Cloudera Data Science Workbench

Managing Engine Dependencies

Date published: 2020-02-28
Date modified: 2021-11-30

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

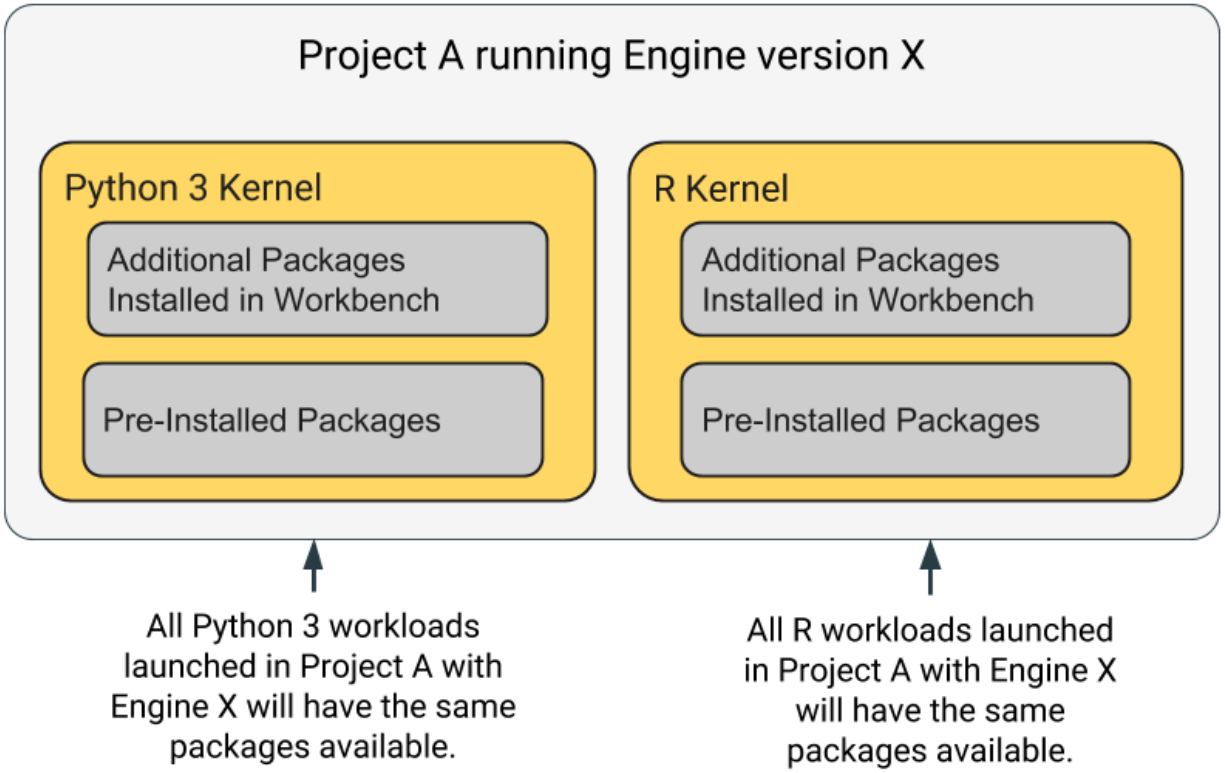
Managing Engine Dependencies

This page describes the options available to you for mounting a project's dependencies into its engine environment:

Important: Even though experiments and models are created within the scope of a project, the engines they use are completely isolated from those used by sessions or jobs launched within the same project.

Installing Packages Directly Within Projects

Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera Data Science Workbench allows you to install any other packages or libraries required by your projects just as you would on your local computer.



Each project's environment is completely isolated from others, which means you can install different versions of libraries pinned to different projects.

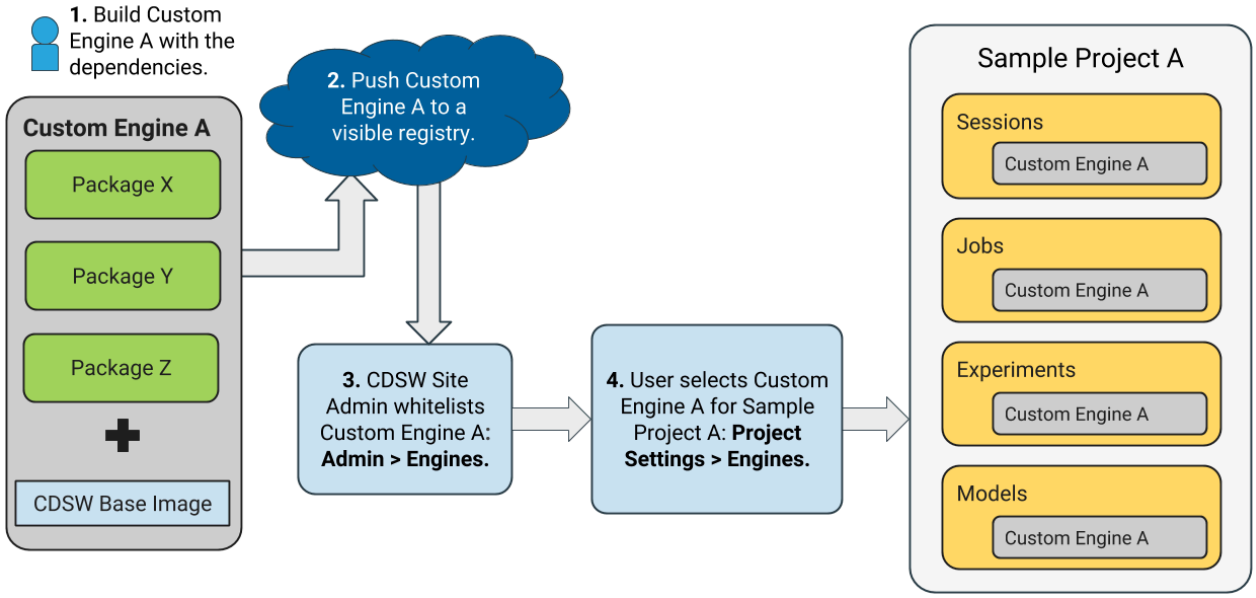
Libraries can be installed from the workbench using the inbuilt interactive command prompt or the terminal. Any dependencies installed this way are mounted to the project environment at /home/cdsw. Alternatively, you could choose to use a package manager such as Conda to install and maintain packages and their dependencies.

Note that overriding pre-installed packages by installing packages directly in the workbench can have unwanted side effects. It is not recommended or supported.

For detailed instructions, see Installing Additional Packaes.

Creating a Customized Engine with the Required Package(s)

Directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside /home/cdsw (outside the project mount), cannot be installed directly from the workbench. For such circumstances, Cloudera recommends you extend the base Cloudera Data Science Workbench engine image to build a customized image with all the required packages installed to it.

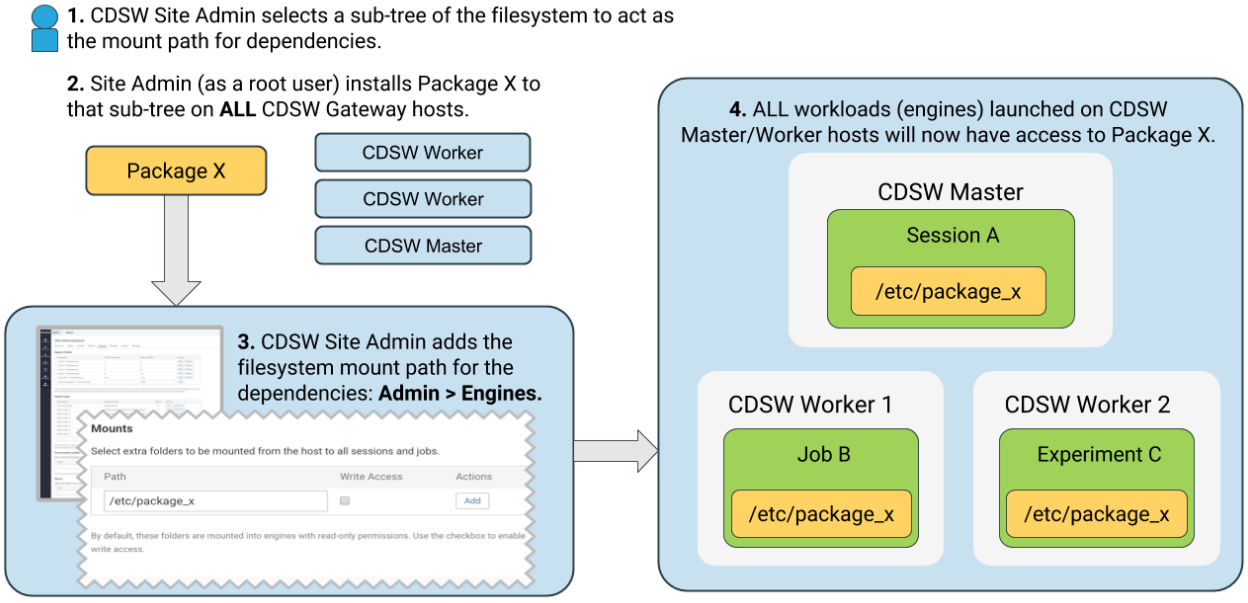


This approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies out of the box or if a package just has a complicated setup, it might be easier to simply provide users with an engine environment that has already been customized for their project(s).

For detailed instructions with an example, see Customized Engine Images.

Mounting Additional Dependencies from the Host

As described previously, all Cloudera Data Science Workbench projects run within an engine.



By default, Cloudera Data Science Workbench automatically mounts the CDH parcel directory and client configuration for required services, such as HDFS, Spark, and YARN, into the engine. However, if users want to reference any additional files/folders on the host, site administrators need to explicitly load them into engine containers at runtime.

Limitation

If you use this option, you must self-manage the dependencies installed on the gateway hosts to ensure consistency across them all. Cloudera Data Science Workbench cannot manage or control the packages with this method. For example, you must manually ensure that version mismatches do not occur. Additionally, this method can lead to issues with experiment repeatability since CDSW does not control or keep track of the packages on the host. Contrast this with Installing Packages Directory Within Projects and Creating a Customized Engine, where Cloudera Data Science Workbench is aware of the packages and manages them.

For instructions on how to configure host mounts, see Configuring the Engine Environment. Note that the directories specified here will be available to all projects across the deployment.

Managing Dependencies for Spark 2 Projects

With Spark projects, you can add external packages to Spark executors on startup.

To add external dependencies to Spark jobs, specify the libraries you want added by using the appropriate configuration parameters in a spark-defaults.conf file.

For a list of the relevant properties and examples, see Managing Dependencies for Spark 2 Jobs.