 0.10.9.5, the SPARK-37004 bug in Spark workloads is fixed, making those workloads more robust against kernel interruptions.

Improvements

- This release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

What's New in ML Runtimes older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Machine Learning Runtimes.

What's New in ML Runtimes version 2024.02.1

Review the major features and updates for Machine Learning Runtimes.

This release includes Machine Learning Runtimes version 2024.02.1.



Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- Pandas is now included in the Python 3.8 and higher Runtimes.
- You can now see hidden files in the JupyterLab file browser.

Improvements

- Multiple small improvements and bug fixes
- Multiple CVE fixes
- Added the following libraries to improve vision and audio processing model support:
 - libgl-dev
 - libjpeg-dev
 - libpng-dev
 - ffmpeg

ML Runtimes Version 2023.12.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.12.1.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New Runtimes

- Python 3.11 PBJ Standard Workbench
- Python 3.11 PBJ Nvidia GPU Workbench
- Python 3.11 JupyterLab Standard
- Python 3.11 JupyterLab Nvidia GPU



Note: Python 3.11 Runtimes are not compatible with Spark versions prior to Spark 3.4.

Improvements

- Python maintenance versions - Python maintenance versions have been upgraded to 3.8.18, 3.9.18, and 3.10.13.

Discontinued Runtimes

The following runtimes are no longer included.

- R 3.6 PBJ
- R 4.0 PBJ
- R 4.1 PBJ
- R 3.6 Workbench
- R 4.0 Workbench
- R 4.1 Workbench

Fixed Issue

Workbench editor for Python 3.8 and 3.9 now can parse multiline strings properly. (DSE-27222)

ML Runtimes Version 2023.08.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.08.2.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Improvements

- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

ML Runtimes Version 2023.08.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.08.1.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New Features / Improvements

- New Runtimes - Python 3.10 Standard Workbench, R 4.3 Standard Workbench, and R 4.3 Standard PBJ Workbench are included in this release.

Fixed Issues

- Chunker (DSE-27222) - Fixed an issue where the chunker failed to correctly parse multiline string inputs in classic Workbench Runtimes.
- Jupyter (DSE-28134) - `jupyter-kernel-gateway` has been upgraded to version 2.5.2. As a result, users cannot install incompatible versions of `jupyter-client` breaking their Python environments.
- Security - Various security fixes were implemented.

ML Runtimes Version 2023.05.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.05.2.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Fixed Issues

- This maintenance release is to automatically update those ML Runtimes versions that were referenced with a wrong image identifier when ML Runtimes 2023.05.1 was released.

ML Runtimes Version 2023.05.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.05.1.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- This release of ML Runtimes includes the following features that improve support for Large Language (LLMs) within CML:
 - Git Large File Storage (LFS) support is now included in all ML Runtimes images. Users can now easily download Large Language models via Git.
LFS replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.
 - Upgraded CUDA to version 11.8.0 in JupyterLab Python CUDA runtimes. This change allows customers to use the latest pytorch version in CML. Affected Python versions are 3.7, 3.8, 3.9, and 3.10.
 - NVIDIA GPU versions of PBJ Runtimes are now available for Python 3.7, 3.8, 3.9 and 3.10. The base image used for CUDA ML Runtimes are the “devel” series CUDA image provided by NVIDIA. This change, for example, enables the usage of the Numba JIT compiler.
This change also supports the latest version of tensorflow in CML and some other libraries that need to compile GPU code in run-time.
 - wget is now installed on all ML Runtimes. This enables users another way to download large language models in CML in addition to git-lfs.
- This release introduces a new Technical Preview Runtime, adding support for Anaconda, based on the JupyterLab editor. Conda environments can be created, modified, and activated from the Terminal without any limitations. For more information, see [Using Conda Runtimes](#).
- New Python 3.10 Runtimes are available.
- Added the following extensions to our JupyterLab Runtimes (Python 3.8, 3.9, 3.10): Language Server Protocol (jupyterlab-lsp) and git (jupyterlab-git). Language Server Protocol integration provides coding assistance for JupyterLab (code navigation, hover suggestions, linters, autocompletion, rename).
- Added the git (jupyterlab-git) extension to our JupyterLab Runtimes (Python 3.7).
- The Workbench Runtimes is being deprecated in favor of PBJ. As a result, the CUDA version for the Workbench Runtimes is not updated for the current release.
- Implya client is now updated to version 0.18.0 which has support for authentication via JWT token.

Version 2023.05.1

Fixed Issues

- DSE-26724 wget is now installed on all ML Runtimes. This enables users to download large language models in CML two ways: via Gi and via wget.
- DSE-25078 The OS package cmake is now installed for all ML Runtime images. This change enables users to install packages like nloptr.
- DSE-23762 Updated installed R package versions to make sparklyr R package usable in the same session as it was installed in.
- DSE-23751 Upgraded Jupyter to version 3.6.0 in all JupyterLab Runtimes.
- DSE-23613 Upgraded CUDA to version 11.8.0 in JupyterLab Python CUDA runtimes. This change was needed to allow customers to use the latest pytorch version in CML. Affected Python versions are 3.7, 3.8, 3.9, and 3.10.
- DSE-22425 Added OS packages to R Runtimes to support recent versions of the "devtools" package.

- DSE-20234 Jupyter session-specific data is no longer persisted in the Project directory. This prevents errors when starting multiple Jupyter sessions.

ML Runtimes Version 2022.11.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.11.2.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.11.2

Fixed Issues

- Multiple CVEs have been fixed.

ML Runtimes Version 2022.11

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.11



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.11

New features

- PBJ (Powered by Jupyter) Runtime GA: PBJ Runtimes were previously available as a Tech Preview only. In ML Runtimes 2022.11 the PBJ Runtimes are available to all users (GA). The following PBJ Runtimes are included:
 - Python 3.7
 - Python 3.8
 - Python 3.9
 - R 3.6
 - R 4.0
 - R 4.1

These PBJ Runtimes are available on Cloudera Public Cloud (from 2022.10-1) and Cloudera Private Cloud (from 1.5.0) form factors, but are not available on CDSW.

The new PBJ Runtimes appear on the Runtime Selector elements when users start new workloads. Their usage is the same as the non-PBJ Runtimes, apart from Models. Models require the usage of a new library called cml for Python and R, that was introduced in Cloudera Public Cloud (2022.10-1) and the upcoming Private Cloud (1.5.0) versions.

Usage of cml with regard to models is detailed here: [PBJ Runtimes and Models](#)

Examples can be found here: [Example models with PBJ Runtimes](#).

Fixed Issues

- DSE-22433 Upgrade numpy version in Python runtimes to resolve issues.
- DSE-22374 Reduced log level to WARN log level can be manipulated through the HADOOP_ROOT_LOGGER environment variable. This is set to WARN,console in order to keep console logging, while reducing log level from INFO to WARN.
- DSE-22251 Add flextable dependencies to R Runtimes Added flextable dependencies (libfontconfig1-dev, libcairo2-dev) to R runtimes. Added two OS libraries so flextable can be installed and used in CML.

Known issues

- The CUDA version (11.4.1) being used to build Nvidia GPU Runtimes is not supported by newer Torch versions (1.13+).

Workaround: Use Torch versions up to 1.12.1.

Cloudera Bug: DSE-24038

- When installing sparklyr for R, the installation will succeed, but importing the library in the same session will fail. This happens, because earlier versions of dependent packages (vctrs, lifecycle, rlang, cli) are already loaded in the R session, therefore newer versions cannot be loaded.

Workaround: Start a new session, import and use sparklyr there.

Cloudera Bug: DSE-23762

ML Runtimes Version 2022.04

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.04.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.04

New features

- *Powered By Jupyter (PBJ) Workbench* (Preview) - The Powered By Jupyter (PBJ) Workbench features the classic workbench UI backed by the open-source Jupyter protocol pre-packaged in an ML Runtime image. Users can easily choose this Runtime when launching a session. The open-source Jupyter infrastructure eliminates the dependency on proprietary CML code when building a Docker image, allowing runtime images to be built with open standards. The first PBJ Workbench Runtimes are available as Tech Preview.
- The Global Registry for npm is set to <https://registry.npmjs.org/> in all of our runtimes. With this change, customers can use npm install in their Dockerfiles, when they build custom runtimes on top of ours.



Note: RAPIDS Runtimes are now an optional extension of the Cloudera ML Runtime distribution. The RAPIDS images will no longer be distributed automatically. Administrators can Register them in the Runtime Catalog.

The Vim text editor is no longer supported for Cloudera provided ML Runtimes.

Fixed issues

- DSE-2064 Upgrade of Python Kernels across of all ML Runtimes:
 - Python 3.7.12 -> 3.7.13
 - Python 3.8.12 -> 3.8.13
 - Python 3.9.7 -> Python 3.9.11

ML Runtimes Version 2021.12

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2021.12.



Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2021.12

New features

- New versions provided for all ML Runtimes.
- Added R 4.1 Runtimes
- RAPIDS base image was updated to use 21.12 RAPIDS release (DSE-19745)



Note: Python 3.6 ML Runtimes are not longer provided due to Python 3.6 end of life

Fixed issues

- DSE-17817 psycpg2 now can be installed for Models.
- DSE-17547 Fixed a matplotlib warning in RAPIDS Runtimes.
- DSE-10146 Fixed issue causing R workbench to crash due to an illegal character.
- DSE-17657 Upgrade jupyterlab and notebook for CVE-2021-32797 and CVE-2021-32798.
- Various security and library version updates
- DSE-19805 Fixed CVE-2022-22817 critical vulnerability in Python Runtimes

ML Runtimes Version 2021.09.02

Major features and updates for ML Runtimes.

Version 2021.09.02

Fixed issues

- DSE-19496 The log4j build dependency is removed for the Scala ML Runtime. Once this ML Runtimes version is deployed, the user interface will always point to this latest version of the 2021.09.02 Scala Runtime to run workloads using the Scala kernel (until a newer version is released). Users will not be able to select the originally released 2021.09 Scala Runtime.

ML Runtimes Version 2021.09

Major features and updates for ML Runtimes.

Version 2021.09

New features

- Python 3.9 - ML Runtimes now support Python 3.9 kernels.
- Scala - Scala kernel is supported for running Sessions and Jobs.
- CUDA 11.4 - Nvidia GPU Editions are using CUDA 11.4.

Fixed issues

- DSE-17126 Starting a worker from Runtime session will create a worker with engine image

This issue is resolved in ML Runtimes 2021.09 when used with the latest ML Workspace versions.

For workers to function properly with ML Runtimes, please use ML Runtimes 2021.09 or later with CML Workspace version of 2.0.22 or later.

ML Runtimes Version 2021.06

Major features and updates for ML Runtimes.

Version 2021.06

New features

- Impyla - Impyla is now preinstalled in standard Python Runtimes.

Cloudera bug: DSE-14293

- Runtimes support for Models - Runtimes now support running models.

Cloudera bug: DSE-14294

- JupyterLab shutdown_no_activity_timeout configurable - JupyterLab shutdown_no_activity_timeout is now configurable through environment variable.

Cloudera bug: DSE-14817

Fixed issues

- DSE-15191 - Fixed installation issue with thrift-sasl python package.
- DSE-14886 - Fixed an issue preventing matplotlib installation into project.
- DSE-14737 - Fixed warnings appearing in the log due to using JupyterLab 3.
- DSE-15215 - Fixed an issue due to decorators that could not be used with Python 3.8 and Workbench editor.

ML Runtimes Version 2021.04

Major features and updates for ML Runtimes.

Version 2021.04

New features

- RAPIDS Runtimes - The RAPIDS Edition Runtimes are built on top of community built RAPIDS docker images. The RAPIDS suite of software libraries relies on NVIDIA CUDA primitives for low-level compute optimization, but exposes that GPU parallelism and high-bandwidth memory speed through user-friendly Python interfaces.

Fixed issues

The following fixed issues relate only to RAPIDS Runtimes.

- DSE-13743 - Idle JupyterLab sessions are now ended after around `IDLE_MAXIMUM_MINUTES`. (See *ML Runtimes Environment Variables Environment Variables*).



Note: These sessions may run for an additional 5 minutes after `IDLE_MAXIMUM_MINUTES`. (DSE-13743)

- DSE-14979 - matplotlib figures restore the styling used in engine:13 (MLRuntimes 2020.11 used the matplotlib defaults)
- DSE-12881 - For Python runtimes, py4j is now installed.
- Security fixes, python library version updates

ML Runtimes Version 2021.02

Major features and updates for ML Runtimes.

Version 2021.02

New features

- Nvidia GPU support - Runtimes supporting Nvidia GPUs are available with preinstalled CUDA software for both Workbench and JupyterLab editors. (See *ML Runtimes Nvidia GPU Edition*.)
- Runtimes supporting R 3.6 and R 4.0 available for Workbench editor



Note: User R libraries are installed in `/home/cdsw/.local/lib/R` rather than `/home/cdsw/R`, which is where they were installed in engine:13. This change was made to support simultaneous use of R 3.6 and R 4.0. Because of this change you will need to reinstall user R packages when migrating an existing project from engine:13 to MLRuntimes 2021.02, or from R 3.6 to R 4.0

Cloudera bug: DSE-3686

- Support for JupyterLab runtimes - The JupyterLab runtimes now use JupyterLab 3.0 (was 2.2) and are considered generally available using Standard Edition ML Runtimes. See [JupyterLab blog for notes on this upgrade](#).
- Python runtimes include a C++ compiler - Python runtimes now include a C++ compiler (g++), which enables the installation and use of certain Python libraries such as `impyla` and `pystan`.

Cloudera bug: DSE-14492

- Preinstalled Python libraries (See *ML Runtimes 2021.02*) are now installed in `/usr/local/lib` rather than `/var/lib/cdsw`, which is where they were installed in engine:13 and runtimes 2020.11 (DSE-12177). This means that you can upgrade these packages more easily, but there are some packages you should not upgrade. (See *Upgrading R and Python Packages*)
- ML runtimes are supported on CDSW (version 1.9.0 or higher).

Fixed issues

- DSE-13743 - Idle JupyterLab sessions are now ended after around `IDLE_MAXIMUM_MINUTES`. (See *ML Runtimes Environment Variables Environment Variables*).



Note: These sessions may run for an additional 5 minutes after `IDLE_MAXIMUM_MINUTES`. (DSE-13743)

- DSE-14979 - matplotlib figures restore the styling used in engine:13 (MLRuntimes 2020.11 used the matplotlib defaults)
- DSE-12881 - For Python runtimes, py4j is now installed.
- Security fixes, python library version updates

ML Runtimes Version 2020.11

Release notes and fixed issues

Version 2020.11

New features

- New Runtimes versions added



Note: Nvidia GPU Edition comes with CUDA 11.1 preinstalled.

Fixed issues

- DSE-13904 - Fixed an issue where CML workspace installation may take up to 10 minutes longer when the Autoscaling Group minimum is set to 0.
- DSE-13898 - Updated TGT image to fix an issue so that the freeIPA HA release retries on failure.
- CDPSPDX-2673 - Added a Retry step to the login context to reduce the chance of `PRE_AUTH` failures.

ML Runtimes Known Issues and Limitations

This section lists the known issues you might run into while using ML Runtimes.

Known Issues and Limitations in ML Runtimes version 2024.10.01

You might run into some known issues while using ML Runtimes 2024.10.01.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the `plot` function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable `R_LIBS_USER` must be passed to the Spark executors with the value set to `"/home/cdsw/.local/lib/R/<R_VERSION>/library"`.

E.g. when using `sparklyr` with a PBJ Workbench R 4.3 Runtime, the correct way to set up a `sparklyr` connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER = "/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in Cloudera Machine Learning fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in Cloudera Machine Learning fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the Cloudera Machine Learning project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable `SETUPTOOLS_USE_DISTUTILS=stdlib` either on a project level under Project Settings -> Advanced or on a workspace level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing `pip3 uninstall jupyter-client`. Verify your change by running `pip3 list` and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Technical Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like `!which python` will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the `-n` or `--name` argument to specify the environment, for example `conda -n myenv install pandas`
- When installing packages with pip, use the `%pip` magic to install packages in the active kernel's environment, for example `%pip install pandas`

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: !

```
source activate <conda_env_name> && python <job / application script.py>
```

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

For example:

```
mkdir -p /certs/
```

2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

Do not delete any existing objects.

```
#kubectl edit deployment runtime-manager
```

6. Under VolumeMounts, add the following lines.

Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
  name: mycert
  subPath: cert-chain.crt #this should match the new file name created in
  step 4
```

Under Volumes add the following text in the same edit:

```
- hostPath:
  path: /certs/ #this needs to match the folder created in step 1
```

```
type: ""
name: mycert
```

7. Save your changes:

wq!

Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.

8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: <https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241>

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on Cloudera Machine Learning require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes version 2024.05.02

You might run into some known issues while using ML Runtimes 2024.05.02.

TensorFlow 2.16.1 fails to work with GPUs

Due to the known TensorFlow issue <https://github.com/tensorflow/tensorflow/issues/63362>, it does not work with GPUs.

Workaround: Set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:\$LD_LIBRARY_PATH. Here, python3.X is the Python version you use, for example python3.9.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using sparklyr with a PBJ Workbench R 4.3 Runtime, the correct way to set up a sparklyr connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```


DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in CML fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in CML fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the CML project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable `SETUPTOOLS_USE_DISTUTILS=stdlib` either on a project level under Project Settings -> Advanced or on a workspace level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing `pip3 uninstall jupyter-client`. Verify your change by running `pip3 list` and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Tech Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like `!which python` will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the `-n` or `--name` argument to specify the environment, for example `conda -n myenv install pandas`
- When installing packages with pip, use the `%pip` magic to install packages in the active kernel's environment, for example `%pip install pandas`

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: !

```
source activate <conda_env_name> && python <job / application script.py>
```

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

For example:

```
mkdir -p /certs/
```

2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

Do not delete any existing objects.

```
#kubectl edit deployment runtime-manager
```

6. Under VolumeMounts, add the following lines.

Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
  name: mycert
  subPath: cert-chain.crt #this should match the new file name created in
  step 4
```

Under Volumes add the following text in the same edit:

```
- hostPath:
  path: /certs/ #this needs to match the folder created in step 1
```

```
type: ""
name: mycert
```

7. Save your changes:

wq!

Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.

8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: <https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241>

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on CML require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes version 2024.05.01

You might run into some known issues while using ML Runtimes 2024.05.01.

TensorFlow 2.16.1 fails to work with GPUs

Due to the known TensorFlow issue <https://github.com/tensorflow/tensorflow/issues/63362>, it does not work with GPUs.

Workaround: Set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:\$LD_LIBRARY_PATH. Here, python3.X is the Python version you use, for example python3.9.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using sparklyr with a PBJ Workbench R 4.3 Runtime, the correct way to set up a sparklyr connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in CML fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in CML fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the CML project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable `SETUPTOOLS_USE_DISTUTILS=stdlib` either on a project level under Project Settings -> Advanced or on a workspace level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing `pip3 uninstall jupyter-client`. Verify your change by running `pip3 list` and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Tech Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like `!which python` will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the `-n` or `--name` argument to specify the environment, for example `conda -n myenv install pandas`
- When installing packages with pip, use the `%pip` magic to install packages in the active kernel's environment, for example `%pip install pandas`

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: !

```
source activate <conda_env_name> && python <job / application script.py>
```

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

For example:

```
mkdir -p /certs/
```

2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

Do not delete any existing objects.

```
#kubectl edit deployment runtime-manager
```

6. Under VolumeMounts, add the following lines.

Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
  name: mycert
  subPath: cert-chain.crt #this should match the new file name created in
  step 4
```

Under Volumes add the following text in the same edit:

```
- hostPath:
  path: /certs/ #this needs to match the folder created in step 1
```

```
type: ""
name: mycert
```

7. Save your changes:

wq!

Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.

8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: <https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241>

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on CML require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes older releases

You might run into some known issues while using ML Runtimes.

TensorFlow 2.16.1 fails to work with GPUs

Due to the known TensorFlow issue <https://github.com/tensorflow/tensorflow/issues/63362>, it does not work with GPUs.

Workaround: Set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:\$LD_LIBRARY_PATH. Here, python3.X is the Python version you use, for example python3.9.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using sparklyr with a PBJ Workbench R 4.3 Runtime, the correct way to set up a sparklyr connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in CML fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in CML fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the CML project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable `SETUPTOOLS_USE_DISTUTILS=stdlib` either on a project level under Project Settings -> Advanced or on a workspace level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing `pip3 uninstall jupyter-client`. Verify your change by running `pip3 list` and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Tech Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like `!which python` will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the `-n` or `--name` argument to specify the environment, for example `conda -n myenv install pandas`
- When installing packages with pip, use the `%pip` magic to install packages in the active kernel's environment, for example `%pip install pandas`

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: !

```
source activate <conda_env_name> && python <job / application script.py>
```

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

For example:

```
mkdir -p /certs/
```

2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

Do not delete any existing objects.

```
#kubectl edit deployment runtime-manager
```

6. Under VolumeMounts, add the following lines.

Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
  name: mycert
  subPath: cert-chain.crt #this should match the new file name created in
  step 4
```

Under Volumes add the following text in the same edit:

```
- hostPath:
  path: /certs/ #this needs to match the folder created in step 1
```



```
type: ""  
name: mycert
```

7. Save your changes:

wq!

Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.

8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: <https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241>

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on CML require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.