cloudera®

Cloudera Data Science Workbench

Cloudera, Inc.
395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Data Science Workbench 1.2.x
Date: June 7, 2021

Table of Contents

About Cloudera Data Science Workbench

Cloudera Data Science Workbench is a product that enables fast, easy, and secure self-service data science for the enterprise. It allows data scientists to bring their existing skills and tools, such as R, Python, and Scala, to securely run computations on data in Hadoop clusters. It is a collaborative, scalable, and extensible platform for data exploration, analysis, modeling, and visualization. Cloudera Data Science Workbench lets data scientists manage their own analytics pipelines, thus accelerating machine learning projects from exploration to production.

Benefits of Cloudera Data Science Workbench include:

Bringing Data Science to Hadoop

- Easily access HDFS data
- Use Hadoop engines such as Apache Spark 2 and Apache Impala

A Self-Service Collaborative Platform

- Use Python, R, and Scala from your web browser
- Customize and reuse analytic project environments
- Work in teams and easily share your analysis and results

Enterprise-Ready Technology

- Self-service analytics for enterprise business teams
- Ensures security and compliance by default, with full Hadoop security features
- Deploys on-premises or in the cloud

Core Capabilities

Core capabilities of Cloudera Data Science Workbench include:

- **For Data Scientists**

  Projects

  Organize your data science efforts as isolated projects, which might include reusable code, configuration, artifacts, and libraries. Projects can also be connected to Github repositories for integrated version control and collaboration.

  Workbench

  A workbench for data scientists and data engineers that includes support for:

  - Interactive user sessions with Python, R, and Scala through flexible and extensible *engines*.
  - Project workspaces powered by Docker containers for control over environment configuration. You can install new packages or run command-line scripts directly from the built-in terminal.
  - Distributing computations to your Cloudera Manager cluster using Cloudera Distribution of Apache Spark 2 and Apache Impala.
  - Sharing, publishing, and collaboration of projects and results.

  Jobs

  Automate analytics workloads with a lightweight job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts.

- **For IT Administrators**

  Native Support for the Cloudera Enterprise Data Hub

  Direct integration with the Cloudera Enterprise Data Hub makes it easy for end users to interact with existing clusters, without having to bother IT or compromise on security. No additional setup is required. They can just start coding.

Enterprise Security

Cloudera Data Science Workbench can leverage your existing authentication systems such as SAML or LDAP/Active Directory. It also includes native support for Kerberized Hadoop clusters.

Native Spark 2 Support

Cloudera Data Science Workbench connects to existing Spark-on-YARN clusters with no setup required.

Flexible Deployment

Deploy on-premises or in the cloud (on IaaS) and scale capacity as workloads change.

Multitenancy Support

A single Cloudera Data Science Workbench deployment can support different business groups sharing common infrastructure without interfering with one another, or placing additional demands on IT.

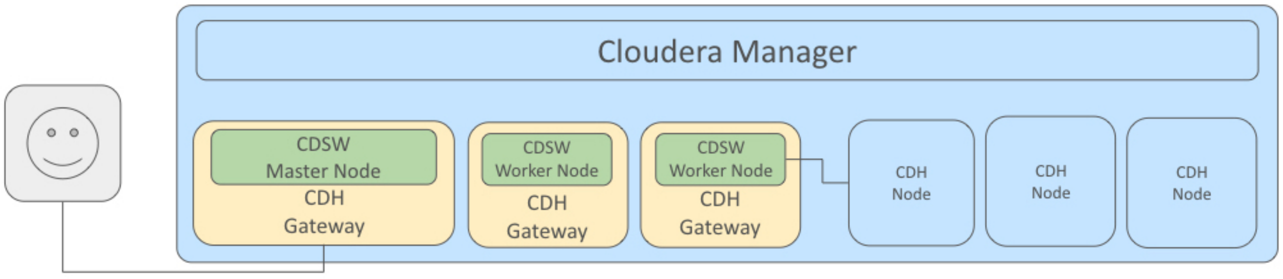
Architecture Overview

Important: The rest of this documentation assumes you are familiar with CDH and Cloudera Manager. If not, make sure you read the documentation for CDH and Cloudera Manager before you proceed.

Cloudera Manager

Important: Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0. Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

Cloudera Manager is an end-to-end application used for managing CDH clusters. When a CDH service (such as Impala, Spark, etc.) is added to the cluster, Cloudera Manager configures cluster hosts with one or more functions, called *roles*. In a Cloudera Manager cluster, a gateway role is one that designates that a host should receive client configuration for a CDH service even though the host does not have any role instances for that service running on it. Gateway roles provide the configuration required for clients that want to access the CDH cluster. Hosts that are designated with gateway roles for CDH services are referred to as gateway nodes.



Cloudera Data Science Workbench runs on one or more dedicated gateway hosts on CDH clusters. Each of these hosts has the Cloudera Manager Agent installed on them. The Cloudera Management Agent ensures that Cloudera Data Science Workbench has the libraries and configuration necessary to securely access the CDH cluster.

Cloudera Data Science Workbench does not support running any other services on these gateway nodes. Each gateway node must be dedicated solely to Cloudera Data Science Workbench. This is because user workloads require dedicated CPU and memory, which might conflict with other services running on these nodes. Any workloads that you run on Cloudera Data Science Workbench nodes will have immediate secure access to the CDH cluster.

From the assigned gateway nodes, one will serve as the *master* node, while others will serve as *worker* nodes.

Master Node

The master node keeps track of all critical persistent and stateful data within Cloudera Data Science Workbench. This data is stored at `/var/lib/cdsw`.

Project Files

Cloudera Data Science Workbench uses an NFS server to store project files. Project files can include user code, any libraries you install, and small data files. The NFS server provides a persistent POSIX-compliant filesystem that allows you to install packages interactively and have their dependencies and code available on all the Data Science Workbench nodes without any need for synchronization. The files for *all* the projects are stored on the master node at `/var/lib/cdsw/current/projects`. When a job or session is launched, the project's filesystem is mounted into an isolated Docker container at `/home/cdsw`.

Relational Database

The Cloudera Data Science Workbench uses a PostgreSQL database that runs within a container on the master node at `/var/lib/cdsw/current/postgres-data`.

Livelog

Cloudera Data Science Workbench allows users to work interactively with R, Python, and Scala from their browser and display results in realtime. This realtime state is stored in an internal database, called Livelog, which stores data at `/var/lib/cdsw/current/livelog`. Users do not need to be connected to the server for results to be tracked or jobs to run.

Worker Nodes

While the master node hosts the stateful components of the Cloudera Data Science Workbench, the worker nodes are transient. These can be added or removed which gives you flexibility with scaling the deployment. As the number of users and workloads increases, you can add more worker nodes to Cloudera Data Science Workbench over time.

Engines

Cloudera Data Science Workbench engines are responsible for running R, Python, and Scala code written by users and intermediating access to the CDH cluster. You can think of an engine as a virtual machine, customized to have all the necessary dependencies to access the CDH cluster while keeping each project's environment entirely isolated. To ensure that every engine has access to the parcels and client configuration managed by the Cloudera Manager Agent, a number of folders are mounted from the host into the container environment. This includes the parcel path `-/opt/cloudera`, client configuration, as well as the host's `JAVA_HOME`.

Cloudera Data Science Workbench allows you to run code in one of two ways, using either a *session* or a *job*. A session is a way to interactively launch an engine and execute code, whereas a job lets you batch process those actions and can be scheduled to run recursively. Each session or job gets its own Docker container that lives as long as the analysis. After a session or job is complete, the only artifacts that remain are a log of the analysis and any files that were generated or modified inside the project's filesystem, which is mounted to each engine at `/home/cdsw`.

Docker and Kubernetes

Cloudera Data Science Workbench uses Docker containers to deliver application components and run isolated user workloads. On a per project basis, users can run R, Python, and Scala workloads with different versions of libraries and system packages. CPU and memory are also isolated, ensuring reliable, scalable execution in a multi-tenant setting. Each Docker container running user workloads, also referred to as an *engine*, provides a visualized gateway with secure access to CDH cluster services such as HDFS, Spark 2, Hive, and Impala. CDH dependencies and client configuration, managed by Cloudera Manager, are mounted from the underlying gateway host. Workloads that leverage CDH services such as HDFS, Spark, Hive, and Impala are executed across the full CDH cluster.

To enable multiple users and concurrent access, Cloudera Data Science Workbench transparently subdivides and schedules containers across multiple nodes dedicated as gateway hosts. This scheduling is done using Kubernetes, a container orchestration system used internally by Cloudera Data Science Workbench. Neither Docker nor Kubernetes are directly exposed to end users, with users interacting with Cloudera Data Science Workbench through a web application.

Cloudera Data Science Workbench Web Application

The Cloudera Data Science Workbench web application is typically hosted on the master node, at `http://cdsw.<company>.com`. The web application provides a rich GUI that allows you to create projects, collaborate with your team, run data science workloads, and easily share the results with your team.

You can log in to the web application either as a site administrator or a regular user. See the Administration and User Guides respectively for more details on what you can accomplish using the web application.

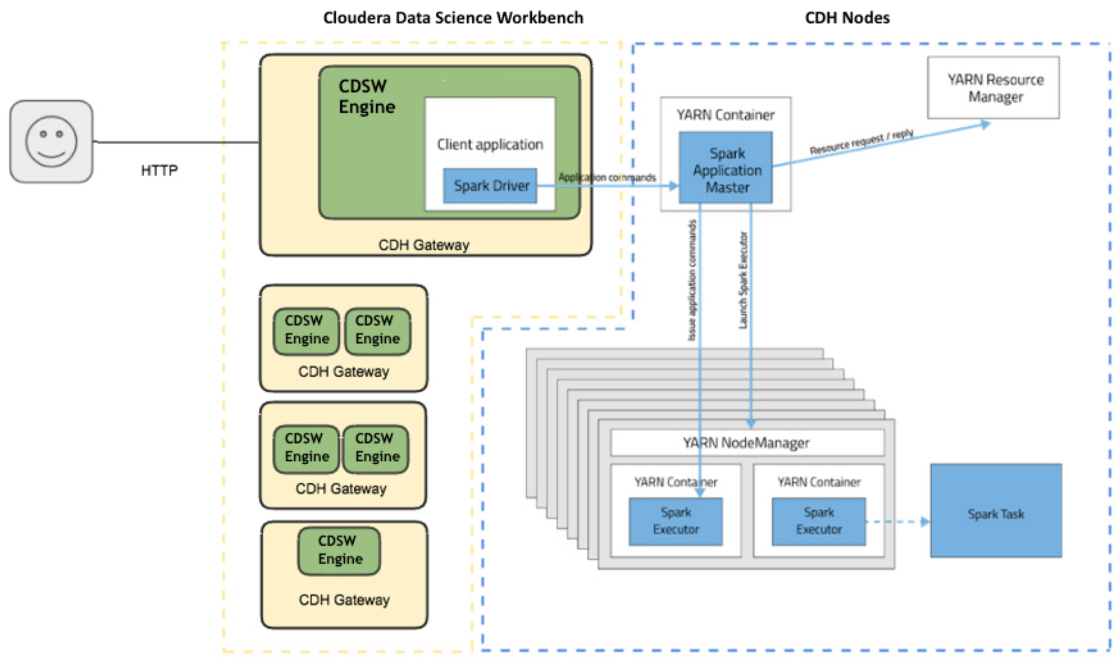
Cloudera Distribution of Apache Spark 2

Important: The rest of this topic assumes you are familiar with Apache Spark and Cloudera Distribution of Apache Spark 2. If not, make sure you read the Spark 2 documentation before you proceed.

Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.

Cloudera Data Science Workbench provides interactive and batch access to Spark 2. Connections are fully secure without additional configuration, with each user accessing Spark using their Kerberos principal. With a few extra lines of code, you can do anything in Cloudera Data Science Workbench that you might do in the Spark shell, as well as leverage all the benefits of the workbench. Your Spark applications will run in an isolated project workspace.

Cloudera Data Science Workbench's interactive mode allows you to launch a Spark application and work iteratively in R, Python, or Scala, rather than the standard workflow of launching an application and waiting for it to complete to view the results. Because of its interactive nature, Cloudera Data Science Workbench works with Spark on YARN's `client` mode, where the driver persists through the lifetime of the job and runs executors with full access to the CDH cluster resources. This architecture is illustrated the following figure:



More resources:

- Documentation for Cloudera Distribution of Spark 2
- Apache Spark 2 documentation

Cloudera Data Science Workbench Release Notes

These release notes provide information on new features, fixed issues and incompatible changes for all generally-available (GA) versions of Cloudera Data Science Workbench. For the current known issues and limitations, see Known Issues and Limitations in Cloudera Data Science Workbench 1.2.x on page 20.

Cloudera Data Science Workbench 1.2.2

This section lists the issues fixed in Cloudera Data Science Workbench 1.2.2.

New Features in Cloudera Data Science Workbench 1.2.2

- Added support for SUSE Linux Enterprise Server 12 SP2.
- Added support for multi-homed networks.
- Cloudera Director now allows you to deploy CSD-based Cloudera Data Science Workbench 1.2.x deployments on AWS. For more specifics on supported platforms, see Cloudera Altus Director Support (AWS Only) on page 33.
- Cloudera Data Science Workbench now ships version 4 of the base engine image which includes bug fixes related to Python development and Kerberos authentication. Make sure you upgrade existing projects to **Base Image v4** (**Admin** > **Engines**) to take advantage of these fixes.

  The new engine also changes how you configure and use Conda in Python sessions and extended engines. For more details, see Using Conda with Cloudera Data Science Workbench on page 59.
- Added a new environment variable called `MAX_TEXT_LENGTH` that allows you to set the maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated.

Issues Fixed In Cloudera Data Science Workbench 1.2.2

- Fixed an issue where Conda environmental variables were not being propagated to the Terminal correctly.

  Cloudera Bug: DSE-2256
- Fixed an issue where GPUs were not being detected by Cloudera Data Science Workbench due to incorrect mount settings.

  Cloudera Bug: DSE-2957
- Fixed an issue where jobs were failing due to Kerberos TGT renewal issues.

  Cloudera Bug: DSE-1007
- Fixed an issue on Internet Explorer 10 and 11 where the browser would fail to render console output after launching too many interactive sessions.

  Cloudera Bug: DSE-2998, DSE-2979
- Cloudera Data Science Workbench now correctly renders HTML that contains iFrames with the `srcdoc` attribute.

  Cloudera Bug: DSE-2034
- Fixed an issue where logging in using LDAP/Active Directory would sometimes crash the Cloudera Data Science Workbench web application.

  Cloudera Bug: DSE-2672
- The file tree in the Workbench now refreshes correctly when switching between sessions or launching a new session.

  Cloudera Bug: DSE-2829

- Fixed a file descriptors leak that would cause the "Failed to get Kubernetes client configuration" error in Cloudera Manager.

  Cloudera Bug: DSE-2910

- Fixed an issue where the host-controller process was consuming too much CPU. This was occurring due to a bug in the Kubernetes `client-go` library.

  Cloudera Bug: DSE-2993

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.2

For a list of the current known issues and limitations in Cloudera Data Science Workbench 1.2.x, see Known Issues and Limitations in Cloudera Data Science Workbench 1.2.x on page 20.

Cloudera Data Science Workbench 1.2.1

This section lists the issues fixed in Cloudera Data Science Workbench 1.2.1.

Issues Fixed In Cloudera Data Science Workbench 1.2.1

- The **Master Node IPv4 Address** parameter has been added to Cloudera Manager's Add Service wizard and is now a required parameter for installation on AWS. This should fix any related installation issues for deployments on AWS.

  Cloudera Bug: DSE-2879

- Fixed an issue with CSD-based deployments where certain operations would fail because the **Prepare Node** command was not installing all the required packages during First Run of the service. To see the updated list of packages that are now being installed by the **Prepare Node** command, refer the CSD install guide.

  Cloudera Bug: DSE-2869

- Fixed an issue where the `LD_LIBRARY_PATH` environmental variable was not getting propagated to CUDA engines.

  Cloudera Bug: DSE-2828

- Fixed an issue where stopping Cloudera Data Science Workbench on worker nodes resulted in the application hanging indefinitely.

  Cloudera Bug: DSE-2880

Incompatible Changes in Cloudera Data Science Workbench 1.2.1

Upgrading from Cloudera Data Science Workbench 1.2.0 to 1.2.1 on *CSD-based deployments*

After upgrading from Cloudera Data Science Workbench 1.2.0 to 1.2.1 on a CSD-based deployment, CLI commands might not work as expected due to missing binaries in the environment. Note that this issue does not affect fresh installs.

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.1

For a list of the current known issues and limitations in Cloudera Data Science Workbench 1.2.x, see Known Issues and Limitations in Cloudera Data Science Workbench 1.2.x on page 20.

Cloudera Data Science Workbench 1.2.0

This section lists the new features, fixed issues, and known issues in Cloudera Data Science Workbench 1.2.0.

New Features and Changes in Cloudera Data Science Workbench 1.2.0

- Cloudera Data Science Workbench is now available as an add-on service for Cloudera Manager. To this end, Cloudera Data Science Workbench is now distributed in a parcel that integrates with Cloudera Manager using a Custom Service Descriptor (CSD). You can use Cloudera Manager to install, upgrade, and monitor Cloudera Data Science Workbench. Diagnostic data bundles can be generated and submitted to Cloudera through Cloudera Manager.
- Cloudera Data Science Workbench now enables secure sharing of job and session consoles. Additionally, site administrators can disable anonymous sharing from the Site Administrator dashboard (**Admin** > **Security**). See Sharing Job and Session Console Outputs on page 70.
- The **Admin** > **Usage** page now includes graphs for monitoring usage activity such as number of CPUs or GPUs used, memory usage, and total session runs, over customizable periods of time. See Monitoring Site Usage on page 89.
- Cloudera Data Science Workbench now lets you configure session, job, and idle timeouts. These can be configured using environmental variables either for the entire deployment or per-project.
- The `cdsw enable` and `disable` commands are no longer needed. The master node will now automatically detect the IP addresses of worker nodes joining or leaving Cloudera Data Science Workbench. See the revised Cloudera Data Science Workbench Command Line Reference on page 120.
- The Kudu Python client is now included in the Cloudera Data Science Workbench base engine image.
- Interactive session names can now be modified by project contributors and admins. By default, session names are set to 'Untitled Session'.
- All-numeric usernames are now accepted.
- Cloudera Data Science Workbench now ships version 3 of the base engine image which includes `matplotlib` improvements and the Kudu client libraries. Make sure you upgrade existing projects to **Base Image v3** (**Admin** > **Engines**) to take advantage of the new features and bug fixes included in the new engine.

Issues Fixed in Cloudera Data Science Workbench 1.2.0

Privilege Escalation and Database Exposure in Cloudera Data Science Workbench

Several web application vulnerabilities allowed malicious authenticated Cloudera Data Science Workbench (CDSW) users to escalate privileges in CDSW. In combination, such users could exploit these vulnerabilities to gain root access to CDSW nodes, gain access to the CDSW database which includes Kerberos keytabs of CDSW users and bcrypt hashed passwords, and obtain other privileged information such as session tokens, invitations tokens, and environmental variables.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench 1.0.0, 1.0.1, 1.1.0, 1.1.1

Users affected: All users of Cloudera Data Science Workbench 1.0.0, 1.0.1, 1.1.0, 1.1.1

Date/time of detection: September 1, 2017

Detected by: NCC Group

Severity (Low/Medium/High): High

Impact: Privilege escalation and database exposure.

CVE: CVE-2017-15536

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.2.0 or higher.

Immediate action required: Upgrade to the latest version of Cloudera Data Science Workbench.

Other Notable Fixed Issues in Cloudera Data Science Workbench 1.2.0

- Fixed an issue where the Workbench editor screen jumps unexpectedly when typing or scrolling.
- Fixed auto-scroll behavior in the Workbench console. This was a browser compatibility issue that affected Chrome and Firefox, but not Safari.

- Fixed an issue where if a user logged out of Cloudera Data Science Workbench, and logged back in as a different user, they may see a `SecurityError` message in the Workbench.
- Fixed an issue that was preventing site administrators from uploading the SAML metadata file.
- Fixed several issues related to plotting with `matplotlib`. If you have previously used any workarounds for plotting, you might consider removing them now.
- Engines now use the same build of Kerberos utilities (`ktutil`, `kinit`, and `klist`) as the rest of Cloudera Data Science Workbench. This will improve logs obtained from kinit and make debugging Kerberos issues easier.
- KRB5_TRACE is now included in the error logs obtained when you `kinit`.
- Fixed an issue that was affecting health checks in deployments using AWS elastic load balancing.

Incompatible Changes in Cloudera Data Science Workbench 1.2.0

Proxy Configuration Change: If you are using a proxy server, you must ensure that the IP addresses for the web and Livelog services are skipped from the proxy.

Depending on your deployment (parcel or package), append the following IP addresses to either the **No Proxy** property in the Cloudera Manager CDSW service, or to the NO_PROXY parameter in `cdsw.conf`.

```
100.77.0.129
100.77.0.130
```

These have also been added to the installation instructions.

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.0

For a list of the current known issues and limitations in Cloudera Data Science Workbench 1.2.x, see Known Issues and Limitations in Cloudera Data Science Workbench 1.2.x on page 20.

Cloudera Data Science Workbench 1.1.1

This section lists the release notes for Cloudera Data Science Workbench 1.1.1. The documentation for version 1.1.x can be found at Cloudera Data Science Workbench 1.1.x.

New Features in Cloudera Data Science Workbench 1.1.1

- **Keytab Authentication** - With version 1.1.1, you can now authenticate yourself to the CDH cluster by uploading your Kerberos keytab to Cloudera Data Science Workbench. To use this feature, go to the top-right dropdown menu, click **Account settings** > **Hadoop Authentication**, enter your Kerberos principal and click **Upload Keytab**.

Issues Fixed In Cloudera Data Science Workbench 1.1.1

- Fixed an issue with airgapped installations where the installer could not pull the alpine 3.4 image into the airgapped environment.
- Fixed an issue where Cloudera Data Science Workbench would fail to log a command trace when the Kerberos process exits.
- Fixed authentication issues with older versions of MIT KDC.

Known Issues and Limitations in Cloudera Data Science Workbench 1.1.1

For a list of known issues and limitations, refer the documentation for version 1.1.x at Cloudera Data Science Workbench 1.1.x.

Cloudera Data Science Workbench 1.1.0

This section lists the release notes for Cloudera Data Science Workbench 1.1.0. The documentation for version 1.1.x can be found at Cloudera Data Science Workbench 1.1.x.

New Features and Changes in Cloudera Data Science Workbench 1.1.0

- Added support for RHEL/CentOS 7.3 and Oracle Linux 7.3.
- Cloudera Data Science Workbench now allows you to run GPU-based workloads. For more details, see Using GPUs for Cloudera Data Science Workbench Workloads on page 91.
- For Cloudera Manager and CDH clusters that are not connected to the Internet, Cloudera Data Science Workbench now supports fully offline installations. See the installation guide for more details.
- Web UIs for processing frameworks such as Spark 2, Tensorflow, and Shiny, are now embedded in Cloudera Data Science Workbench and can be accessed directly from active sessions and jobs. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 64.
- Added support for a Jobs REST API that lets you orchestrate jobs from 3rd party workflow tools. See Cloudera Data Science Workbench Jobs API on page 71.
- DataFrames are now scrollable in the workbench session output pane. For examples, see the section on Grid Displays on page 68.
- Cloudera Data Science Workbench now ships version 2 of the base engine image that includes newer versions of Pandas, seaborn, and assorted bug fixes.
- Added support for rich visualizations in Scala engine using Jupyter jvm-repr. For an example, see HTML Visualizations - Scala.
- `JAVA_HOME` is now set in `cdsw.conf`, and not from the Site Administrator dashboard (**Admin** > **Engines**).

Issues Fixed in Cloudera Data Science Workbench 1.1.0

- Improved support for dynamic data visualizations in Python, including Bokeh.
- Fixed issues with the Python template project. The project now supports offline mode and will therefore work on airgapped clusters.
- Fixed issues related to cached responses in Internet Explorer 11.
- Fixed issues with Java symlinks outside of `JAVA_HOME`.
- The `cdsw status` command can now be run on worker nodes.
- Removed unauthenticated localhost access to Kubernetes.
- Fixed Kerberos authentication issues with specific enc-types and Active Directory.
- Removed restrictions on usernames with special characters for better compatibility with external authentication systems such as Active Directory.
- Fixed issues with LDAP configuration validation that caused application crashes.
- Improved LDAP test configuration form to avoid confusion on parameters being sent.

Incompatible Changes in Cloudera Data Science Workbench 1.1.0

- **Upgrading from version 1.0.x to 1.1.x**

  During the upgrade process, you will encounter incompatibilities between the two versions of `cdsw.conf`. This is because even though you are installing the latest RPM, your previous configuration settings in `cdsw.conf` will remain unchanged. Depending on the release you are upgrading from, you will need to modify `cdsw.conf` to ensure it passes the validation checks run by the 1.1.x release.

  Key changes to note:

    - `JAVA_HOME` is now a required parameter. Make sure you add `JAVA_HOME` to `cdsw.conf` before you start Cloudera Data Science Workbench.
    - Previous versions allowed `MASTER_IP` to be set to a DNS hostname. If you are still using a DNS hostname, switch to an IP address.

- **Python engine updated in version 1.1.x**

  Version 1.1.x includes an updated base engine image for Python which no longer uses the deprecated `pylab` mode in Jupyter to import the numpy and `matplotlib` functions into the global scope. With version 1.1.x, engines will now use built-in functions like any rather than the `pylab` counterpart, `numpy.any`. As a result of this change, you might see certain behavioral changes and differences in results between the two versions.

  Also note that Python projects originally created with engine 1 will be running pandas version 0.19, and will not auto-upgrade to version 0.20 by simply selecting engine 2. You will also need to manually install version 0.20.1 of pandas when you launch a project session.

Known Issues and Limitations in Cloudera Data Science Workbench 1.1.0

For a list of known issues and limitations, refer the documentation for version 1.1.x at Cloudera Data Science Workbench 1.1.x.

Cloudera Data Science Workbench 1.0.1

This section lists the release notes for Cloudera Data Science Workbench 1.0.1. The documentation for version 1.0.x can be found at Cloudera Data Science Workbench 1.0.x.

Issues Fixed in Cloudera Data Science Workbench 1.0.1

- Fixed a random port conflict that could prevent Scala engines from running.
- Improved formatting of validation, and visibility of some errors.
- Fixed an issue with Firefox that was resulting in duplicate jobs on job creation.
- Removed the Mathjax external dependency on CDN.
- Improved `PATH` and `JAVA_HOME` handling that previously broke Hadoop CLIs.
- Fixed an issue with Java security policy files that caused Kerberos issues.
- Fixed an issue that caused `git clone` to fail on some repositories.
- Fixed an issue where updating LDAP admin settings deactivated the local fallback login.
- Fixed an issue where bad LDAP configuration crashed the application.
- Fixed an issue where job environmental variable settings did not persist.

Known Issues and Limitations in Cloudera Data Science Workbench 1.0.x

For a list of known issues and limitations, refer the documentation for version 1.0.x at Cloudera Data Science Workbench 1.0.x.

Cloudera Data Science Workbench 1.0.0

Version 1.0 represents the first generally available (GA) release of Cloudera Data Science Workbench. For information about the main features and benefits of Cloudera Data Science Workbench, as well as an architectural overview of the product, see About Cloudera Data Science Workbench on page 10.

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.x

This topic lists the current known issues and limitations in Cloudera Data Science Workbench 1.2.x.

Upgrades

Risk of Data Loss During Cloudera Data Science Workbench (CDSW) Shutdown and Restart

Stopping Cloudera Data Science Workbench involves unmounting the NFS volumes that store CDSW project directories and then cleaning up a folder where the kubelet stores its temporary state. However, due to a race condition, this NFS unmount process can take too long or fail altogether. If this happens, CDSW projects that remain mounted will be deleted by the cleanup step.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench versions -

- 1.0.x
- 1.1.x
- 1.2.x
- 1.3.0, 1.3.1
- 1.4.0, 1.4.1

Users affected: This potentially affects all CDSW users.

Detected by: Nehmé Tohmé (Cloudera)

Severity (Low/Medium/High): High

Impact: If the NFS unmount fails during shutdown, data loss can occur. All CDSW project files might be deleted.

CVE: N/A

Immediate action required: If you are running any of the affected Cloudera Data Science Workbench versions, you must run the following script on the CDSW master node every time before you stop or restart Cloudera Data Science Workbench. Failure to do so can result in data loss.

This script should also be run before initiating a Cloudera Data Science Workbench upgrade. As always, we recommend creating a full backup prior to beginning an upgrade.

cdsw_protect_stop_restart.sh - Available for download at: cdsw_protect_stop_restart.sh.

```
#!/bin/bash

set -e

cat << EXPLANATION


This script is a workaround for Cloudera TSB-346. It protects your
CDSW projects from a rare race condition that can result in data loss.
Run this script before stopping the CDSW service, irrespective of whether
the stop precedes a restart, upgrade, or any other task.

Run this script only on the master node of your CDSW cluster.

You will be asked to specify a target folder on the master node where the
```

```
script will save a backup of all your project files. Make sure the target
folder has enough free space to accommodate all of your project files. To
determine how much space is required, run 'du -hs /var/lib/cdsw/current/projects'
on the CDSW master node.

This script will first back up your project files to the specified target
folder. It will then temporarily move your project files aside to protect
against the data loss condition. At that point, it is safe to stop the CDSW
service. After CDSW has stopped, the script will move the project files back
into place.

Note: This workaround is not required for CDSW 1.4.2 and higher.



EXPLANATION

read -p "Enter target folder for backups: " backup_target

echo "Backing up to $backup_target..."
rsync -azp /var/lib/cdsw/current/projects "$backup_target"

read -n 1 -p "Backup complete. Press enter when you are ready to stop CDSW: "

echo "Deleting all Kubernetes resources..."
kubectl delete
configmaps,deployments,daemonsets,replicasets,services,ingress,secrets,persistentvolumes,persistentvolumeclaims,jobs
 --all
kubectl delete pods --all

echo "Temporarily saving project files to /var/lib/cdsw/current/projects_tmp..."
mkdir /var/lib/cdsw/current/projects_tmp
mv /var/lib/cdsw/current/projects/* /var/lib/cdsw/current/projects_tmp

echo -e "Please stop the CDSW service."

read -n 1 -p "Press enter when CDSW has stopped: "

echo "Moving projects back into place..."
mv /var/lib/cdsw/current/projects_tmp/* /var/lib/cdsw/current/projects
rm -rf /var/lib/cdsw/current/projects_tmp

echo -e "Done. You may now upgrade or start the CDSW service."
echo -e "When CDSW is running, if desired, you may delete the backup data at
$backup_target"
```

Addressed in release/refresh/patch: This issue is fixed in Cloudera Data Science Workbench 1.4.2.

Note that you are required to run the workaround script above when you upgrade from an affected version to a release with the fix. This helps guard against data loss when the affected version needs to be shut down during the upgrade process.

For the latest update on this issue see the corresponding Knowledge article:

TSB 2018-346: Risk of Data Loss During Cloudera Data Science Workbench (CDSW) Shutdown and Restart

In version 1.2.2, the `cdsw status` command fails to perform all the required checks

In Cloudera Data Science Workbench 1.2.2, the `cdsw status` command fails to perform certain required system checks and displays the following error message.

```
Sending detailed logs to [/tmp/cdsw_status_abc.log] ...
CDSW Version: [1.2.2....]
OK: Application running as root check
OK: Sysctl params check
Failed to run CDSW Nodes Check.
Failed to run CDSW system pods check.
Failed to run CDSW application pods check.
Failed to run CDSW services check.
Failed to run CDSW config maps check.
Failed to run CDSW secrets check.
...
```

Affected Versions: Cloudera Data Science Workbench 1.2.2.

Fixed In: Cloudera Data Science Workbench 1.3.0. If you cannot upgrade, use the following workaround.

Workaround: To work around this issue, install the following package on all Cloudera Data Science Workbench master and worker nodes.

```
pip install backports.ssl_match_hostname==3.5.0.1
```

Cloudera Bug: DSE-3070

Note: If TLS/SSL is enabled on your deployment, the `cdsw status` might still fail to display the `Cloudera Data Science Workbench is ready.` message. For more details, see the associated Known Issue.

Environment setup incomplete after upgrade from version 1.2.0 to 1.2.1 on CSD-based deployments

After upgrading from Cloudera Data Science Workbench 1.2.0 to 1.2.1 on a CSD-based deployment, CLI commands might not work as expected due to missing binaries in the environment. Note that this issue does not affect fresh installs.

Fixed In: Cloudera Data Science Workbench 1.2.2. If you cannot upgrade, use the following workaround.

Workaround: Deactivate and reactivate the Cloudera Data Science Workbench parcel in Cloudera Manager. To do so, go to the Cloudera Manager Admin Console. In the top navigation bar, click **Hosts** > **Parcels**.

Locate the current active CDSW parcel and click **Deactivate**. Once deactivation is complete, click **Activate**.

Cloudera Bug: DSE-2928

Output from CLI commands is inconsistent with Cloudera Manager CDSW service configuration

Clusters that have migrated from package-based to a CSD-based deployment might face issues with CLI commands showing output that is inconsistent with their settings in the Cloudera Manager CDSW service.

Affected Versions: Cloudera Data Science Workbench 1.2.0, 1.2.1

Fixed In: Cloudera Data Science Workbench 1.2.2. If you cannot upgrade, use the following workaround.

Workaround: Delete the `/etc/cdsw/config` directory on all the Cloudera Data Science Workbench gateway hosts.

Cloudera Bug: DSE-3021

CDH Integration

Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0.

Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

CDH client configuration changes require a full Cloudera Data Science Workbench reset

Cloudera Data Science Workbench does not automatically detect configuration changes on the CDH cluster. Therefore, any changes made to CDH services, ranging from updates to service configuration properties to complete parcel upgrades, must be followed by a full reset of Cloudera Data Science Workbench.

Affected Versions: Cloudera Data Science Workbench 1.2.x, 1.3.x

Workaround: Run the following commands to ensure client configuration changes are picked up by Cloudera Data Science Workbench.

```
cdsw reset
cdsw init
```

Cloudera Bug: DSE-3350

Cloudera Manager Integration

- **Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0.**

  Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

- **CSD distribution/activation fails on mixed-OS clusters when there are third-party parcels running on OSs that are not supported by Cloudera Data Science Workbench**

  For example, adding a new CDSW gateway host on a RHEL 6 cluster running RHEL-6 compatible parcels will fail. This is because Cloudera Manager will not allow distribution of the RHEL 6 parcels on the new host which will likely be running a CDSW-compatible operating system such as RHEL 7.

  Affected Versions: Cloudera Data Science Workbench 1.2.x, 1.3.x

  Workaround: To ensure adding a new CDSW gateway host is successful, you must create a copy of the 'incompatible' third-party parcel files and give them the corresponding RHEL 7 names so that Cloudera Manager allows them to be distributed on the new gateway host. Use the following sample instructions to do so:

  1. SSH to the Cloudera Manager Server host.
  2. Navigate to the directory that contains all the parcels. By default, this is `/opt/cloudera/parcel-repo`.

```
cd /opt/cloudera/parcel-repo
```

  3. Make a copy of the incompatible third-party parcel with the new name. For example, if you have a RHEL 6 parcel that cannot be distributed on a RHEL 7 CDSW host:

```
cp <PARCELNAME.cdh5.x.x.p0.123>-el6.parcel <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel
```

  4. Repeat the previous step for parcel's SHA file.

```
cp <PARCELNAME.cdh5.x.x.p0.123>-el6.parcel.sha <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
```

  5. Update the new files' owner and permissions to match those of existing parcels in the `/opt/cloudera/parcel-repo` directory.

```
chown cloudera-scm:cloudera-scm <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel

chown cloudera-scm:cloudera-scm <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
chmod 640 <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel
chmod 640 <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
```

  You should now be able to add new gateway hosts for Cloudera Data Science Workbench to your cluster.

  Cloudera Bug: OPSAPS-42130, OPSAPS-31880

- **CDSW Service health status after a restart does not match the actual state of the application**

  After a restart, the Cloudera Data Science Workbench service in Cloudera Manager will display **Good** health even though the Cloudera Data Science Workbench web application might need a few more minutes to get ready to serve requests.

- On a CSD-based deployment, the `cdsw status` CLI command always reports: "Cloudera Data Science Workbench is not ready yet". This occurs even if the Cloudera Data Science Workbench service is up and running as expected. To see the correct status, use the Cloudera Manager UI.

  Fixed In: Cloudera Data Science Workbench 1.2.2.

  Cloudera Bug: DSE-2927

- A file descriptor leak causes "Failed to get Kubernetes client configuration" errors in Cloudera Manager.

  Affected Versions: Cloudera Data Science Workbench 1.2.0, 1.2.1.

  Fixed In: Cloudera Data Science Workbench 1.2.2.

  Cloudera Bug: DSE-2910

CDS 2.x Powered By Apache Spark

With Spark 2.2 Release 1, a PySpark application can run only *once* per active Workbench session

With Spark 2.2 Release 1, you can run a PySpark application only *once* per active Workbench session. Subsequent runs of the application will fail with a `getDelegationToken` error.

Affected Versions: Cloudera Distribution of Apache Spark 2.2 Release 1

Fixed In: Cloudera Distribution of Apache Spark 2.2 Release 2. If you cannot upgrade, use one of the following workarounds.

- Use Cloudera Distribution of Apache Spark 2.1.

  or

- Launch a new PySpark session every time you want to run the application.

Cloudera Bug: CDH-58475

Certain CDS releases require additional configuration to run PySpark on Cloudera Data Science Workbench versions 1.3.x (and lower)

Affected Versions: Due to a security fix in CDS, there is now a mismatch between the versions of `py4j` that ship with the following versions of the two products:

- Cloudera Data Science Workbench 1.3.x (and lower) include `py4j 0.10.4`, and,
- CDS 2.1 release 3, CDS 2.2 release 3, and CDS 2.3 release 3 include `py4j 0.10.7`.

This version mismatch results in PySpark session/job failures on Cloudera Data Science Workbench.

The following error messages are indicative of this issue:

```
TypeError: __init__() got an unexpected keyword argument 'auth_token'
```

```
TypeErrorTraceback (most recent call last)
in engine
----> 1 spark = SparkSession    .builder    .appName("PythonPi")    .getOrCreate()
```

Use one of the following workarounds to continue running PySpark jobs on Cloudera Data Science Workbench 1.3.x. Alternatively, upgrade to Cloudera Data Science Workbench 1.4.x.

- **Workaround 1: Use the `PYTHONPATH` environmental variable to use CDS's version of py4j**

  This process requires site administrator privileges.

  1. Log in to Cloudera Data Science Workbench.

2. Click **Admin** > **Engines**.
3. Under the **Environmental variables** section, add a new variable:
    - **Name:** `PYTHONPATH`
    - **Value:** `$SPARK_HOME/python/lib/py4j-0.10.7-src.zip`
4. Click **Add**.

- **Workaround 2: Install py4j `0.10.7` directly in project sessions**

  If you do not want to make site-wide changes as described in the previous workaround, individual users can install `py4j 0.10.7` directly in their project's sessions to continue running PySpark.

  For example, in a Python 3 session, run the following command in the workbench command prompt:

```
!pip3 install py4j==0.10.7
```

Fixed Versions: Cloudera Data Science Workbench 1.4.x

Version 1.4 ships with engine 5 that includes a CDS-compatible version of `py4j`. Note that the upgrade itself will not automatically upgrade existing projects to engine 5. Once the upgrade to 1.4 is complete, project administrators/collaborators must manually configure their projects to use engine 5 as the base image (go to the project's **Settings** > **Engine** page).

Cloudera Bug: DSE-4046, DSE-4316

Crashes and Hangs

- On a deployment with at least one worker node, running `cdsw stop` or stopping the CDSW service in Cloudera Manager results in the application hanging indefinitely.

  Fixed In: Cloudera Data Science Workbench 1.2.1. If you cannot upgrade to Cloudera Data Science Workbench 1.2.1, use the following workaround.

  Workaround: Stop the master node first, then stop the worker node(s).

  Cloudera Bug: DSE-2880

- High I/O utilization on the application block device can cause the application to stall or become unresponsive. Users should read and write data directly from HDFS rather than staging it in their project directories.

- Installing ipywidgets or a Jupyter notebook into a project can cause Python engines to hang due to an unexpected configuration. The issue can be resolved by deleting the installed libraries from the R engine terminal.

GPU Support

Only CUDA-enabled NVIDIA GPU hardware is supported

Cloudera Data Science Workbench only supports CUDA-enabled NVIDIA GPU cards.

Heterogeneous GPU hardware is not supported

You must use the same GPU hardware across a single Cloudera Data Science Workbench deployment.

GPUs are not detected after a machine reboot

This issue occurs because certain NVIDIA modules do not load automatically after a reboot.

Workaround: To work around this issue, use the following steps to manually load the required modules before Cloudera Data Science Workbench services start. The following commands load the `nvidia.ko` module, create the `/dev/nvidiactl` device, and create the list of devices at `/dev/nvidia0`. They will also create the `/dev/nvidia-uvm` and `/dev/nvidia-uvm-tools` devices, and assign execute privileges to `/etc/rc.modules`. Run these commands once on all the machines that have GPU hardware.

```
# Manually load the required NVIDIA modules
sudo cat >> /etc/rc.modules <<EOMSG
/usr/bin/nvidia-smi
/usr/bin/nvidia-modprobe -u -c=0
EOMSG

# Set execute permission for /etc/rc.modules
sudo chmod +x /etc/rc.modules
```

Cloudera Bug: DSE-2847

NVIDIA driver directory mount is not set correctly

GPUs are not detected by Cloudera Data Science Workbench. This is because the mount for the NVIDIA driver directory is set incorrectly.

Affected Versions: Cloudera Data Science Workbench 1.2.1.

Fixed In: Cloudera Data Science Workbench 1.2.2. If you cannot upgrade, use the following workaround.

Workaround: To work around this, you will need to mount the directory specified by the `NVIDIA_LIBRARY_PATH` in your site administrator settings, and the project's environment.

For example, if `NVIDIA_LIBRARY_PATH` is set to `/var/lib/nvidia-docker/volumes/nvidia_driver/381.22/`, go to **Admin** > **Engines** and add the NVIDIA driver directory path to the **Mounts** section:

```
/var/lib/nvidia-docker/volumes/nvidia_driver/381.22/
```

Then go to your project's **Settings** > **Engine** page and add the `LD_LIBRARY_PATH` environmental variable to your project. Set it to:

```
/var/lib/nvidia-docker/volumes/nvidia_driver/381.22/bin:$LD_LIBRARY_PATH
```

Cloudera Bug: DSE-2957

CUDA engines cannot access GPU libraries in sessions and jobs

The value for the `LD_LIBRARY_PATH` environmental variable is not propagated to CUDA engines. As a result, CUDA engines will not be able to access the libraries required to enable GPU usage in sessions and jobs.

Affected Versions: Cloudera Data Science Workbench 1.2.0.

Fixed In: Cloudera Data Science Workbench 1.2.1. If you cannot upgrade to Cloudera Data Science Workbench 1.2.1, use the following workaround.

Workaround: As a workaround, set `LD_LIBRARY_PATH` in your project's environment by going to the project's **Settings** > **Engine** page:

```
LD_LIBRARY_PATH=/usr/local/nvidia/lib64:/usr/local/cuda/lib64:/usr/local/nvidia/lib:/usr/local/cuda/lib:$LD_LIBRARY_PATH
```

Cloudera Bug: DSE-2828

Networking

- Custom `/etc/hosts` entries on Cloudera Data Science Workbench hosts do not propagate to sessions and jobs running in containers.

Cloudera Bug: DSE-2598

- Initialisation of Cloudera Data Science Workbench (`cdsw init`) will fail if localhost does not resolve to `127.0.0.1`.
- Cloudera Data Science Workbench does not support DNS servers running on `127.0.0.1:53`. This IP address resolves to the container localhost within Cloudera Data Science Workbench containers. As a workaround, use either a non-loopback address or a remote DNS server.
- Due to limits in `libc`, only two DNS servers are supported in `/etc/resolv.conf`. Kubernetes uses one additional entry for the cluster DNS.

Security

SSH access to Cloudera Data Science Workbench nodes must be disabled

The container runtime and application data storage is not fully secure from untrusted users who have SSH access to the gateway nodes. Therefore, SSH access to the gateway nodes for untrusted users should be disabled for security and resource utilization reasons.

TSB-328: Unauthenticated User Enumeration in Cloudera Data Science Workbench

Unauthenticated users can get a list of user accounts of Cloudera Data Science Workbench.

Affected Versions: Cloudera Data Science Workbench 1.4.0 (and lower)

Fixed Versions: Cloudera Data Science Workbench 1.4.2 (and higher)

Immediate action required: Upgrade to the latest version of Cloudera Data Science Workbench (1.4.2 or higher).

For more details, see the Security Bulletins - TSB-328.

Remote Command Execution and Information Disclosure in Cloudera Data Science Workbench

A configuration issue in Kubernetes used by Cloudera Data Science Workbench can allow remote command execution and privilege escalation in CDSW. A separate information permissions issue can cause the LDAP bind password to be exposed to authenticated CDSW users when LDAP bind search is enabled.

Affected Versions: Cloudera Data Science Workbench 1.3.0 (and lower)

Fixed Versions: Cloudera Data Science Workbench 1.3.1 (and higher)

For more details, see the Security Bulletins - TSB-313.

TLS/SSL

- **Issues with 'cdsw status' and Cloudera Manager health checks**

  With TLS/SSL enabled, the `cdsw status` command does not display the `Cloudera Data Science Workbench is ready` message.

  Similarly, Cloudera Manager health checks for the CDSW service will also always report **Bad** health and display the following message: "Web is not yet up."

  Affected Versions and Workarounds:

  - **Cloudera Data Science Workbench 1.2.0**
  - **Cloudera Data Science Workbench 1.2.1, 1.2.2** - This issue occurs on SLES 12 systems when you are using TLS/SSL certificates signed by your organization's internal Certificate Authority (CA). This is due to an incompatibility with the version of Python (v2.7.9) that ships with SLES 12. This issue will also occur on RHEL 7 systems if the system Python version is 2.7.5 (or higher).

  Fixed Versions: Cloudera Data Science Workbench 1.3.0

Workarounds: Use one of the following methods to work around this issue:

- Import your organization's root CA certificate into your machine’s trust store.

  To do so, copy the internal CA certificate in PEM format to the `/etc/pki/ca-trust/source/anchors/` directory. If the certificate is in OpenSSL’s '`BEGIN TRUSTED CERTIFICATE`' format, copy it to the `/etc/pki/ca-trust/source` directory. Then run:

```
sudo update-ca-trust
```

  OR

- Use TLS/SSL certificates signed by a well-known public CA.

Cloudera Bug: DSE-2871, DSE-3090

- Self-signed certificates where the Certificate Authority is not part of the user's trust store are not supported for TLS termination. For more details, see Enabling TLS/SSL - Limitations.
- Cloudera Data Science Workbench does not support the use of encrypted private keys for TLS.

  Cloudera Bug: DSE-1708

LDAP

- LDAP group search filters are currently not supported. To limit access to Cloudera Data Science Workbench to certain groups, use "memberOf" or the equivalent user attribute in LDAP User Filter.

  Fixed Version: Cloudera Data Science Workbench 1.4.0

  Cloudera Bug: DSE-1616

Kerberos

- PowerBroker-equipped Active Directory is not supported.

  Cloudera Bug: DSE-1838
- Using Kerberos plugin modules in `krb5.conf` is not supported.
- Modifying the `default_ccache_name` parameter in `krb5.conf` does not work in Cloudera Data Science Workbench. Only the default path for this parameter, `/tmp/krb5cc_${uid}`, is supported.
- When you upload a Kerberos keytab to authenticate yourself to the CDH cluster, Cloudera Data Science Workbench displays a fleeting error message ('cancelled') in the bottom right corner of the screen, even if authentication was successful. This error message can be ignored.

  Cloudera Bug: DSE-2344
- Cloudera Data Science Workbench does not support the use of a FreeIPA KDC.

  Cloudera Bug: DSE-1482

Jobs API

- Cloudera Data Science Workbench does not support changing your API key, or having multiple API keys.
- Currently, you cannot create a job, stop a job, or get the status of a job using the Jobs API.

Engines

- Packages installed within an extensible engine Dockerfile using `pip` or `pip3` will not be usable by Cloudera Data Science Workbench.

  Fixed In: Cloudera Data Science Workbench 1.2.1.

  Cloudera Bug: DSE-1873

- Cloudera Data Science Workbench only supports custom extended engines that are based on the Cloudera Data Science Workbench base image.

- Autofs mounts are not supported with Cloudera Data Science Workbench.

  Cloudera Bug: DSE-2238

- Cloudera Data Science Workbench does not support pulling images from registries that require Docker credentials.

  Cloudera Bug: DSE-1521

- When using Conda to install Python packages, you must specify the Python version to match the Python versions shipped in the engine image (2.7.11 and 3.6.1). If not specified, the conda-installed Python version will not be used within a project. Pip (pip and pip3) does not face this issue.

Usability

- In a scenario where 100s of users are logged in and creating processes, the `nproc` and `nofile` limits of the system may be reached. Use `ulimits` or other methods to increase the maximum number of processes and open files that can be created by a user on the system.

- When rebooting, Cloudera Data Science Workbench nodes can take a significant amount of time (about 30 minutes) to become ready.

- Long-running operations such as `fork` and `clone` can time out when projects are large or connections outlast the HTTP timeouts of reverse proxies.

- The Scala kernel does not support autocomplete features in the editor.

- Scala and R code can sometimes indent incorrectly in the workbench editor.

  Cloudera Bug: DSE-1218

Cloudera Data Science Workbench 1.2.x Requirements and Supported Platforms

This topic lists the software and hardware configuration required to successfully install and run Cloudera Data Science Workbench. Cloudera Data Science Workbench does not support hosts or clusters that do not conform to the requirements listed on this page.

Cloudera Manager and CDH Requirements

Cloudera Data Science Workbench 1.2.x is supported on the following versions of CDH and Cloudera Manager:

- CDH 5.7 or higher 5.x versions.
- **CSD-based deployments:** Cloudera Manager 5.13 or higher 5.x versions.

  Package-based deployments: Cloudera Manager 5.11 or higher 5.x versions.

  All cluster hosts must be managed by Cloudera Manager. Note that all Cloudera Data Science Workbench administrative tasks require root access to the cluster's gateway hosts where Cloudera Data Science Workbench is installed. Therefore, Cloudera Data Science Workbench does not support single-user mode installations.
- CDS 2.1 Powered by Apache Spark and higher.

Important: Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0. Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

Operating System Requirements

Cloudera Data Science Workbench 1.2.x is supported on the following operating systems:

- RHEL/CentOS 7.2, 7.3, 7.4
- Oracle Linux 7.3 (UEK - default)
- SLES 12 SP2 (supported for Cloudera Data Science Workbench 1.2.2 and higher)

A gateway node that is dedicated to running Cloudera Data Science Workbench must use one of the aforementioned supported versions even if the remaining CDH hosts in your cluster are running any of the other operating systems supported by Cloudera Enterprise.

Cloudera Data Science Workbench publishes placeholder parcels for other operating systems as well. However, note that these do not work and have only been included to support mixed-OS clusters.

Important: The operating system you use must have memory cgroups enabled.

JDK Requirements

The entire CDH cluster, including Cloudera Data Science Workbench gateway nodes, must use Oracle JDK. OpenJDK is not currently supported by Cloudera Data Science Workbench.

For more specifics on the versions of Oracle JDK recommended for CDH and Cloudera Manager clusters, see the Cloudera Product Compatibility Matrix - Supported JDK Versions.

JDK 8 Requirement for Spark 2.2

CSD-based deployments:

Spark 2.2 requires JDK 1.8. On CSD-based deployments, Cloudera Manager automatically detects the path and version of Java installed on Cloudera Data Science Workbench gateway hosts. However, if a host has both JDK 1.7 and JDK 1.8 installed, Cloudera Manager might choose to use JDK 1.7 over JDK 1.8. If you are using Spark 2.2, this will create a problem during the first run of the service because Spark 2.2 will not work with JDK 1.7. To work around this, configure Cloudera Manager to use JDK 1.8 on Cloudera Data Science Workbench gateway hosts. For instructions, see Configuring a Custom Java Home Location in Cloudera Manager.

To upgrade your entire CDH cluster to JDK 1.8, see Upgrading to Oracle JDK 1.8.

Package-based deployments:

Set `JAVA_HOME` to the JDK 8 path in `cdsw.conf` during the installation process. If you need to modify `JAVA_HOME` after the fact, restart the master and worker nodes to have the changes go into effect.

Networking and Security Requirements

- A wildcard subdomain such as `*.cdsw.`*`company`*`.com`. Wildcard subdomains are used to provide isolation for user-generated content.
- Disable all pre-existing `iptables` rules. While Kubernetes makes extensive use of `iptables`, it's difficult to predict how pre-existing iptables rules will interact with the rules inserted by Kubernetes. Therefore, Cloudera recommends you use the following commands to disable all pre-existing rules before you proceed with the installation.

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

- Cloudera Data Science Workbench sets the following `sysctl` options in `/etc/sysctl.d/k8s.conf`:
  - `net.bridge.bridge-nf-call-iptables=1`
  - `net.bridge.bridge-nf-call-ip6tables=1`
  - `net.ipv4.ip_forward=1`

  Underlying components of Cloudera Data Science Workbench (Docker, Kubernetes, and NFS) require these options to work correctly. Make sure they are not overridden by high-priority configuration such as `/etc/sysctl.conf`.
- SELinux must either be disabled or run in *permissive* mode.
- Multi-homed networks are supported only with Cloudera Data Science Workbench 1.2.2 (and higher).
- No firewall restrictions across Cloudera Data Science Workbench or CDH hosts.
- Non-root SSH access is not allowed on Cloudera Data Science Workbench hosts.
- `localhost` must resolve to `127.0.0.1`.
- Cloudera Data Science Workbench does not support DNS servers running on `127.0.0.1:53`. This IP address resolves to the container localhost within Cloudera Data Science Workbench containers. As a workaround, use either a non-loopback address or a remote DNS server.

Cloudera Data Science Workbench does not support hosts or clusters that do not conform to these restrictions.

Recommended Hardware Configuration

Cloudera Data Science Workbench hosts are added to your CDH cluster as gateway hosts. The recommended minimum hardware configuration for the master host is:

- **CPU:** 16+ CPU (vCPU) cores
- **RAM:** 32+ GB RAM
- **Disk**
  - Root Volume: 100+ GB.

    The Cloudera Data Science Workbench installer temporarily decompresses the engine image file located in `/etc/cdsw/images` to the `/var/lib/docker/tmp/` directory. If you are going to partition the root volume, make sure you allocate at least 20 GB to `/var/lib/docker/tmp` so that the installer can proceed without running out of space.
  - Application Block Device or Mount Point (Master Host Only): 500+ GB
  - Docker Image Block Device: 500+ GB

Important: Do not reuse existing CDH gateway hosts for the Cloudera Data Science Workbench. Running other services on the same hosts can lead to port conflicts, unreliable execution of user workloads, and out-of-memory errors.

Scaling Guidelines

New nodes can be added and removed from a Cloudera Data Science Workbench deployment without interrupting any jobs already scheduled on existing hosts. Therefore, it is rather straightforward to increase capacity based on observed usage. At a minimum, Cloudera recommends you allocate at least 1 CPU core and 2 GB of RAM per concurrent session or job. CPU can burst above a 1 CPU core share when spare resources are available. Therefore, a 1 CPU core allocation is often adequate for light workloads. Allocating less than 2 GB of RAM can lead to out-of-memory errors for many applications.

As a general guideline, Cloudera recommends nodes with RAM between 60GB and 256GB, and between 16 and 48 cores. This provides a useful range of options for end users. SSDs are strongly recommended for application data storage.

For some data science and machine learning applications, users can collect a significant amount of data in memory within a single R or Python process, or use a significant amount of CPU resources that cannot be easily distributed into the CDH cluster. If individual users frequently run larger workloads or run workloads in parallel over long durations, increase the total resources accordingly. Understanding your users' concurrent workload requirements or observing actual usage is the best approach to scaling Cloudera Data Science Workbench.

Python Supported Versions

The default Cloudera Data Science Workbench engine (Base Image Version 1) includes **Python 2.7.11** and **Python 3.6.1**. To use PySpark with lambda functions that run within the CDH cluster, the Spark executors must have access to a matching version of Python. For many common operating systems, the default system Python will not match the minor release of Python included in Data Science Workbench.

To ensure that the Python versions match, Python can either be installed on every CDH node or made available per job run using Spark's ability to distribute dependencies. Given the size of a typical isolated Python environment and the desire to avoid repeated uploads from gateway hosts, Cloudera recommends installing Python 2.7 and 3.6 on the cluster if you are using PySpark with lambda functions. You can install Python 2.7 and 3.6 on the cluster using any method and set the corresponding `PYSPARK_PYTHON` variable in your project.

Anaconda - Continuum Analytics and Cloudera have partnered to create an Anaconda parcel for CDH to enable simple distribution, installation, and management of popular Python packages and their dependencies. The public Anaconda parcel ships Python 2.7.11. Note that the Anaconda parcel is not directly supported by Cloudera and no publicly available parcel exists for Python 3.6. For an example on distributing Python dependencies dynamically, see Example: Distributing Dependencies on a PySpark Cluster on page 82.

Docker and Kubernetes Support

Cloudera Data Science Workbench only supports the versions of Docker and Kubernetes that are shipped with each release. Upgrading Docker or Kubernetes, or running on third-party Kubernetes clusters is not supported.

Supported Browsers

- Chrome (latest stable version)
- Firefox (latest released version and latest ESR version)
- Safari 9+
- Internet Explorer (IE) 11+
  - IE's Compatibility View mode is not supported.

Cloudera Altus Director Support (AWS Only)

Starting with Cloudera Data Science Workbench 1.2.x, you can use Cloudera Altus Director to deploy clusters with Cloudera Data Science Workbench.

Altus Director support is available only for the following platforms:

- Cloudera Altus Director 2.6.0 (and higher)
- Cloudera Manager 5.13.1 (and higher)
- CSD-based Cloudera Data Science Workbench 1.2.x (and higher)
- Currently, only installations on Amazon Web Services (AWS) are supported.

Deploying Cloudera Data Science Workbench with Cloudera Altus Director

Points to note when using Cloudera Altus Director to install Cloudera Data Science Workbench:

- (**Required**) Before you run the command to bootstrap a new cluster, set the `lp.normalization.mountAllUnmountedDisksRequired` property to `false` in the Altus Director server's `application.properties` file, and *then restart Altus Director*.
- Use the following sample configuration file to bootstrap a cluster with the Altus Director CLI: aws.cdsw.conf. This will deploy a Cloudera Manager cluster with Cloudera Data Science Workbench on AWS.

  Note that this sample file installs a very limited CDH cluster with just the following services: HDFS, YARN, and Spark 2. You can extend it as needed to match your use case.

Related Topics:

- Using Products outside CDH with Altus Director
- Altus Director CLI
- The Altus Director Configuration File
- Setting Altus Director Properties

Recommended Configuration on Amazon Web Services (AWS)

On AWS, Cloudera Data Science Workbench must be used with persistent/long-running Apache Hadoop clusters only.

CDH and Cloudera Manager Hosts

- For instructions on deploying CDH and Cloudera Manager on AWS, refer the Cloudera Reference Architecture for AWS deployments.

Cloudera Data Science Workbench Hosts

- **Operations**
  - Use Cloudera Director to orchestrate operations. Use Cloudera Manager to monitor the cluster.
- **Networking**
  - No security group or network restrictions between hosts.
  - HTTP connectivity to the corporate network for browser access. Do not use proxies or manual SSH tunnels.
- **Recommended Instance Types**
  - `m4.4xlarge–m4.16xlarge`

    In this case, bigger is better. That is, one m4.16large is better than four m4.4xlarge hosts. AWS pricing scales linearly, and larger instances have more EBS bandwidth.
- **Storage**
  - 100 GB root volume block device (gp2) on all hosts
  - 500 GB Docker block devices (gp2) on all hosts
  - 1 TB Application block device (io1) on master host

Recommended Configuration on Microsoft Azure

CDH and Cloudera Manager Hosts

- For instructions on deploying CDH and Cloudera Manager on Azure, refer the Cloudera Reference Architecture for Azure deployments.

Cloudera Data Science Workbench Hosts

- **Operations**
  - Use Cloudera Director to orchestrate operations. Use Cloudera Manager to monitor the cluster.
- **Networking**
  - No security group or network restrictions between hosts.
  - HTTP connectivity to the corporate network for browser access. Do not use proxies or manual SSH tunnels.
- **Recommended Instance Types**
  - DS13-DS14 v2 instances on all hosts.
- **Storage**
  - P30 premium storage for the Application and Docker block devices.

Installing and Upgrading Cloudera Data Science Workbench 1.2.x

This topic walks you through the installation and upgrade paths available for Cloudera Data Science Workbench 1.2.x. It also describes the steps needed to configure your cluster gateway hosts and block devices before you can begin installing the Cloudera Data Science Workbench parcel/package.

Installing Cloudera Data Science Workbench 1.2.x

You can use one of the following ways to install Cloudera Data Science Workbench 1.2.x:

- **Using a Custom Service Descriptor (CSD) and Parcel** - Starting with version 1.2.x, Cloudera Data Science Workbench is available as an add-on service for Cloudera Manager 5.13.x. Two files are required for this type of installation: a CSD JAR file that contains all the configuration needed to describe and manage the new Cloudera Data Science Workbench service, and the Cloudera Data Science Workbench parcel. To install this service, first download and copy the CSD file to the Cloudera Manager Server host. Then use Cloudera Manager to distribute the Cloudera Data Science Workbench parcel to the relevant gateway nodes.

  or

- **Using a Package** - Alternatively, you can install the Cloudera Data Science Workbench package directly on the CDH cluster's gateway nodes. In this case, the Cloudera Data Science Workbench service will not be available in Cloudera Manager.

To begin the installation process, continue reading Pre-Installation on page 35.

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.2.x

Depending on your deployment, choose from one of the following upgrade paths:

- Migrating from a package-based deployment (version 1.1.x and lower) to a CSD and parcel-based deployment in Cloudera Data Science Workbench 1.2.x. For instructions, see Upgrading to the Latest Version of Cloudera Data Science Workbench 1.2.x Using Cloudera Manager on page 39.
- Upgrading an existing deployment to a package-based Cloudera Data Science Workbench 1.2.x deployment. For instructions, see Upgrading to the Latest Version of Cloudera Data Science Workbench 1.2.x Using Packages on page 43.

Airgapped Installations

Sometimes organizations choose to restrict parts of their network from the Internet for security reasons. Isolating segments of a network can provide assurance that valuable data is not being compromised by individuals out of maliciousness or for personal gain. However, in such cases isolated hosts are unable to access Cloudera repositories for new installations or upgrades. Effective version 1.1.1, Cloudera Data Science Workbench supports installation on CDH clusters that are not connected to the Internet.

For CSD-based installs in an airgapped environment, put the Cloudera Data Science Workbench parcel into a new hosted or local parcel repository, and then configure the Cloudera Manager Server to target this newly-created repository.

Pre-Installation

The rest of this topic describes the steps you should take to review your platforms and configure your hosts before you begin to install Cloudera Data Science Workbench.

Review Requirements and Supported Platforms

Review the complete list of Cloudera Data Science Workbench 1.2.x Requirements and Supported Platforms on page 30 before you proceed with the installation.

Important: Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0. Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

Set Up a Wildcard DNS Subdomain

Cloudera Data Science Workbench uses subdomains to provide isolation for user-generated HTML and JavaScript, and routing requests between services.. To access Cloudera Data Science Workbench, you must configure the wildcard DNS name `*.cdsw.<company>.com` for the master host as an A record, along with a root entry for `cdsw.<company>.com`.

For example, if your master IP address is `172.46.47.48`, configure two A records as follows:

```
cdsw.<company>.com.    IN A 172.46.47.48
*.cdsw.<company>.com.    IN A 172.46.47.48
```

You can also use a wildcard CNAME record if it is supported by your DNS provider.

Disable Untrusted SSH Access

Cloudera Data Science Workbench assumes that users only access the gateway hosts through the web application. Untrusted users with SSH access to a Cloudera Data Science Workbench host can gain full access to the cluster, including access to other users' workloads. Therefore, untrusted (non-sudo) SSH access to Cloudera Data Science Workbench hosts must be disabled to ensure a secure deployment.

For more information on the security capabilities of Cloudera Data Science Workbench, see the Cloudera Data Science Workbench Security Guide on page 104.

Configure Block Devices

Docker Block Device

The Cloudera Data Science Workbench installer will ***format*** and mount Docker on each gateway host. Make sure there is no important data stored on these devices. ***Do not mount these block devices prior to installation.***

Every Cloudera Data Science Workbench gateway host must have one or more block devices with at least 500 GB dedicated to storage of Docker images. The Docker block devices store the Cloudera Data Science Workbench Docker images including the Python, R, and Scala engines. Each engine image can weigh 15GB.

Application Block Device or Mount Point

The master host on Cloudera Data Science Workbench requires at least 500 GB for database and project storage. This recommended capacity is contingent on the expected number of users and projects on the cluster. While large data files should be stored on HDFS, it is not uncommon to find gigabytes of data or libraries in individual projects. Running out of storage will cause the application to fail. Cloudera recommends allocating at least 5 GB per project and at least 1 TB of storage in total. Make sure you continue to carefully monitor disk space usage and I/O using Cloudera Manager.

Cloudera Data Science Workbench will store all application data at `/var/lib/cdsw`. In a CSD-based deployment, this location is not configurable. Cloudera Data Science Workbench will assume the system administrator has formatted and mounted one or more block devices to `/var/lib/cdsw`.

Regardless of the application data storage configuration you choose, `/var/lib/cdsw` must be stored on a separate block device. Given typical database and user access patterns, an SSD is *strongly* recommended.

By default, data in `/var/lib/cdsw` is not backed up or replicated to HDFS or other nodes. Reliable storage and backup strategy is critical for production installations. See Backup and Disaster Recovery for Cloudera Data Science Workbench on page 100 for more information.

Install Cloudera Data Science Workbench

To use the Cloudera Manager CSD and parcel to install Cloudera Data Science Workbench, follow the steps at Installation and Upgrade Using Cloudera Manager.

OR

To install the Cloudera Data Science Workbench package on the cluster gateway hosts, follow the steps at Installation and Upgrade Using Packages.

Installing and Upgrading Cloudera Data Science Workbench 1.2.x Using Cloudera Manager

This topic describes how to install and upgrade Cloudera Data Science Workbench using Cloudera Manager.

Installing Cloudera Data Science Workbench 1.2.x Using Cloudera Manager

Use the following steps to install Cloudera Data Science Workbench using Cloudera Manager.

Prerequisites

Before you begin installing Cloudera Data Science Workbench, make sure you have completed the steps to secure your hosts, set up DNS subdomains, and configure block devices.

Install Cloudera Distribution of Apache Spark 2

If you have not already done so, install and configure the Cloudera Distribution of Apache Spark 2 parcel and CSD. For instructions, see Installing Cloudera Distribution of Apache Spark 2.

To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

Configure JAVA_HOME (Required for Spark 2.2)

Spark 2.2 requires JDK 1.8. On CSD-based deployments, Cloudera Manager automatically detects the path and version of Java installed on Cloudera Data Science Workbench gateway hosts. However, if a host has both JDK 1.7 and JDK 1.8 installed, Cloudera Manager might choose to use JDK 1.7 over JDK 1.8. If you are using Spark 2.2, this will create a problem during the first run of the service because Spark 2.2 will not work with JDK 1.7. To work around this, configure Cloudera Manager to use JDK 1.8 on Cloudera Data Science Workbench gateway hosts. For instructions, see Configuring a Custom Java Home Location in Cloudera Manager.

To upgrade your entire CDH cluster to JDK 1.8, see Upgrading to Oracle JDK 1.8.

Download and Install the Cloudera Data Science Workbench CSD

CDSW 1.2.x is no longer available for download. Refer to the CDSW documentation for information on suppported versions.

Install the Cloudera Data Science Workbench Parcel

CDSW 1.2.x is no longer available for download. Refer to the CDSW documentation for information on suppported versions.

Add the Cloudera Data Science Workbench Service

To add the Cloudera Data Science Workbench service to your cluster:

1. Log into the Cloudera Manager Admin Console.
2. On the **Home** > **Status** tab, click

   to the right of the cluster name and select **Add a Service** to launch the wizard. A list of services will be displayed.
3. Select the Cloudera Data Science Workbench service and click **Continue**.
4. Assign the Master and Worker roles to the gateway hosts. You must assign the Cloudera Data Science Workbench Master role to one gateway host , and optionally, assign the Worker role to one or more gateway hosts.

   > **Important:** Do not assign the Master and Worker roles to the same host. The Master can automatically perform the functions of a Worker host as needed.

   There are two more role groups that fall under the Cloudera Data Science Workbench service: the Docker Daemon role, and the Application role.

   - The Docker Daemon role must be assigned to *every* Cloudera Data Science Workbench gateway host. On First Run, Cloudera Manager will assign this role to each Cloudera Data Science Workbench gateway host. However, if any more hosts are added or reassigned to Cloudera Data Science Workbench, you must explicitly assign the Docker Daemon role to them.
   - On First Run, Cloudera Manager will assign the Application role to the host running the Cloudera Data Science Workbench Master role. The Application role is always assigned to the same host as the Master. Consequently, this role must never be assigned to a Worker host.
5. Configure the following parameters and click **Continue**.

Properties	Description
Wildcard DNS Domain	Wildcard DNS domain configured to point to the master node. If the previously configured wildcard DNS entries are `cdsw.<company>.com` and `*.cdsw.<company>.com`, then this parameter should be set to `cdsw.<company>.com`. Users' browsers will then be able to contact the Cloudera Data Science Workbench web application at `http://cdsw.<company>.com`. This domain for DNS only, and is unrelated to Kerberos or LDAP domains.
Master Node IPv4 Address	IPv4 address for the master node that is reachable from the worker nodes. *By default, this field is left blank and Cloudera Manager uses the IPv4 address of the Master node.* Within an AWS VPC, set this parameter to the internal IP address of the master node; for instance, if your hostname is `ip-10-251-50-12.ec2.internal`, set `MASTER_IP` to the corresponding IP address, `10.251.50.12`.

Install Required Packages	When this parameter is enabled, the **Prepare Node** command will install all the required package dependencies on First Run. If you choose to disable this property, you must manually install the following packages on *all* Cloudera Data Science Workbench gateway hosts.
	
	`nfs-utils`
	`libseccomp`
	`lvm2`
	`bridge-utils`
	`libtool-ltdl`
	`iptables`
	`rsync`
	`policycoreutils-python`

Properties	Description

	`selinux-policy-base`
	`selinux-policy-targeted`
	`ntp`
	`ebtables`
	`bind-utils`
	`nmap-ncat`
	`openssl`
	`e2fsprogs`
	`redhat-lsb-core`

Docker Block Device	Block device(s) for Docker images. Use the full path to specify the image(s), for instance, `/dev/xvde`. The Cloudera Data Science Workbench installer will format and mount Docker on each gateway host that is assigned the Docker Daemon role. *Do not mount these block devices prior to installation.*

6. The wizard will now begin a First Run of the Cloudera Data Science Workbench service. This includes deploying client configuration for HDFS, YARN and Spark 2, installing the package dependencies on all hosts, and formatting the Docker block device. The wizard will also assign the Application role to the host running Master, and the Docker Daemon role to *all* the Cloudera Data Science Workbench gateway hosts.
7. Once the First Run command has completed successfully, click **Finish** to go back to the Cloudera Manager home page.

Create the Administrator Account

After your installation is complete, set up the initial administrator account. Go to the Cloudera Data Science Workbench web application at `http://cdsw.<your_domain>.com`.

Note: You must access Cloudera Data Science Workbench from the **Wildcard DNS Domain** configured when setting up the service, and not the hostname of the master node. Visiting the hostname of the master node will result in a 404 error.

The first account that you create becomes the site administrator. You may now use this account to create a new project and start using the workbench to run data science workloads. For a brief example, see Getting Started with the Cloudera Data Science Workbench.

Next Steps

As a site administrator, you can invite new users, monitor resource utilization, secure the deployment, and upload a license key for the product. For more details on these tasks, see the Administration and Security guides.

You can also start using the product by configuring your personal account and creating a new project. For a quickstart that walks you through creating and running a simple template project, see Getting Started with Cloudera Data Science Workbench on page 45. For more details on collaborating with teams, working on projects, and sharing results, see the Cloudera Data Science Workbench User Guide on page 48.

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.2.x Using Cloudera Manager

Important: Before upgrading Cloudera Data Science Workbench, you must read the Cloudera Data Science Workbench Release Notes on page 14 relevant to the minor version you are upgrading to.

1. **(Strongly Recommended)** Safely stop Cloudera Data Science Workbench. To avoid running into the data loss issue described in TSB-346, run the cdsw_protect_stop_restart.sh script on the *master* node and follow the sequence of steps as instructed by the script.

The script will first back up your project files to the specified target folder. It will then temporarily move your project files aside to protect against the data loss condition. At that point, it is safe to stop Cloudera Data Science Workbench. To stop Cloudera Data Science Workbench, run the following command on *all* Cloudera Data science Workbench nodes (master and workers):

```
cdsw reset
```

After Cloudera Data Science Workbench has stopped, press enter to continue running the script as instructed. It will then move your project files back into place.

2. **(Strongly Recommended)** On the master node, backup all your application data that is stored in the `/var/lib/cdsw` directory, and the configuration file at `/etc/cdsw/config/cdsw.conf`.
3. Deactivate the existing Cloudera Data Science Workbench parcel. Go to the Cloudera Manager Admin Console. In the top navigation bar, click **Hosts** > **Parcels**.

   Locate the current active CDSW parcel and click **Deactivate**.

4. Install the latest version of Cloudera Data Science Workbench using the CSD and parcel. Note that when you are configuring role assignments for the Cloudera Data Science Workbench service, ***the Master role must be assigned to the same node that was running as master prior to the upgrade***.

   Important:

   Additional Upgrade Notes for version 1.2.x

   - **Proxy Configuration:** If you are using a proxy, the IP addresses for the web and Livelog services (`100.77.0.129,100.77.0.130`) must be appended to the `NO_PROXY` parameter. They have been included in the installation steps.

   For installation instructions, see Installing Cloudera Data Science Workbench 1.2.x Using Cloudera Manager on page 37. You might be able to skip the first few steps assuming you have the wildcard DNS domain and block devices already set up.

5. Use your copy of the backup `cdsw.conf` created in Step 3 to recreate those settings in Cloudera Manager by configuring the corresponding properties under the Cloudera Data Science Workbench service.

   a. Log into the Cloudera Manager Admin Console.
   b. Go to the Cloudera Data Science Workbench service.
   c. Click the **Configuration** tab.
   d. The following table lists all the `cdsw.conf` properties and their corresponding Cloudera Manager properties (in bold). Use the search box to bring up the properties you want to modify.
   e. Click **Save Changes**.

cdsw.conf Property	Corresponding Cloudera Manager Property and Description
TLS_ENABLE	**Enable TLS:** Enable and enforce HTTPS (TLS/SSL) access to the web application (optional). Both internal and external termination are supported. To enable internal termination, you must also set the TLS Certificate for Internal Termination and TLS Key for Internal Termination parameters. If these parameters are not set, terminate TLS using an external proxy. For more details on TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 104.
TLS_CERT TLS_KEY	**TLS Certificate for Internal Termination**, **TLS Key for Internal Termination** Complete path to the certificate and private key (in PEM format) to be used for internal TLS termination. Set these parameters only if you are not terminating TLS externally. You must also set the Enable TLS property to enable and enforce termination. The certificate must include both `DOMAIN` and `*.DOMAIN` as hostnames.

cdsw.conf Property	Corresponding Cloudera Manager Property and Description
	Self-signed certificates are not supported unless trusted fully by clients. Accepting an invalid certificate manually can cause connection failures for unknown subdomains.Set these only if you are not terminating TLS externally. For details on certificate requirements and enabling TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 104.

`HTTP_PROXY`
 	
`HTTPS_PROXY`
	HTTP Proxy, **HTTPS Proxy**
	
	If your deployment is behind an HTTP or HTTPS proxy, set the respective **HTTP Proxy** or **HTTPS Proxy** property to the hostname of the proxy you are using.
	
	`http://<proxy_host>:<proxy-port>`
	`or`
	`https://<proxy_host>:<proxy_port>`
	
	If you are using an intermediate proxy such as Cntlm to handle NTLM authentication, add the Cntlm proxy address to the **HTTP Proxy** *or* **HTTPS Proxy** fields. That is, either `http://localhost:3128` or `https://localhost:3128` respectively.
	
	If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.
	
	To enable trust, copy the proxy's root certificate to the trusted CA certificate store (`ca-trust`) on the Cloudera Data Science Workbench host.
	
	`cp /tmp/<proxy-root-certificate>.crt /etc/pki/ca-trust/source/`
	`anchors/`
	
	Use the following command to rebuild the trusted certificate store.
	
	`update-ca-trust extract`

`ALL_PROXY`	**SOCKS Proxy:** If a SOCKS proxy is in use, set this parameter to `socks5://<host>:<port>/`.

`NO_PROXY`	**No Proxy:** Comma-separated list of hostnames that should be skipped from the proxy.
	
	These include `127.0.0.1`, `localhost`, the value of `MASTER_IP`, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines.
	
	At a minimum, Cloudera *requires* the following `NO_PROXY` configuration.
	
	`127.0.0.1,localhost,<MASTER_IP>,100.66.0.1,100.66.0.2,`
	`100.66.0.3,100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,`
	`100.66.0.9,100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,`
	`100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,100.66.0.20,`
	`100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,100.66.0.25,100.66.0.26,`
	`100.66.0.27,100.66.0.28,100.66.0.29,100.66.0.30,100.66.0.31,100.66.0.32,`
	`100.66.0.33,100.66.0.34,100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,`
	`100.66.0.39,100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,`
	`100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,100.66.0.50,`
	`100.77.0.129,100.77.0.130`

`NVIDIA_GPU_ENABLE`	**Enable GPU Support:** When this property is enabled, and the **NVIDIA GPU Driver Library Path** parameter is set, the GPUs installed on Cloudera Data Science Workbench nodes will be available for use in its workloads. By default, this parameter is disabled.

cdsw.conf Property	Corresponding Cloudera Manager Property and Description
	For instructions on how to enable GPU-based workloads on Cloudera Data Science Workbench, see Using GPUs for Cloudera Data Science Workbench Workloads on page 91.
NVIDIA_LIBRARY_PATH	**NVIDIA GPU Driver Library Path:** Complete path to the NVIDIA driver libraries. For instructions on how to create this directory, see Enable Docker NVIDIA Volumes on GPU Nodes on page 93.

6. Cloudera Manager will prompt you to restart the service if needed.
7. If the release you have just upgraded to includes a new version of the base engine image (see release notes), you need to manually configure existing projects to use the new engine. Cloudera recommends you do so to take advantage of any new features and bug fixes included in the newly released engine.

   To upgrade a project to the new engine, go to the project's **Settings** > **Engine** page and select the new engine from the dropdown. If any of your projects are using custom extended engines, you will need to modify them to use the new base engine image.

Installing and Upgrading Cloudera Data Science Workbench 1.2.x Using Packages

This topic describes how to install and upgrade the Cloudera Data Science Workbench package on a CDH cluster managed by Cloudera Manager.

Installing Cloudera Data Science Workbench 1.2.x from Packages

Use the following steps to install Cloudera Data Science Workbench using RPM packages.

Prerequisites

Before you begin installing Cloudera Data Science Workbench, make sure you have completed the steps to configure your hosts and block devices.

Configure Gateway Hosts Using Cloudera Manager

Cloudera Data Science Workbench hosts must be added to your CDH cluster as gateway hosts, with gateway roles properly configured. To configure gateway hosts:

1. If you have not already done so and plan to use PySpark, install either the Anaconda parcel or Python (versions 2.7.11 and 3.6.1) on your CDH cluster. For more information see, Python Supported Versions on page 32.

2. To support workloads running on Cloudera Distribution of Apache Spark 2, you must configure the Spark 2 parcel and the Spark 2 CSD. For instructions, see Installing Cloudera Distribution of Apache Spark 2.

   To be able to use Spark 2, each user must have their own /home directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

3. Use Cloudera Manager to create add gateway hosts to your CDH cluster.

   1. Create a new host template that includes gateway roles for HDFS, YARN, and Spark 2.
   2. Use the instructions at Adding a Host to the Cluster to add gateway hosts to the cluster. Apply the template created in the previous step to these gateway hosts. If your cluster is kerberized, confirm that the krb5.conf file on your gateway hosts is correct.

4. Test Spark 2 integration on the gateway hosts.

1. SSH to a gateway host.
2. If your cluster is kerberized, run `kinit` to authenticate to the CDH cluster's Kerberos Key Distribution Center. The Kerberos ticket you create is not visible to Cloudera Data Science Workbench users.
3. Submit a test job to Spark 2 by executing the following command:

```
spark2-submit --class org.apache.spark.examples.SparkPi
--master yarn --deploy-mode client
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-example*.jar 100
```

Install Cloudera Data Science Workbench on the Master Node

CDSW 1.2.x is no longer available for installation. Refer to the CDSW documentation for information on suppported versions.

(Optional) Install Cloudera Data Science Workbench on Worker Nodes

CDSW 1.2.x is no longer available for installation.

Create the Administrator Account

Installation typically takes 30 minutes, although it might take an additional 60 minutes for the R, Python, and Scala engine to be available on all hosts.

After your installation is complete, set up the initial administrator account. Go to the Cloudera Data Science Workbench web application at `http://cdsw.<company>.com`.

Note: You must access Cloudera Data Science Workbench from the `DOMAIN` configured in `cdsw.conf`, and not the hostname of the master node. Visiting the hostname of the master node will result in a 404 error.

The first account that you create becomes the site administrator. You may now use this account to create a new project and start using the workbench to run data science workloads. For a brief example, see Getting Started with the Cloudera Data Science Workbench.

Next Steps

As a site administrator, you can invite new users, monitor resource utilization, secure the deployment, and upload a license key for the product. For more details on these tasks, see the Administration and Security guides.

You can also start using the product by configuring your personal account and creating a new project. For a quickstart that walks you through creating a simple template project, see Getting Started with Cloudera Data Science Workbench on page 45. For more details on collaborating with teams, working on projects, and sharing results, see the Cloudera Data Science Workbench User Guide on page 48.

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.2.x Using Packages

Important: Before upgrading Cloudera Data Science Workbench, read the Cloudera Data Science Workbench Release Notes on page 14 relevant to the version you are upgrading to.

1. **(Strongly Recommended)** Safely stop Cloudera Data Science Workbench. To avoid running into the data loss issue described in TSB-346, run the cdsw_protect_stop_restart.sh script on the *master* node and follow the sequence of steps as instructed by the script.

   The script will first back up your project files to the specified target folder. It will then temporarily move your project files aside to protect against the data loss condition. At that point, it is safe to stop Cloudera Data Science

Workbench. To stop Cloudera Data Science Workbench, run the following command on *all* Cloudera Data science Workbench nodes (master and workers):

```
cdsw reset
```

After Cloudera Data Science Workbench has stopped, press enter to continue running the script as instructed. It will then move your project files back into place.

2. **(Strongly Recommended)** On the master node, backup the contents of the `/var/lib/cdsw` directory. This is the directory that stores all your application data.
3. Uninstall the previous release of Cloudera Data Science Workbench. Perform this step on the master node, as well as all the worker nodes.

```
yum remove cloudera-data-science-workbench
```

4. Install the latest version of Cloudera Data Science Workbench on the master node and on all the worker nodes. During the installation process, you might need to resolve certain incompatibilities in `cdsw.conf`. Even though you will be installing the latest RPM, your previous configuration settings in `cdsw.conf` will remain unchanged. Depending on the release you are upgrading from, you will need to modify `cdsw.conf` to ensure it passes the validation checks run by the 1.2.x release.

Important:

Additional Upgrade Notes for version 1.2.x

- **Proxy Configuration** - If you are using a proxy, the IP addresses for the web and Livelog services (`100.77.0.129,100.77.0.130`) must be appended to the `NO_PROXY` parameter. They have been included in the installation instructions.
- **cdsw.conf Parameters**
  - `JAVA_HOME` is a required parameter. Make sure you add `JAVA_HOME` to `cdsw.conf` before you start Cloudera Data Science Workbench.
  - Previous versions allowed `MASTER_IP` to be set to a DNS hostname. If you are still using a DNS hostname, switch to an IP address.

To install the latest version of Cloudera Data Science Workbench, follow the same process to install the package as you would for a fresh installation.

a. Install Cloudera Data Science Workbench on the Master Node on page 43
b. (Optional) Install Cloudera Data Science Workbench on Worker Nodes on page 43.

5. If the release you have just upgraded to includes a new version of the base engine image (see release notes), you need to manually configure existing projects to use the new engine. Cloudera recommends you do so to take advantage of any new features and bug fixes included in the newly released engine.

To upgrade a project to the new engine, go to the project's **Settings** > **Engine** page and select the new engine from the dropdown. If any of your projects are using custom extended engines, you will need to modify them to use the new base engine image.

Getting Started with Cloudera Data Science Workbench

Important: This topic provides a suggested method for quickly getting started with running workloads on the Cloudera Data Science Workbench. For detailed instructions on using the Cloudera Data Science Workbench, see the Cloudera Data Science Workbench User Guide on page 48.

Signing up

Sign up by opening Cloudera Data Science Workbench console in a web browser. The first time you log in, you are prompted to create a username and password.

If your site administrator has configured your cluster to require invitations, you will need an invitation link to sign up.

(On Secure Clusters) Apache Hadoop Authentication with Kerberos

Cloudera Data Science Workbench users can authenticate themselves using Kerberos against the cluster KDC defined in the host's `/etc/krb5.conf` file. Cloudera Data Science Workbench does not assume that your Kerberos principal is always the same as your login information. Therefore, you will need to make sure Cloudera Data Science Workbench knows your Kerberos identity when you sign in.

Important:

- If the `/etc/krb5.conf` file is not available on all Cloudera Data Science Workbench nodes, authentication will fail.
- If you do not see the **Hadoop Authentication** tab, make sure you are accessing your personal account's settings from the top right menu. If you have selected a team account, the tab will not be visible when accessing the Team Settings from the left sidebar.

Authenticate against your cluster’s Kerberos KDC by going to the top-right dropdown menu and clicking **Account settings** > **Hadoop Authentication**. Once successfully authenticated, Cloudera Data Science Workbench uses your stored keytab to ensure that you are secure when running your workloads.

After you authenticate with Kerberos, Cloudera Data Science Workbench will store your keytab. This keytab is then injected into any running engines so that users are automatically authenticated against the CDH cluster when using an engine. Type `klist` at the engine terminal, to see your Kerberos principal. You should now be able to connect to Spark, Hive, and Impala without manually running `kinit`.

If your cluster is not kerberized, your Hadoop username is set to your login username. To override this username you can set an alternative HADOOP_USER_NAME by going to **Account settings** > **Hadoop Authentication**.

Create a Project from a Template

Cloudera Data Science Workbench is organized around projects. Projects hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

To get oriented in Cloudera Data Science Workbench, start by creating a template project in your programming language of choice. Using a template project is not required, and does not limit you to a particular language, but does contain example files to help you get started.

To create a Cloudera Data Science Workbench template project:

1. Sign in to Cloudera Data Science Workbench.
2. On the **Project Lists** page, click **New Project**.
3. Enter a **Project Name**.
4. In the **Template** tab, choose a programming language from the pop-up menu.
5. Click **Create Project**.

After creating your project, you see your project files and the list of jobs defined in your project. These project files are stored on an internal NFS server, and are available to all your project sessions and jobs, regardless of the gateway nodes they run on. Any changes you make to the code or libraries you install into your project will be immediately available when running an engine.

Start Using the Workbench

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala. It currently supports R, Python, and Scala engines. You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

The workbench includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.

Typically, you would use the following steps to run a project in the workbench:

Launch a Session

To launch a session:

1. Navigate to a project's **Overview** page.
2. Click **Open Workbench**.
3. Use **Select Engine Kernel** to choose your language.
4. Use **Select Engine Profile** to select the number of CPU cores and memory.
5. Click **Launch Session**.

The command prompt at the bottom right of your browser window turns green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Execute Code

You can enter and execute code at the command prompt or the editor. The editor is best for code you want to keep, while the command prompt is best for quick interactive exploration.

If you want to enter more than one line of code at the command prompt, use **Shift-Enter** to move to the next line. Press **Enter** to run your code. The output of your code, including plots, appears in the console.

If you created your project from a template, there are code files in the editor. You can open a file in the editor by double-clicking the file name in the file list.

To run code in the editor:

1. Select a code file in the list on the left.
2. Highlight the code you want to run.
3. Press **Ctrl-Enter** (Windows/Linux) or **Command-Enter** (OSX).

When doing real analysis, writing and executing your code from the editor rather than the command prompt makes it easy to iteratively develop your code and save it along the way.

If you require more space for your editor, you can collapse the file list by double-clicking between the file list pane and the editor pane. You can hide the editor using editor's **View** menu.

Access the Terminal

Cloudera Data Science Workbench provides full terminal access to running engines from the web console. If you run `klist` you should see your authenticated Kerberos principal. If you run `hdfs dfs -ls` you will see the files stored in your HDFS home directory. You do not need to worry about Kerberos authentication.

Use the terminal to move files around, run Git commands, access the YARN and Hadoop CLIs, or install libraries that cannot be installed directly from the engine. You can access the Terminal from a running Session page by clicking **Terminal Access** above the session log pane.

All of your project files are in `/home/cdsw`. Any modifications you make to this folder will persist across runs, while modifications to other folders are discarded.

By default, the terminal does not provide root or sudo access to the container. To install packages that require root access, see Customizing Engine Images.

Stop a Session

When you are done with the session, click **Stop** in the menu bar above the console, or use code to exit by typing the following command:

R

```
quit()
```

Python

```
exit
```

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Next Steps

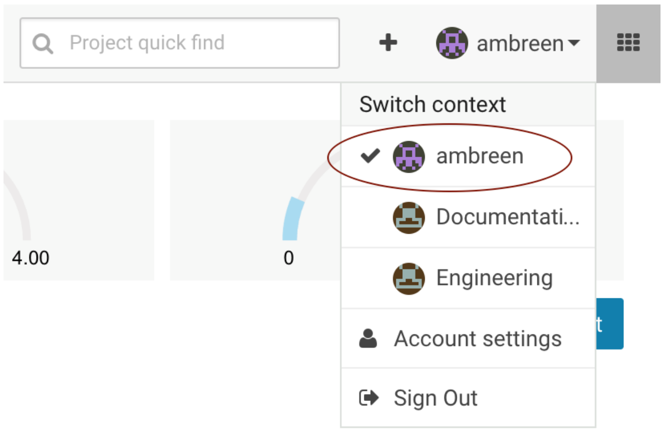
Now that you have successfully run a sample workload with the Cloudera Data Science Workbench, further acquaint yourself with Cloudera Data Science Workbench by reading the User, Administration, and Security guides to learn more about the types of users, how to collaborate on projects, how to use Spark 2 for advanced analytics, and how to secure your deployment.

- Cloudera Data Science Workbench User Guide on page 48
- Cloudera Data Science Workbench Administration Guide on page 88
- Cloudera Data Science Workbench Security Guide on page 104
- Using Cloudera Distribution of Apache Spark 2 with Cloudera Data Science Workbench on page 77

Cloudera Data Science Workbench User Guide

As a Cloudera Data Science Workbench user, you can create and run data science workloads, either individually or in teams. Cloudera Data Science Workbench uses the notion of contexts to separate your personal account from any team accounts you belong to. Depending on the context you are in, you will be able to modify settings for either your personal account, or a team account, and see the projects created in each account. Shared personal projects will show up in your personal account context. Context changes in the UI are subtle, so if you're wondering where a project or setting lives, first make sure you are in the right context.

The application header will tell you which context you are currently in. You can switch to a different context by going to the drop-down menu in the upper right-hand corner of the page.



The rest of this topic features instructions for some common tasks a Cloudera Data Science Workbench user can be expected to perform.

Managing your Personal Account

To manage your personal account settings:

1. Sign in to Cloudera Data Science Workbench.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click **Settings**.

   Profile

   You can modify your name, email, and bio on this page.

   Teams

   This page lists the teams you are a part of and the role assigned to you for each team.

   SSH Keys

   Your public SSH key resides here. SSH keys provide a useful way to access to external resources such as databases or remote Git repositories. For instructions, see SSH Keys on page 113.

   Hadoop Authentication

   Enter your Hadoop credentials here to authenticate yourself against the cluster KDC. For more information, see Hadoop Authentication with Kerberos for Cloudera Data Science Workbench on page 108.

Important: You can also access your personal account settings by clicking **Account settings** in the upper right-hand corner drop-down menu. This option will always take you to your personal settings page, irrespective of the context you are currently in.

Managing Team Accounts

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams allow streamlined administration of projects. Team projects are owned by the team, rather than an individual user. Team administrators can add or remove members at any time, assigning each member different permissions.

Creating a Team

To create a team:

1. Click the plus sign (+) in the title bar, to the right of the **Search** field.
2. Select **Create Team**.
3. Enter a **Team Name**.
4. Click **Create Team**.
5. Add or invite team members. Team members can have one of the following privilege levels:
   - **Viewer** - Cannot create new projects within the team but can be added to existing ones
   - **Contributor** - Can create new projects within the team. They can also be added to existing team projects.
   - **Admin** - Has complete access to all team projects, and account and billing information.
6. Click **Done**.

Modifying Team Account Settings

Team administrators can modify account information, add or invite new team members, and view/edit privileges of existing members. To make these changes:

1. From the upper right drop-down menu, switch context to the team account.
2. Click **Settings** to open up the Account Settings dashboard.

   Profile

   Modify the team description on this page.

   Members

   You can add new team members on this page, and modify privilege levels for existing members.

   SSH Keys

   The team's public SSH key resides here. Team SSH keys provide a useful way to give an entire team access to external resources such as databases. For instructions, see SSH Keys on page 113. Generally, team SSH keys should not be used to authenticate against Git repositories. Use your personal key instead.

Next Steps

Once you have set up your personal or team account, the next steps are:

1. Create a project in Cloudera Data Science Workbench. You can either create a new blank project or import an existing project. For instructions, see Managing Projects in Cloudera Data Science Workbench on page 50.
2. Open the workbench and launch an engine to run your project. For help, see Using the Workbench Console in Cloudera Data Science Workbench on page 52
3. (Optional) For more mature data science projects that need to run on a recurring schedule, Cloudera Data Science Workbench allows you to create jobs and pipelines. For more details on what you can accomplish by scheduling jobs and pipelines, see Managing Jobs and Pipelines in Cloudera Data Science Workbench on page 53

Managing Projects in Cloudera Data Science Workbench

Projects form the heart of Cloudera Data Science Workbench. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

Creating a Project

To create a Cloudera Data Science Workbench project:

1. Go to Cloudera Data Science Workbench and on the left sidebar, click **Projects**.
2. Click **New Project**.
3. If you are a member of a team, from the drop-down menu, select the **Account** under which you want to create this project. If there is only one account on the deployment, you will not see this option.
4. Enter a **Project Name**.
5. Select **Project Visibility** from one of the following options.
   - **Private** - Only project collaborators can view or edit the project.
   - **Team** - If the project is created under a team account, all members of the team can view the project. Only explicitly-added collaborators can edit the project.
   - **Public** - All authenticated users of Cloudera Data Science Workbench will be able to view the project. Collaborators will be able to edit the project.
6. Under **Initial Setup**, you can either create a blank project, or select one of the following sources for your project files.
   - **Template** - Template projects contain example code that can help you get started with the Cloudera Data Science Workbench. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it does give you the impetus to start using the Cloudera Data Science Workbench right away.
   - **Local** - If you have an existing project on your local disk, use this option to upload compressed file or folder to Cloudera Data Science Workbench.
   - **Git** - If you already use Git for version control and collaboration, you can continue to do so with the Cloudera Data Science Workbench. Specifying a Git URL will clone the project into Cloudera Data Science Workbench. If you use a Git SSH URL, your personal private SSH key will be used to clone the repository. This is the recommended approach. However, you must add the public SSH key from your personal Cloudera Data Science Workbench account to the remote Git hosting service before you can clone the project.
7. Click **Create Project**. After the project is created, you can see your project files and the list of jobs defined in your project.
8. **(Optional)** To work with team members on a project, use the instructions in the following section to add them as collaborators to the project.

Adding Project Collaborators

If you want to work closely with colleagues on a particular project, use the following steps to add them to the project.

1. Navigate to the project overview page.
2. Click **Team** to open the Collaborators page.
3. Search for collaborators by either name or email address and click **Add**.

   For a project created under your personal account, anyone who belongs to your organization can be added as a collaborator. For a project created under a team account, you can only add collaborators that already belong to the team. If you want to work on a project that requires collaborators from different teams, create a new team with the required members, then create a project under that account. If your project was created from a Git repository, each collaborator will have to create the project from the same central Git repository.

   You can grant collaborators one of three levels of access:

   - **Viewer** - Read-only access to code, data, and results.

- **Contributor**: Can view, edit, create, and delete files and environmental variables, run jobs and execute code in running jobs.
- **Admin**: This user has complete access to all aspects of the project, including adding new collaborators, and deleting the entire project.

Warning:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that giving a project collaborator write access to your project code is tantamount to giving them access to your data in CDH. Any collaborators with the *Contributor* and *Admin* roles have write-level access to the same project files as you, and therefore have the ability to use the project code to read/modify the data on your CDH cluster.

Another point to note is that project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of *your* active sessions and modifying project code on the fly. Therefore, it is extremely important to restrict project access to trusted collaborators only.

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

For more information on collaborating effectively, see Sharing Projects and Analysis Results on page 69.

Modifying Project Settings

Project contributors and administrators can modify aspects of the project environment such as the engine being used to launch sessions, the environment variables, and create SSH tunnels to access external resources. To make these changes:

1. Switch context to the account where the project was created.
2. Click **Projects**.
3. From the list of projects, select the one you want to modify.
4. Click **Settings** to open up the Project Settings dashboard.

Options

Modify the project name and its privacy settings on this page.

Engine

Cloudera Data Science Workbench ensures that your code is always run with the specific engine version you selected. You can select the version here. For advanced use cases, Cloudera Data Science Workbench projects can use custom Docker images for their projects. Site administrators can whitelist images for use in projects, and project administrators can use this page to select which of these whitelisted images is installed for their projects. For an example, see Creating Extended Engine Images on page 97.

Environment - If there are any environmental variables that should be injected into all the engines running this project, you can add them to this page. For more details, see Project Environment Variables on page 60.

Tunnels

In some environments, external databases and data sources reside behind restrictive firewalls. Cloudera Data Science Workbench provides a convenient way to connect to such resources using your SSH key. For instructions, see SSH Tunnels on page 113.

Delete Project

This page can only be accessed by project administrators. Remember that deleting a project is irreversible. All files, data, sessions, and jobs will be lost.

Managing Project Files

Important: For use cases beyond simple projects, Cloudera strongly recommends using Git to manage your projects using version control.

Cloudera Data Science Workbench allows you to move, rename, copy, and delete files within the scope of the project where they live. You can also upload new files to a project, or download project files. Files can only be uploaded within the scope of a single project. Therefore, to access a script or data file from multiple projects, you will need to manually upload it to all the relevant projects.

1. Switch context to the account where the project was created.
2. Click **Projects**.
3. From the list of projects, click on the project you want to modify. This will take you to the project overview.
4. Click **Files**.

   Upload Files to a Project

   Click **Upload**. Select **Files** or **Folder** from the dropdown, and choose the files or folder you want to upload from your local filesystem.

   Download Project Files

   Click **Download** to download the entire project in a .zip file. To download only a specific file, select the checkbox next to the file(s) to be download and click **Download**.

   You can also use the checkboxes to **Move**, **Rename**, or **Delete** files within the scope of this project.

Using the Workbench Console in Cloudera Data Science Workbench

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala. It currently supports R, Python, and Scala engines. You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

The workbench includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.

Typically, you would use the following steps to run a project in the workbench:

Launch a Session

To launch a session:

1. Navigate to a project's **Overview** page.
2. Click **Open Workbench**.
3. Use **Select Engine Kernel** to choose your language.
4. Use **Select Engine Profile** to select the number of CPU cores and memory.
5. Click **Launch Session**.

The command prompt at the bottom right of your browser window turns green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Execute Code

You can enter and execute code at the command prompt or the editor. The editor is best for code you want to keep, while the command prompt is best for quick interactive exploration.

If you want to enter more than one line of code at the command prompt, use **Shift-Enter** to move to the next line. Press **Enter** to run your code. The output of your code, including plots, appears in the console.

If you created your project from a template, there are code files in the editor. You can open a file in the editor by double-clicking the file name in the file list.

To run code in the editor:

1. Select a code file in the list on the left.
2. Highlight the code you want to run.
3. Press **Ctrl-Enter** (Windows/Linux) or **Command-Enter** (OSX).

When doing real analysis, writing and executing your code from the editor rather than the command prompt makes it easy to iteratively develop your code and save it along the way.

If you require more space for your editor, you can collapse the file list by double-clicking between the file list pane and the editor pane. You can hide the editor using editor's **View** menu.

Access the Terminal

Cloudera Data Science Workbench provides full terminal access to running engines from the web console. If you run `klist` you should see your authenticated Kerberos principal. If you run `hdfs dfs -ls` you will see the files stored in your HDFS home directory. You do not need to worry about Kerberos authentication.

Use the terminal to move files around, run Git commands, access the YARN and Hadoop CLIs, or install libraries that cannot be installed directly from the engine. You can access the Terminal from a running Session page by clicking **Terminal Access** above the session log pane.

All of your project files are in `/home/cdsw`. Any modifications you make to this folder will persist across runs, while modifications to other folders are discarded.

By default, the terminal does not provide root or sudo access to the container. To install packages that require root access, see Customizing Engine Images.

Stop a Session

When you are done with the session, click **Stop** in the menu bar above the console, or use code to exit by typing the following command:

R

```
quit()
```

Python

```
exit
```

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Managing Jobs and Pipelines in Cloudera Data Science Workbench

Cloudera Data Science Workbench allows you to automate analytics workloads with a built-in job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts. A *job* automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

As data science projects mature beyond ad hoc scripts, you might want to break them up into multiple steps. For example, a project may include one or more data acquisition, data cleansing, and finally, data analytics steps. For such projects, Cloudera Data Science Workbench allows you to schedule multiple jobs to run one after another in what is called a *pipeline*, where each job is dependent on the output of the one preceding it.

Creating a Job

Jobs are created within the scope of a project. When you create a job, you will be asked to select a script to execute as part of the job, and create a schedule for when the job should run. Optionally, you can configure a job to be dependent on another existing job, thus creating a pipeline of tasks to be accomplished in a sequence. Note that the script files and any other job dependencies must exist within the scope of the same project.

1. Navigate to the project for which you want to create a job.
2. On the left-hand sidebar, click **Jobs**.
3. Click **New Job**.
4. Enter a **Name** for the job.
5. Select a script to execute for this job by clicking on the folder icon. You will be able to select a script from a list of files that are already part of the project. To upload more files to the project, see Managing Project Files on page 52.
6. Depending on the code you are running, select an **Engine Kernel** for the job from one of the following options: Python 2, Python 3, R, or Scala.
7. Select a **Schedule** for the job runs from one of the following options.
    - **Manual** - Select this option if you plan to run the job manually each time.
    - **Recurring** - Select this option if you want the job to run in a recurring pattern every X minutes, or on an hourly, daily, weekly or monthly schedule.
    - **Dependent** - Use this option when you are building a pipeline of jobs to run in a predefined sequence. From a dropdown list of existing jobs in this project, select the job that this one should depend on. Once you have configured a dependency, this job will run only after the preceding job in the pipeline has completed a successful run.
8. Select an **Engine Profile** to specify the number of cores and memory available for each session.
9. Enter an optional timeout value in minutes.
10. Click **Set environment variables** if you want to set any values to override the overall project environment variables.
11. Specify a list of **Job Report Recipients** to whom you can send email notifications with detailed job reports for job success, failure, or timeout. You can send these reports to yourself, your team (if the project was created under a team account), or any other external email addresses.
12. Add any **Attachments** such as the console log to the job reports that will be emailed.
13. Click **Create Job**.

    Starting with version 1.1.x, you can use the Jobs API to schedule jobs from third partly workflow tools. For details, see Cloudera Data Science Workbench Jobs API on page 71.

Creating a Pipeline

The **Jobs** overview presents a list of all existing jobs created for a project along with a dependency graph to display any pipelines you've created. Job dependencies do not need to be configured at the time of job creation. Pipelines can be created after the fact by modifying the jobs to establish dependencies between them. From the job overview, you can modify the settings of a job, access the history of all job runs, and view the session output for individual job runs.

Let's take an example of a project that has two jobs, Data Acquisition and Data Analytics. Given that you must acquire the data before you can analyze it, the Data Analytics job should only be triggered after the Data Acquisition job completes a successful run. To create such a two-step pipeline:

1. Navigate to the project where the Data Acquisition and Data Analytics jobs were created.
2. Click **Jobs**.
3. From the list of jobs, select Data Analytics.

4. Click the **Settings** tab.
5. Click on the Schedule dropdown and select **Dependent**. Select Data Acquisition from the dropdown list of existing jobs in the project.
6. Click **Update Job**.

Viewing Job History

1. Navigate to the project where the job was created.
2. Click **Jobs**.
3. Select the relevant job.
4. Click the **History** tab. You will see a list of all the job runs with some basic information such as who created the job, run duration, and status. Click individual runs to see the session output for each run.

Importing Data into Cloudera Data Science Workbench

To work with Cloudera Data Science Workbench, you must import data from local files, Apache HBase, Apache Kudu, Apache Impala, Apache Hive or other external database and data stores such as Amazon S3.

Uploading Data From Your Computer

If you want to create a new project around one or more data files on your computer, select the **Local** option when creating the project.

To add data files from your computer to an existing project, click **Upload** in the Project Overview page.

Accessing Data from HDFS

There are many ways to access HDFS data from R, Python, and Scala libraries. For example, see Example: Reading Data from HDFS (Wordcount) on page 86 for an example of reading data to a Spark 2 program.

You can also use HDFS command line tools from the terminal or by executing system commands in your code. See the documentation at HDFS CLI.

Using Impyla to Access Data from Impala

Impyla is a Python client that can be used to execute queries on Impala tables. It communicates with Impala using the same standard Impala protocol as the ODBC/JDBC drivers. For more details on Impyla, refer to the Impyla project documentation on GitHub and this Cloudera blog post.

To begin, click **Open Workbench**, launch a Python session, and use the workbench prompt to install the following dependencies in your Cloudera Data Science Workbench project. These are required so that Impyla can work with Hive and Kerberos.

Python 2

```
!pip install thrift==0.9.3
!pip install thrift_sasl
!pip install impyla
```

Then, create a new project file and use the following sample code to establish a connection to an Impala daemon on the CDH cluster, and run the `SHOW TABLES` query on Impala's `default` database.

Python 2

```
#impyla_example.py

import os

# Specify IMPALA_HOST as an environment variable in your project settings
IMPALA_HOST = os.getenv('IMPALA_HOST', '<FQDN_Impala_daemon_host>')
```

```
import pandas
from impala.dbapi import connect
from impala.util import as_pandas

# Connect to Impala using Impyla
#
# * If you have not already established your Kerberos credentials in CDSW do so before
 running this script.
# * Remove auth_mechanism and use_ssl parameters on non-secure clusters.
conn = connect(host=IMPALA_HOST,
               port=21050,
               auth_mechanism='GSSAPI',
               use_ssl=True)

# Get the available tables
cursor = conn.cursor()
cursor.execute('SHOW TABLES')

# Pretty output using Pandas
tables = as_pandas(cursor)
tables
```

Accessing Data in Amazon S3 Buckets

Every language in Cloudera Data Science Workbench has libraries available for uploading to and downloading from Amazon S3.

To work with S3:

1. Add your Amazon Web Services access keys to your project's environment variables as AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.
2. Pick your favorite language from the code samples below. Each one downloads the R 'Old Faithful' dataset from S3.

R

```
library("devtools")
install_github("armstrtw/AWS.tools")

Sys.setenv("AWSACCESSKEY"=Sys.getenv("AWS_ACCESS_KEY_ID"))
Sys.setenv("AWSSECRETKEY"=Sys.getenv("AWS_SECRET_ACCESS_KEY"))

library("AWS.tools")

s3.get("s3://sense-files/faithful.csv")
```

Python

```
# Install Boto to the project
!pip install boto

# Create the Boto S3 connection object.
from boto.s3.connection import S3Connection
aws_connection = S3Connection()

# Download the dataset to file 'faithful.csv'.
bucket = aws_connection.get_bucket('sense-files')
key = bucket.get_key('faithful.csv')
key.get_contents_to_filename('/home/cdsw/faithful.csv')
```

Accessing External SQL Databases

Every language in Cloudera Data Science Workbench has multiple client libraries available for SQL databases.

If your database is behind a firewall or on a secure server, you can connect to it by creating an SSH tunnel to the server, then connecting to the database on `localhost`.

If the database is password-protected, consider storing the password in an environmental variable to avoid displaying it in your code or in consoles. The examples below show how to retrieve the password from an environment variable and use it to connect.

Accessing Data From R

R

```
# db.r lets you make direct SQL queries.
install.packages("db.r")
library("db.r")
db <- DB(username="cdswuser", hostname="localhost", port=5432, dbname="test_db",
dbtype="postgres", password=Sys.getenv("POSTGRESQL_PASSWORD"))
db$query("select user_id, user_name from users")

# dplyr lets you program the same way with local data frames and remote SQL databases.

install.packages("dplyr")
library("dplyr")
db <- src_postgres(dbname="test_db", host="localhost", port=5432, user="cdswuser",
password=Sys.getenv("POSTGRESQL_PASSWORD"))
flights_table <- tbl(db, "flights")
select(flights_table, year:day, dep_delay, arr_delay)
```

Accessing Data From Python

You can access data using pyodbc or SQLAlchemy

Python

```
# pyodbc lets you make direct SQL queries.
!wget https://pyodbc.googlecode.com/files/pyodbc-3.0.7.zip
!unzip pyodbc-3.0.7.zip
!cd pyodbc-3.0.7;python setup.py install --prefix /home/cdsw
import os

# See http://www.connectionstrings.com/ for information on how to construct ODBC
connection strings.
db = pyodbc.connect("DRIVER={PostgreSQL
Unicode};SERVER=localhost;PORT=5432;DATABASE=test_db;USER=cdswuser;OPTION=3;PASSWORD=%s"
 % os.environ["POSTGRESQL_PASSWORD"])
cursor = cnxn.cursor()
cursor.execute("select user_id, user_name from users")

# sqlalchemy is an object relational database client that lets you make database queries
 in a more Pythonic way.
!pip install sqlalchemy
import os

import sqlalchemy
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
db = create_engine("postgresql://cdswuser:%s@localhost:5432/test_db" %
os.environ["POSTGRESQL_PASSWORD"])
session = sessionmaker(bind=db)
user = session.query(User).filter_by(name='ed').first()
```

Collaborating Effectively with Git

Cloudera Data Science Workbench provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera Data Science Workbench. Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera Data Science Workbench project from the central Git repository.

For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera Data Science Workbench the same way you would work locally, and for most data scientists and developers that means using Git.

Cloudera Data Science Workbench does not include significant UI support for Git, but instead allows you to use the full power of the command line. If you run an engine and open a terminal, you can run any Git command, including `init`, `add`, `commit`, `branch`, `merge` and `rebase`. Everything should work exactly as it does locally, except that you are running on a distributed edge host directly connected to your Apache Hadoop cluster.

Importing a Project From Git

When you create a project, you can optionally supply an HTTPS or SSH Git URL that points to a remote repository. The new project is a clone of that remote repository. You can commit, push and pull your code by running a console and opening a terminal.

If you want to use SSH to clone the repo, add your personal SSH key to your Git server. For instructions, see Adding SSH Key to GitHub on page 113.

Linking an Existing Project to a Git Remote

If you did not create your project from a Git repository, you can link an existing project to a Git remote (for example, `git@github.com:username/repo.git`) so that you can push and pull your code.

To link to a Git remote:

1. Launch a new session.
2. Open a terminal.
3. Enter the following commands:

   Shell

```
git init
git add *
git commit -a -m 'Initial commit'
git remote add origin git@github.com:username/repo.git
```

You can run `git status` after `git init` to make sure your `.gitignore` includes a folder for libraries and other non-code artifacts.

Installing Packages and Libraries

Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. However, a key feature of Cloudera Data Science Workbench is the ability of different projects to install and use libraries pinned to specific versions, just as you would on your local computer.

You can install additional libraries and packages from the workbench, either using the command prompt or terminal. Alternatively, you might choose to use a package manager such as **Conda** to install and maintain packages and their dependencies. For some basic usage guidelines, see Using Conda with Cloudera Data Science Workbench on page 59.

To install a package from the command prompt:

1. Launch a session.
2. At the command prompt in the bottom right, enter the command to install the package. Some examples using Python and R have been provided.

R

```
# Install from CRAN
install.packages("ggplot2")

# Install using devtools
install.packages('devtools')
library(devtools)
install_github("hadley/ggplot2")
```

Python 2

```
# Installing from console using ! shell operator and pip:
!pip install beautifulsoup

# Installing from terminal
pip install beautifulsoup
```

Python 3

```
# Installing from console using ! shell operator and pip3:
!pip3 install beautifulsoup

# Installing from terminal
pip3 install beautifulsoup
```

Generally, Cloudera recommends you install *all* packages locally into your project. This will ensure you have the exact versions you want and that these libraries will not be upgraded when Cloudera upgrades the base engine image. You only need to install libraries and packages once per project. From then on, they are available to any new engine you spawn throughout the lifetime of the project.

Specify the packages you want in a `requirements.txt` file that lives in your project, then install them using pip/pip3. For example, if you list the following packages in `requirements.txt`:

```
beautifulsoup4==4.6.0
seaborn==0.7.1
```

To install the packages, just run:

```
!pip3 install -r requirements.txt
```

Cloudera Data Science Workbench does not currently support customization of system packages that require root access. However, Cloudera Data Science Workbench site administrators and project administrators can add libraries and other dependencies to the Docker image in which their engines run. See Creating Extended Engine Images on page 97.

Using Conda with Cloudera Data Science Workbench

Cloudera Data Science Workbench recommends using pip for package management along with a `requirements.txt` file (as described in the previous section). However, for users that prefer Conda, the default engine in Cloudera Data Science Workbench includes two environments called `python2.7`, and `python3.6`. These environments are added to `sys.path`, depending on the version of Python selected when you launch a new session.

In Python 2 and Python 3 sessions and attached terminals, Cloudera Data Science Workbench automatically sets the `CONDA_DEFAULT_ENV` and `CONDA_PREFIX` environment variables to point to Conda environments under `/home/cdsw/.conda`.

However, Cloudera Data Science Workbench does not automatically configure Conda to pin the actual Python version. Therefore if you are using Conda to install a package, you must specify the version of Python. For example, to use Conda to install the `feather-format` package into the `python3.6` environment, run the following command in the Workbench command prompt:

```
!conda install -y -c conda-forge python=3.6.1 feather-format
```

To install a package into the `python2.7` environment, run:

```
!conda install -y -c conda-forge python=2.7.11 feather-format
```

Note that on `sys.path`, pip packages have precedence over conda packages.

Note:

- If your project is using an older base engine image (version 3 and lower), you will need to specify both the Python version as well as the Conda environment. For example:

  ```
  !conda install -y -c conda-forge --name python3.6 python=3.6.1
  feather-format
  ```

  The Conda environment is also required when you create an extensible engine using Conda (as described in the following section).

- Cloudera Data Science Workbench does not automatically configure a Conda environment for R and Scala sessions and attached terminals. If you want to use Conda to install packages from an R or Scala session or terminal, you must manually configure Conda to install packages into the desired environment.

Creating an Extensible Engine With Conda

Cloudera Data Science Workbench also allows you to extend its base engine image to include packages of your choice such as Conda. To create an extended engine:

1. Add the following lines to a Dockerfile to extend the base engine, push the engine image to your Docker registry, and whitelist the new engine for your project. For more details on this step, see Extensible Engines.

   Python 2

```
RUN mkdir -p /opt/conda/envs/python2.7
RUN conda install -y nbconvert python=2.7.11 -n python2.7
```

   Python 3

```
RUN mkdir -p /opt/conda/envs/python3.6
RUN conda install -y nbconvert python=3.6.1 -n python3.6
```

2. Set the PYTHONPATH environmental variable as shown below. You can set this either globally in the site administrator dashboard, or for a specific project by going to the project's **Settings** > **Engine** page.

   Python 2

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python2.7/lib/python2.7/site-packages
```

   Python 3

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python3.6/lib/python3.6/site-packages
```

Project Environment Variables

Sometimes your code needs to use secrets, such as passwords and authentication tokens, in order to access external resources.

In general, Cloudera recommends that you not paste secrets into your code. Anyone with read access to your project would be able to view the secrets. Even if you did not give anyone read access, you would have to remember to carefully check any code that you copy and paste into another project, or add to a Git repository.

A better place to store secrets is in your project's environment variables, which you can manage by going to the project's Overview page and from the left sidebar, click **Settings** > **Engine**.

Note: Environmental variable **values** are only visible to collaborators with *contributor* or higher access. They can be used to securely store confidential information such as your AWS or database credentials. The names of the variables are available to all users with access to the project.

These environment variables are set in every engine that runs in your project. The code samples that follow show how to access the environment variable *DATABASE_PASSWORD* from your code.

R

```
database.password <- Sys.getenv("DATABASE_PASSWORD")
```

Python

```
import os
database_password = os.environ["DATABASE_PASSWORD"]
```

Scala

```
System.getenv("DATABASE_PASSWORD")
```

Engine Environment Variables

The following table lists environment variables that can be set in every engine.

Environment Variable	Description
`CDSW_PROJECT`	The project to which this engine belongs.
`CDSW_CREATOR`	The username of the creator of this engine.
`CDSW_ENGINE_ID`	The ID of this engine. For sessions, this appears in your browser's URL bar.
`CDSW_MASTER_ID`	If this engine is a worker, this is the `CDSW_ENGINE_ID` of its master.
`CDSW_MASTER_IP`	If this engine is a worker, this is the IP address of its master.
`CDSW_PUBLIC_PORT`	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind `CDSW_PUBLIC_PORT` will be available in browsers at: http(s)://<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. By default, `CDSW_PUBLIC_PORT` is set to **8080**. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera Data Science Workbench web application, as long as the job or session is still running. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 64. **Note:** In Cloudera Data Science Workbench 1.2.x, setting `CDSW_PUBLIC_PORT` to a non-default port number is not supported.
`CDSW_DOMAIN`	The domain on which Cloudera Data Science Workbench is being served. This can be useful for iframing services, as demonstrated in the Shiny example.
`CDSW_CPU_MILLICORES`	The number of CPU cores allocated to this engine, expressed in thousandths of a core.
`CDSW_MEMORY_MB`	The number of megabytes of memory allocated to this engine.

Environment Variable	Description
`CDSW_IP_ADDRESS`	Other engines in the Cloudera Data Science Workbench cluster can contact this engine on this IP address.
`IDLE_MAXIMUM_MINUTES`	Maximum number of minutes a session can remain idle before it exits. **Default:** 60 minutes **Maximum Value:** 35,000 minutes
`SESSION_MAXIMUM_MINUTES`	Maximum number of minutes a session can run before it times out. **Default:** 60*24*7 minutes (7 days) **Maximum Value:** 35,000 minutes
`JOB_MAXIMUM_MINUTES`	Maximum number of minutes a job can run before it times out. **Default:** 60*24*7 minutes (7 days) **Maximum Value:** 35,000 minutes
`CONDA_DEFAULT_ENV`	Points to the default Conda environment so you can use Conda to install/manage packages in the Workbench. For more details on when to use this variable, see Using Conda with Cloudera Data Science Workbench on page 59.

Distributed Computing with Workers

For distributed computing, such as cross-validating a model or tuning some hyper parameters, Cloudera Data Science Workbench provides basic support for leveraging multiple engine instances from a single run. Any R or Python engine can spawn other engines, known as *workers*, and give them code to execute when they start up. Worker output is displayed in the main console to allow you to debug your code. These workers are terminated when the session exits.

For more significant distributed computing needs, using Cloudera Distribution of Apache Spark 2 from within Cloudera Data Science Workbench is strongly recommended.

Spawning Workers

Select a language from the code samples below to launch workers:

R

```
library("cdsw")
workers <- launch.workers(n=2, cpu=0.2, memory=0.5, code="print('Hello from a CDSW
Worker')")
```

Python

```
import cdsw
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, code="print 'Hello from a CDSW
 Worker'")
```

Worker Network Communication

Workers are a low-level feature to help use higher level libraries that can operate across multiple nodes. As such, you will generally want to use workers only to launch the backends for these libraries.

To help you get your workers or distributed computing framework components talking to one another, every worker engine run includes an environmental variable `CDSW_MASTER_IP` with the fully addressable IP of the master engine. Every engine has a dedicated IP access with no possibility of port conflicts.

For instance, the following are trivial examples of two worker engines talking to the master engine.

R

From the master engine, the following `master.R` script will launch two workers and accept incoming connections from them.

```
# master.R

library("cdsw")

# Launch two CDSW workers. These are engines that will run in
# the same project, execute a given code or script, and exit.
workers <- launch.workers(2, cpu=0.2, memory=0.5, script="worker.R")

# Accept two connections, one from each worker. Workers will
# execute worker.R.
for(i in c(1,2)) {
  # Receive a message from each worker and return a response.
  con <- socketConnection(host="0.0.0.0", port = 6000, blocking=TRUE, server=TRUE,
open="r+")
  data <- readLines(con, 1)
  print(paste("Server received:", data))
  writeLines("Hello from master!", con)
  close(con)
}
```

The workers will execute the following `worker.R` script and respond to the master.

```
# worker.R

print(Sys.getenv("CDSW_MASTER_IP"))
con <- socketConnection(host=Sys.getenv("CDSW_MASTER_IP"), port = 6000, blocking=TRUE,
 server=FALSE, open="r+")
write_resp <- writeLines("Hello from Worker", con)
server_resp <- readLines(con, 1)
print(paste("Worker received:  ", server_resp))
close(con)
```

Python

From the master engine, the following `master.py` script will launch two workers and accept incoming connections from them.

```
# master.py

import cdsw, socket

# Launch two CDSW workers. These are engines that will run in
# the same project, execute a given code or script, and exit.
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, script="worker.py")

# Listen on TCP port 6000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 6000))
s.listen(1)

# Accept two connections, one from each worker. Workers will
# execute worker.py.
conn, addr = s.accept()
for i in range(2):
    # Receive a message from each worker and return a response.
    data = conn.recv(20)
    if not data: break
    print "Master received:", data
    conn.send("Hello From Server!")
conn.close()
```

The workers will execute the following `worker.py` script and respond to the master.

```
# worker.py

import os, socket

# Open a TCP connection to the master.
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((os.environ["CDSW_MASTER_IP"], 6000))

# Send some data and receive a response.
s.send("Hello From Worker!")
data = s.recv(1024)
s.close()

print "Worker received:", data
```

Accessing Web User Interfaces from Cloudera Data Science Workbench

This topic describes the different ways in which Cloudera Data Science Workbench allows you to access user interfaces for applications such as Cloudera Manager, Hue, and even the transient per-session UIs for frameworks such as Spark 2, TensorFlow, Shiny, and so on.

Cloudera Manager, Hue, and the Spark History Server

Cloudera Data Science Workbench also gives you a way to access your CDH cluster's Cloudera Manager and Hue UIs from within the Cloudera Data Science Workbench application. Spark 2 provides a UI that displays information and logs for completed Spark applications, which is useful for debugging and performance monitoring. This UI, called the History Server, runs on the CDH cluster, on a configurable node and port.

To access these applications, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the UI you want to visit from the dropdown.

Web UIs Embedded in Jobs and Sessions

Many data science libraries and processing frameworks include user interfaces to help track progress of your jobs and break down workflows. These are instrumental in debugging and using the platforms themselves. For example, Spark provides a Spark Web UI to monitor running applications and TensorFlow visualizations can be run on TensorBoard. Other web application frameworks such as Shiny and Flask are popular ways for data scientists to display additional interactive analysis in the languages they already know.

With version 1.1.x, Cloudera Data Science Workbench allows you to access these web UIs directly from sessions and jobs. This feature is particularly helpful when you want to monitor and track progress for batch jobs. Even though jobs don't give you access to the interactive workbench console, you can still track long running jobs through the UI. However, note that the UI is only active so long as the job/session is active. If your session times out after 60 minutes (default timeout value), so will the UI.

Cloudera Data Science Workbench exposes web applications in one of the following ways:

Spark 2 Web UIs (CDSW_SPARK_PORT)

Spark 2 exposes one web UI for each Spark application driver running in Cloudera Data Science Workbench. The UI will be running within the container, on the port specified by the environmental variable `CDSW_SPARK_PORT`. By default, `CDSW_SPARK_PORT` is set to **20049**. The web UI will exist only as long as a SparkContext is active within a session. The port is freed up when the SparkContext is shutdown.

Spark 2 web UIs are available in browsers at: `https://spark-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>`. To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select **Spark UI** from the dropdown. For a job, navigate to the job overview page and click the **History** tab. Click on a job run to open the session output for the job. You can now click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application to access the Spark UI for this session.

TensorBoard, Shiny, and others (`CDSW_PUBLIC_PORT`)

`CDSW_PUBLIC_PORT` is an environment variable that points to a general purpose public port. By default, `CDSW_PUBLIC_PORT` is set to port **8080**. Any HTTP services running in containers that bind to `CDSW_PUBLIC_PORT` are available in browsers at: `http://<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>`. Therefore, TensorBoard, Shiny, Flask or any other web framework accompanying a project can be accessed directly from within a session or job, as long as it is run on `CDSW_PUBLIC_PORT`.

Note: In Cloudera Data Science Workbench 1.2.x, setting `CDSW_PUBLIC_PORT` to a non-default port number is not supported.

To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the UI from the dropdown. For a job, navigate to the job overview page and click the **History** tab. Click on a job run to open the session output for the job. You can now click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application to access the UI for this session.

Example: A Shiny Application

This example demonstrates how to create and run a Shiny application and view the associated UI while in an active session.

Create a new, blank project and run an R console. Create the files, `ui.R` and `server.R`, in the project, and copy the contents of the following example files provided by Shiny by RStudio:

R

```
# ui.R

library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

R

```
# server.R

library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
```

```
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Run the following code in the interactive workbench prompt to install the Shiny package, load the library into the engine, and run the Shiny application.

R

```
install.packages('shiny')

library('shiny')

runApp(port=as.numeric(Sys.getenv("CDSW_PUBLIC_PORT")),
host=Sys.getenv("CDSW_IP_ADDRESS"), launch.browser="FALSE")
```

Finally, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the Shiny UI, **Hello Shiny!**, from the dropdown. The UI will be active as long as the session is still running.

Data Visualization

Each language on Cloudera Data Science Workbench has a visualization system that you can use to create plots, including rich HTML visualizations.

Simple Plots

To create a simple plot, run a console in your favorite language and paste in the following code sample:

R

```
# A standard R plot
plot(rnorm(1000))

# A ggplot2 plot
library("ggplot2")
qplot(hp, mpg, data=mtcars, color=am,
facets=gear~cyl, size=I(3),
xlab="Horsepower", ylab="Miles per Gallon")
```

Python 2

```
import matplotlib.pyplot as plt
import random
plt.plot([random.normalvariate(0,1) for i in xrange(1,1000)])
```

For some libraries such as `matplotlib`, new plots are displayed as each subsequent command is executed. Therefore, when you run a series of commands, you will see incomplete plots for each intermediate command until the final command is executed. If this is not the desired behavior, an easy workaround is to put all the plotting commands in one Python function.

Saved Images

You can also display images, using a command in the following format:

R

```
library("cdsw")

download.file("https://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png",
"/cdn/Minard.png")
image("Minard.png")
```

Python 2

```
import urllib
from IPython.display import Image
urllib.urlretrieve("http://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png",
"Minard.png")

Image(filename="Minard.png")
```

HTML Visualizations

Your code can generate and display HTML. To create an HTML widget, paste in the following:

R

```
library("cdsw")
html('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Python 2

```
from IPython.display import HTML
HTML('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Scala

Cloudera Data Science Workbench allows you to build visualization libraries for Scala using jvm-repr. The following example demonstrates how to register a custom HTML representation with the `"text/html"` mimetype in Cloudera Data Science Workbench. This output will render as HTML in your workbench session.

```
//HTML representation
case class HTML(html: String)

//Register a displayer to render html
Displayers.register(classOf[HTML],
  new Displayer[HTML] {
    override def display(html: HTML): java.util.Map[String, String] = {
      Map(
        "text/html" -> html.html
      ).asJava
    }
  })

val helloHTML = HTML("<h1> <em> Hello World </em> </h1>")

display(helloHTML)
```

IFrame Visualizations

Most visualizations require more than basic HTML. Embedding HTML directly in your console also risks conflicts between different parts of your code. The most flexible way to embed a web resource is using an IFrame:

R

```
library("cdsw")
iframe(src="https://www.youtube.com/embed/8pHzROP1D-w", width="854px", height="510px")
```

Python 2

```
from IPython.display import HTML
HTML('<iframe width="854" height="510"
src="https://www.youtube.com/embed/8pHzROP1D-w"></iframe>')
```

You can generate HTML files within your console and display them in IFrames using the `/cdn` folder. The cdn folder persists and services static assets generated by your engine runs. For instance, you can embed a full HTML file with IFrames.

R

```
library("cdsw")
f <- file("/cdn/index.html")
html.content <- paste("<p>Here is a normal random variate:", rnorm(1), "</p>")
writeLines(c(html.content), f)
close(f)
iframe("index.html")
```

Python 2

```
from IPython.display import HTML
import random

html_content  = "<p>Here is a normal random variate: %f </p>" % random.normalvariate(0,1)

file("/cdn/index.html", "w").write(html_content)
HTML("<iframe src=index.html>")
```

Cloudera Data Science Workbench uses this feature to support many rich plotting libraries such as htmlwidgets, Bokeh, and Plotly.

Grid Displays

Cloudera Data Science Workbench supports native grid displays of DataFrames across several languages.

Python 3

Using DataFrames with the pandas package requires per-session activation:

```
import pandas as pd
pd.options.display.html.table_schema = True
pd.DataFrame(data=[range(1,100)])
```

For PySpark DataFrames, use pandas and run `df.toPandas()` on a PySpark DataFrame. This will bring the DataFrame into local memory as a pandas DataFrame.

Note:

A Python project originally created with engine 1 will be running pandas version 0.19, and will not auto-upgrade to version 0.20 by simply selecting engine 2 in the project's **Settings** > **Engine** page.

The pandas data grid setting only exists starting in version 0.20.1. To upgrade, manually install version 0.20.1 at the session prompt.

```
!pip install pandas==0.20.1
```

R

In R, DataFrames will display as grids by default. For example, to view the Iris data set, you would just use:

```
iris
```

Similar to PySpark, bringing Sparklyr data into local memory with `as.data.frame` will output a grid display.

```
sparkly_df %>% as.data.frame
```

Scala

Calling the `display()` function on an existing dataframe will trigger a collect, much like `df.show()`.

```
val df = sc.parallelize(1 to 100).toDF()
display(df)
```

Documenting Your Analysis

Cloudera Data Science Workbench supports Markdown documentation of your code written in comments. This allows you to generate reports directly from valid Python and R code that runs anywhere, even outside Cloudera Data Science Workbench. To add documentation to your analysis, create comments in Markdown format:

R

```
# Heading
# -------
#
# This documentation is **important.**
#
# Inline math: $e^ x$
#
# Display math: $$y = \Sigma x + \epsilon$$

print("Now the code!")
```

Python

```
# Heading
# -------
#
# This documentation is **important.**
#
# Inline math: $e^ x$
#
# Display math: $$y = \Sigma x + \epsilon$$

print("Now the code!")
```

Sharing Projects and Analysis Results

Cloudera Data Science Workbench supports several collaboration models.

Project Collaborators

If you want to work closely with trusted colleagues on a particular project, add them to the project as collaborators. For instructions, see Adding Project Collaborators on page 50.

Warning:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that giving a project collaborator write access to your project code is tantamount to giving them access to your data in CDH. Any collaborators with the *Contributor* and *Admin* roles have write-level access to the same project files as you, and therefore have the ability to use the project code to read/modify the data on your CDH cluster.

Another point to note is that project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of *your* active sessions and modifying project code on the fly. Therefore, it is extremely important to restrict project access to trusted collaborators only.

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

Sharing Projects Publicly

Public projects on Cloudera Data Science Workbench grant read-level access to *everyone* with access to the Cloudera Data Science Workbench application. That means everyone can view the project's files and results, but only those whom you have granted write-level access or higher can edit files, run engines, or view the project's environment variables.

You can include a Markdown-formatted `README.md` file in public projects to document your project's purpose and usage.

If you are a project admin, you can set a project's visibility to **Public** from the **Project** > **Settings** > **Options** page. For instructions, see Modifying Project Settings on page 51.

Forking Projects

You can fork another user's project by clicking **Fork** on the **Project** page. Forking creates a new project under your account that contains all the files, libraries, configuration, and jobs from the original project.

Creating sample projects that other users can fork helps to bootstrap new projects and encourage common conventions.

Note: An issue exists where a timeout might occur when forking large projects.

Sharing Job and Session Console Outputs

Cloudera Data Science Workbench lets you easily share the results of your analysis with one click. Using rich visualizations and documentation comments, you can arrange your console log so that it is a readable record of your analysis and results. This log continues to be available even after the session stops. This method of sharing allows you to show colleagues and collaborators your progress without your having to spend time creating a report.

To share results from an interactive session, click **Share** at the top of the console page. From here you can generate a link that includes a secret token that gives access to that particular console output. For jobs results, you can either share a link to the latest job result or a particular job run. To share the latest job result, click the **Latest Run** link for a job on the Overview page. This link will always have the latest job results. To share a particular run, click on a job run in the job's **History** page and share the corresponding link.

You can share console outputs with one of the following sets of users.

- **All anonymous users with the link** - By default, Cloudera Data Science Workbench allows anonymous access to shared consoles. However, site administrators can disable anonymous sharing at any time by going to **Admin** > **Security**, disabling the **Allow anonymous access to shared console outputs** checkbox, and clicking **Disable anonymous access** to confirm.

Once anonymous sharing has been disabled, all existing publicly shared console outputs will be updated to be viewable only by authenticated users.

- **All authenticated users with the link** - This means any user with a Cloudera Data Science Workbench account will have access to the shared console.
- **Specific users and teams** - Click **Change** to search for users and teams to give access to the shared console. You can also come back to the session and revoke access from a user or team the same way.

Sharing Data Visualizations

If you want to share a single data visualization rather than an entire console, you can embed it in another web page. Click the small circular 'link' button located to the left of most rich visualizations to view the HTML snippet that you can use to embed the visualization.

Cloudera Data Science Workbench Jobs API

Cloudera Data Science Workbench exposes a REST API that allows you to schedule jobs from third-party workflow tools. You must authenticate yourself before you can use the API to submit a job run request. The Jobs API supports HTTP Basic Authentication, accepting the same users and credentials as Cloudera Data Science Workbench.

API Key Authentication

Cloudera recommends using your API key for requests instead of your actual username/password so as to avoid storing and sending your credentials in plaintext. The API key is a randomly generated token that is unique to each user. It must be treated as highly sensitive information because it can be used to start jobs via the API. To look up your Cloudera Data Science Workbench API key:

1. Sign in to Cloudera Data Science Workbench.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click **Settings**.
4. Select the **API Key** tab.

The following example demonstrates how to construct an HTTP request using the standard basic authentication technique. Most tools and libraries, such as Curl and Python Requests, support basic authentication and can set the required **Authorization** header for you. For example, with `curl` you can pass the API Key to the `--user` flag and leave the password field blank.

```
curl -v -XPOST http://cdsw.example.com/api/v1/<path_to_job> --user "<API_KEY>:"
```

To access the API using a library that does not provide Basic Authentication convenience methods, set the request's **Authorization** header to `Basic <API_KEY_encoded_in_base64>`. For example, if your API key is `uysgxtj7jzkps96njextnxxmq05usp0b`, set **Authorization** to `Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMGI6`.

Starting a Job Run Using the API

Once a job has been created and configured through the Cloudera Data Science Workbench web application, you can start a run of the job through the API. This will constitute sending a POST request to a job start URL of the form: `http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start`.

To construct a request, use the following steps to derive the username, project name, and job ID from the job's URL in the web application.

1. Log in to the Cloudera Data Science Workbench web application.
2. Switch context to the team/personal account where the parent project lives.
3. Select the project from the list.

4. From the project's **Overview**, select the job you want to run. This will take you to the job **Overview** page. The URL for this page is of the form: `http://cdsw.example.com/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>`.
5. Use the `$USERNAME`, `$PROJECT_NAME`, and `$JOB_ID` parameters from the job Overview URL to create the following job start URL:
   `http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start`.

   For example, if your job Overview page has the URL `http://cdsw.example.com/alice/sample-project/jobs/123`, then a sample `POST` request would be of the form:

```
curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/sample-project/jobs/123/start
    \
--user "<API_KEY>:" --header "Content-type: application/json"
```

   Note that the request must have the **Content-Type** header set to `application/json`, even if the request body is empty.

Setting Environment Variables

You can set environment variables for a job run by passing parameters in the API request body in a JSON-encoded object with the following format.

```
{
    "environment": {
        "ENV_VARIABLE": "value 1",
        "ANOTHER_ENV_VARIABLE": "value 2"
    }
}
```

The values set here will override the defaults set for the project and the job in the web application. This request body is optional and can be left blank.

Be aware of potential conflicts with existing defaults for environment variables that are crucial to your job, such as PATH and the `CDSW_*` variables.

Sample Job Run

As an example, let’s assume user Alice has created a project titled Risk Analysis. Under the Risk Analysis project, Alice has created a job with the ID, 208. Using `curl`, Alice can use her API Key (`uysgxtj7jzkps96njextnxxmq05usp0b`) to create an API request as follows:

```
curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/start
      \
--user "uysgxtj7jzkps96njextnxxmq05usp0b:" --header "Content-type: application/json"
         \
--data '{"environment": {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}}'
```

In this example, START_DATE and END_DATE are environment variables that are passed as parameters to the API request in a JSON object.

In the resulting HTTP request, `curl` automatically encodes the **Authorization** request header in base64 format.

```
* Connected to cdsw.example.com (10.0.0.3) port 80 (#0)
* Server auth using Basic with user 'uysgxtj7jzkps96njextnxxmq05usp0b'
> POST /api/v1/projects/alice/risk-analysis/jobs/21/start HTTP/1.1
> Host: cdsw.example.com
> Authorization: Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMGI6
> User-Agent: curl/7.51.0
> Accept: */*
> Content-type: application/json
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json; charset=utf-8
< Date: Mon, 10 Jul 2017 12:00:00 GMT
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
<
{
  "engine_id": "cwg6wclmg0x482u0"
}
```

You can confirm that the job was started by going to the Cloudera Data Science Workbench web application.

Starting a Job Run Using Python

To start a job run using Python, Cloudera recommends using Requests, an HTTP library for Python; it comes with a convenient API that makes it easy to submit job run requests to Cloudera Data Science Workbench. Extending the Risk Analysis example from the previous section, the following sample Python code will create an HTTP request to run the job with the job ID, 208.

Python 2

```
# example.py

import requests
import json

HOST = "http://cdsw.example.com"
USERNAME = "alice"
API_KEY = "uysgxtj7jzkps96njextnxxmq05usp0b"
PROJECT_NAME = "risk-analysis"
JOB_ID = "208"

url = "/".join([HOST, "api/v1/projects", USERNAME, PROJECT_NAME, "jobs", JOB_ID, "start"])
job_params = {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}
res = requests.post(
    url,
    headers = {"Content-Type": "application/json"},
    auth = (API_KEY,""),
    data = json.dumps({"environment": job_params})
)

print "URL", url
print "HTTP status code", res.status_code
print "Engine ID", res.json().get('engine_id')
```

When you run the code, you should see output of the form:

```
$ python example.py
URL http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/start
HTTP status code 200
Engine ID r11w5q3q589ryg9o
```

Limitations

- Cloudera Data Science Workbench does not support changing your API key, or having multiple API keys.
- Currently, you cannot create a job, stop a job, or get the status of a job using the Jobs API.

Jupyter Magic Commands

Cloudera Data Science Workbench's Scala and Python kernels are based on Jupyter kernels. Jupyter kernels support varying magic commands that extend the core language with useful shortcuts. This section details the magic commands (magics) supported by Cloudera Data Science Workbench.

Line magics begin with a single %: for example, `%timeit`. *Cell magics* begin with a double %%: for example, `%%bash`.

Python

In the default Python 2.7.11 engine, Cloudera Data Science Workbench supports most line magics, but no cell magics.

Cloudera Data Science Workbench supports the shell magic `!`: for example, `!ls -alh /home/cdsw`.

Cloudera Data Science Workbench supports the help magics ? and ??: for example, `?numpy` and `??numpy`. ? displays the docstring for its argument. ?? attempts to print the source code. You can get help on magics using the ? prefix: for example, `?%timeit`.

Cloudera Data Science Workbench supports the line magics listed at https://ipython.org/ipython-doc/3/interactive/magics.html#line-magics, with the following exceptions:

- `%colors`
- `%debug`
- `%edit`
- `%gui`
- `%history`
- `%install_default_config`
- `%install_profiles`
- `%lsmagic`
- `%macro`
- `%matplotlib`
- `%notebook`
- `%page`
- `%pastebin`
- `%pdb`
- `%prun`
- `%pylab`
- `%recall`
- `%rerun`
- `%save`
- `%sc`

Scala

Cloudera Data Science Workbench's Scala kernel is based on Apache Toree. It supports the line magics documented in the Apache Toree magic tutorial.

Managing the Cloudera Data Science Workbench Service

Important: Cloudera Data Science Workbench (1.4.x and lower) is not supported with Cloudera Manager 6.0.0 and CDH 6.0.0. Cloudera Data Science Workbench will be supported with Cloudera Enterprise 6 in a future release.

This topic describes how to configure and manage Cloudera Data Science Workbench using Cloudera Manager. The contents of this topic only apply to CSD-based deployments. If you installed Cloudera Data Science Workbench using the RPM, the Cloudera Data Science Workbench service will not be available to you in Cloudera Manager.

Adding the Cloudera Data Science Workbench Service

Cloudera Data Science Workbench is available as an add-on service for Cloudera Manager. To install Cloudera Data Science Workbench, you require the following files: a CSD JAR file that contains all the configuration needed to describe and manage the new Cloudera Data Science Workbench service, and the Cloudera Data Science Workbench parcel.

To install this service, first download and copy the CSD file to the Cloudera Manager Server host. Then use Cloudera Manager to distribute the Cloudera Data Science Workbench parcel to the relevant gateway nodes. You can then use Cloudera Manager's **Add Service** wizard to add the Cloudera Data Science Workbench service to your cluster.

For the complete set of instructions, see Install Cloudera Data Science Workbench on page 37.

Accessing Cloudera Data Science Workbench from Cloudera Manager

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click **CDSW Web UI** to visit the Cloudera Data Science Workbench web application.

Configuring Cloudera Data Science Workbench Properties

In a CSD-based deployment, Cloudera Manager allows you to configure Cloudera Data Science Workbench properties without having to directly edit any configuration file.

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click the **Configuration** tab.
4. Use the search bar to look for the property you want to configure. You can use Cloudera Manager to configure proxies, enable TLS, and enable GPU support for Cloudera Data Science Workbench.

   If you have recently upgraded to a CSD-based deployment, a list of the properties in `cdsw.conf`, along with their corresponding properties in Cloudera Manager can be found in the upgrade guide here.
5. Click **Save Changes**.

Starting, Stopping, and Restarting the Service

Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the cdsw_protect_stop_restart.sh script. This is to help avoid the data loss issue detailed in TSB-346.

To start, stop, and restart the Cloudera Data Science Workbench service:

1. Log into the Cloudera Manager Admin Console.
2. On the **Home** > **Status** tab, click

   ▾

   to the right of the **CDSW** service and select the action (**Start**, **Stop**, or **Restart**) you want to perform from the dropdown.
3. Confirm your choice on the next screen. When you see a **Finished** status, the action is complete.

> **Note:** After a restart, the Cloudera Data Science Workbench service in Cloudera Manager will display **Good** health even though the Cloudera Data Science Workbench web application might need a few more minutes to get ready to serve requests.

Managing Cloudera Data Science Workbench Worker Hosts

You can add or remove workers from Cloudera Data Science Workbench using Cloudera Manager. For instructions, see:

- Adding a Worker Host
- Removing a Worker Host

Health Tests

Cloudera Manager runs a few health tests to confirm whether Cloudera Data Science Workbench and it's components (Master and Workers) are running, and ready to serve requests.

You can choose to enable or disable individual or summary health tests, and in some cases specify what should be included in the calculation of overall health for the service, role instance, or host. See Configuring Monitoring Settings for more information.

Creating Diagnostic Bundles

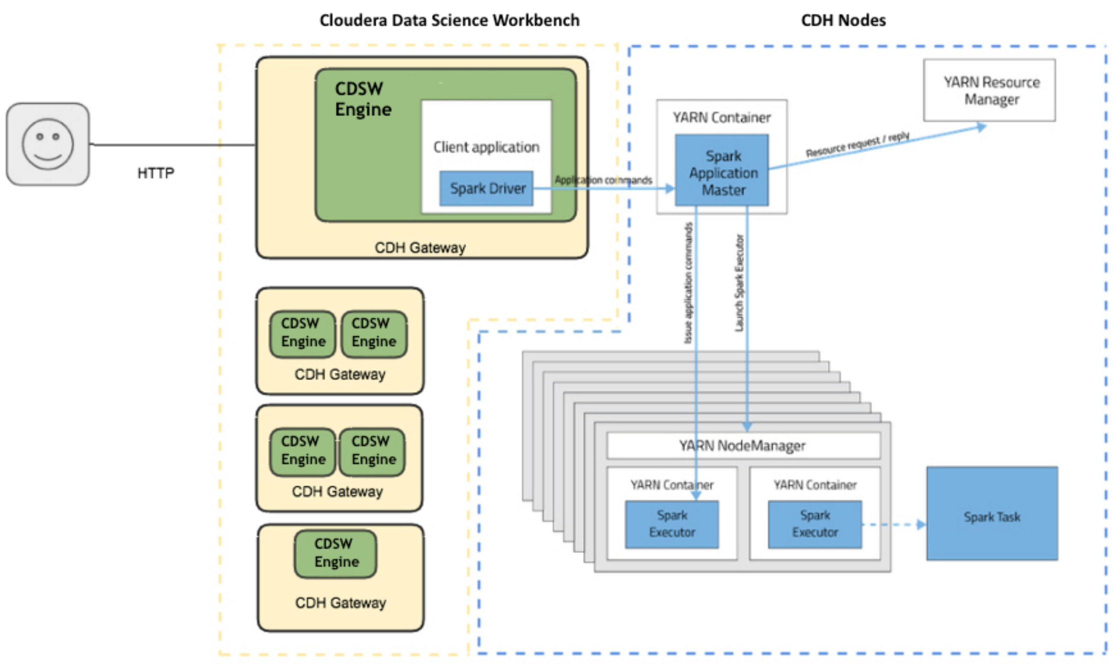
Diagnostic data for Cloudera Data Science Workbench is now available as part of the Cloudera Manager diagnostic bundle. For details on usage and diagnostic data collection in Cloudera Data Science Workbench, see Data Collection in Cloudera Data Science Workbench on page 101.

Using Cloudera Distribution of Apache Spark 2 with Cloudera Data Science Workbench

Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.

Cloudera Data Science Workbench provides interactive and batch access to Spark 2. Connections are fully secure without additional configuration, with each user accessing Spark using their Kerberos principal. With a few extra lines of code, you can do anything in Cloudera Data Science Workbench that you might do in the Spark shell, as well as leverage all the benefits of the workbench. Your Spark applications will run in an isolated project workspace.

Cloudera Data Science Workbench's interactive mode allows you to launch a Spark application and work iteratively in R, Python, or Scala, rather than the standard workflow of launching an application and waiting for it to complete to view the results. Because of its interactive nature, Cloudera Data Science Workbench works with Spark on YARN's `client` mode, where the driver persists through the lifetime of the job and runs executors with full access to the CDH cluster resources. This architecture is illustrated the following figure:



The rest of this guide describes how to set Spark 2 environment variables, manage package dependencies, and how to configure logging. It also consists of instructions and sample code for running R, Scala, and Python projects from Spark 2.

Cloudera Data Science Workbench allows you to access the Spark History Server and even transient per-session UIs for Spark 2 directly from the workbench console. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 64.

Configuring Cloudera Distribution of Apache Spark 2

This topic describes how to set Spark 2 environment variables, manage package dependencies for Spark 2 jobs, and how to configure logging.

Spark Configuration Files

Cloudera Data Science Workbench supports configuring Spark 2 properties on a per project basis with the `spark-defaults.conf` file. If there is a file called `spark-defaults.conf` in your project root, this will be automatically be added to the global Spark defaults. To specify an alternate file location, set the environmental variable, `SPARK_CONFIG`, to the path of the file relative to your project. If you're accustomed to submitting a Spark job with key-values pairs following a `--conf` flag, these can also be set in a `spark-defaults.conf` file instead. For a list of valid key-value pairs, refer the Spark configuration reference documentation.

Administrators can set environment variable paths in the `/etc/spark2/conf/spark-env.sh` file.

You can also use Cloudera Manager to configure `spark-defaults.conf` and `spark-env.sh` globally for all Spark applications as follows.

Configuring Global Properties Using Cloudera Manager

Configure client configuration properties for all Spark applications in `spark-defaults.conf` as follows:

1. Go to the Cloudera Manager Admin Console.
2. Navigate to the Spark service.
3. Click the **Configuration** tab.
4. Search for the **Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf** property.
5. Specify properties described in Application Properties. If more than one role group applies to this configuration, edit the value for the appropriate role group.
6. Click **Save Changes** to commit the changes.
7. Deploy the client configuration.

For more information on using a `spark-defaults.conf` file for Spark jobs, visit the Apache Spark 2 reference documentation.

Configuring Spark Environment Variables Using Cloudera Manager

Configure service-wide environment variables for all Spark applications in `spark-env.sh` as follows:

1. Go to the Cloudera Manager Admin Console.
2. Navigate to the Spark 2 service.
3. Click the **Configuration** tab.
4. Search for the **Spark Service Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-env.sh** property and add the paths for the environment variables you want to configure.
5. Click **Save Changes** to commit the changes.
6. Restart the service.
7. Deploy the client configuration.

Managing Memory Available for Spark Drivers

By default, the amount of memory allocated to Spark driver processes is set to a `0.8` fraction of the total memory allocated for the engine container. If you want to allocate more or less memory to the Spark driver process, you can override this default by setting the `spark.driver.memory` property in `spark-defaults.conf` (as described above).

Managing Dependencies for Spark 2 Jobs

As with any Spark job, you can add external packages to the interpreter on startup. To add external dependencies to Spark jobs, specify the libraries you want added by using the appropriate configuration parameter in a `spark-defaults.conf` file. The following table lists the most commonly used configuration parameters for adding dependencies and how they can be used:

Property	Description
`spark.files`	Comma-separated list of files to be placed in the working directory of each Spark executor.
`spark.submit.pyFiles`	Comma-separated list of `.zip`, `.egg`, or `.py` files to place on `PYTHONPATH` for Python applications.
`spark.jars`	Comma-separated list of local jars to include on the Spark driver and Spark executor classpaths.
`spark.jars.packages`	Comma-separated list of Maven coordinates of jars to include on the Spark driver and Spark executor classpaths. When configured, Spark will search the local Maven repo, and then Maven central and any additional remote repositories configured by `spark.jars.ivy`. The format for the coordinates are `groupId:artifactId:version`.
`spark.jars.ivy`	Comma-separated list of additional remote repositories to search for the coordinates given with `spark.jars.packages`.

Example spark-defaults.conf

Here is a sample `spark-defaults.conf` file that uses some of the Spark configuration parameters discussed in the previous section to add external packages on startup.

```
spark.jars.packages org.scalaj:scalaj-http_2.11:2.3.0
spark.jars my_sample.jar
spark.files data/test_data_1.csv,data/test_data_2.csv
```

`spark.jars.packages`

The `scalaj` package will be downloaded from Maven central and included on the Spark driver and executor classpaths.

`spark.jars`

The pre-existing jar, `my_sample.jar`, residing in the root of this project will be included on the Spark driver and executor classpaths.

`spark.files`

The two sample data sets, `test_data_1.csv` and `test_data_2.csv`, from the `/data` directory of this project will be distributed to the working directory of each Spark executor.

For more advanced configuration options, visit the Apache Spark 2 reference documentation.

Spark Logging Configuration

Cloudera Data Science Workbench allows you to update Spark's internal logging configuration on a per-project basis. Spark 2 uses Apache Log4j, which can be configured through a properties file. By default, a `log4j.properties` file found in the root of your project will be appended to the existing Spark logging properties for every session and job. To specify a custom location, set the environmental variable `LOG4J_CONFIG` to the file location relative to your project.

The Log4j documentation has more details on logging options.

Increasing the log level or pushing logs to an alternate location for troublesome jobs can be very helpful for debugging. For example, this is a `log4j.properties` file in the root of a project that sets the logging level to INFO for Spark jobs.

```
shell.log.level=INFO
```

PySpark logging levels should be set as follows:

```
log4j.logger.org.apache.spark.api.python.PythonGatewayServer=<LOG_LEVEL>
```

And Scala logging levels should be set as:

```
log4j.logger.org.apache.spark.repl.Main=<LOG_LEVEL>
```

Using Spark 2 from Python

Cloudera Data Science Workbench supports using Spark 2 from Python via PySpark.

Setting Up Your PySpark environment

1. Open Cloudera Data Science Workbench.
2. Click **New Project**.
3. Enter a **Project Name**.
4. Choose whether the **Project Visibility** is *Private* or *Public*.
5. Under **Initial Setup**, choose the **Template** tab.
6. From the pop-up menu, select **PySpark**.
7. Click **Create Project**. Your new project displays sample files.
8. Click **Open Workbench**.
9. Select the **Python 2** engine.
10. Click **Launch Session**.

Example: Montecarlo Estimation

Within the template PySpark project, `pi.py` is a classic example that calculates Pi using the Montecarlo Estimation.

What follows is the full, annotated code sample that can be saved to the `pi.py` file.

```
# # Estimating $\pi$
#
# This PySpark example shows you how to estimate $\pi$ in parallel
# using Monte Carlo integration.

from __future__ import print_function
import sys
from random import random
from operator import add
# Connect to Spark by creating a Spark session
from pyspark.sql import SparkSession

spark = SparkSession\
    .builder\
    .appName("PythonPi")\
    .getOrCreate()

partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
n = 100000 * partitions

def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 < 1 else 0

# To access the associated SparkContext
count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
print("Pi is roughly %f" % (4.0 * count / n))

spark.stop()
```

Example: Reading Data from HDFS (Wordcount)

In this PySpark example, Cloudera Data Science Workbench reads in data from the default configured filesystem, HDFS.

`Wordcount.py`

```
# Word count
#
# This example shows how to count the occurrences of each word in a text file.

from __future__ import print_function
import sys, re
from operator import add
from pyspark.sql import SparkSession

spark = SparkSession\
  .builder\
  .appName("PythonWordCount")\
  .getOrCreate()

# The file you use as input must already exist in HDFS.
# Add the data file to hdfs.
!hdfs dfs -put resources/cgroup-v2.txt /tmp

# Access the file from wordcount.py
lines = spark.read.text("/tmp/cgroup-v2.txt").rdd.map(lambda r: r[0])
counts = lines.flatMap(lambda x: x.split(' ')) \
  .map(lambda x: (x, 1)) \
  .reduceByKey(add) \
  .sortBy(lambda x: x[1], False)
output = counts.collect()
for (word, count) in output:
  print("%s: %i" % (word, count))

spark.stop()
```

Note:

- Since HDFS is the configured filesystem, Spark jobs on Cloudera Data Science Workbench read from HDFS by default.
- The file you read must already exist in HDFS. You can call out to the shell from within Python sessions by prefixing an exclamation mark (!) to the command. For example, if you want to load the `cgroup-v2.txt` file in HDFS, you can invoke the HDFS CLI by running `!hdfs` in the code as follows.

```
!hdfs dfs -put resources/cgroup-v2.txt /tmp
```

Example: Locating and Adding JARs to Spark 2 Configuration

This example shows how to discover the location of JAR files installed with Spark 2, and add them to the Spark 2 configuration.

```
# # Using Avro data
#
# This example shows how to use a JAR file on the local filesystem on
# Spark on Yarn.

from __future__ import print_function
import os,sys
import os.path
from functools import reduce
from pyspark.sql import SparkSession
from pyspark.files import SparkFiles
```

```
# Add the data file to HDFS for consumption by the Spark executors.
!hdfs dfs -put resources/users.avro /tmp

# Find the example JARs provided by the Spark parcel. This parcel
# is available on both the driver, which runs in Cloudera Data Science Workbench, and
the
# executors, which run on Yarn.
exampleDir = os.path.join(os.environ["SPARK_HOME"], "examples/jars")
exampleJars = [os.path.join(exampleDir, x) for x in os.listdir(exampleDir)]

# Add the Spark JARs to the Spark configuration to make them available for use.
spark = SparkSession\
    .builder\
    .config("spark.jars", ",".join(exampleJars))\
    .appName("AvroKeyInputFormat")\
    .getOrCreate()
sc = spark.sparkContext

# Read the schema.
schema = open("resources/user.avsc").read()
conf = {"avro.schema.input.key": schema }

avro_rdd = sc.newAPIHadoopFile(
    "/tmp/users.avro", # This is an HDFS path!
    "org.apache.avro.mapreduce.AvroKeyInputFormat",
    "org.apache.avro.mapred.AvroKey",
    "org.apache.hadoop.io.NullWritable",
   keyConverter="org.apache.spark.examples.pythonconverters.AvroWrapperToJavaConverter",

    conf=conf)
output = avro_rdd.map(lambda x: x[0]).collect()
for k in output:
    print(k)
spark.stop()
```

Example: Distributing Dependencies on a PySpark Cluster

Although Python is a popular choice for data scientists, it is not straightforward to make a Python library available on a distributed PySpark cluster. To determine which dependencies are required on the cluster, you must understand that Spark code applications run in Spark executor processes distributed throughout the cluster. If the Python code you are running uses any third-party libraries, Spark executors require access to those libraries when they run on remote executors.

This example demonstrates a way to run the following Python code (`nltk_sample.py`), that includes pure Python libraries (nltk), on a distributed PySpark cluster.

`nltk_sample.py`

```
# This code uses NLTK, a Python natural language processing library.
# NLTK is not installed with conda by default.
# You can use 'import' within your Python UDFs, to use Python libraries.
# The goal here is to distribute NLTK with the conda environment.

import os
import sys
from pyspark.sql import SparkSession

spark = SparkSession.builder \
          .appName("spark-nltk") \
          .getOrCreate()

data = spark.sparkContext.textFile('1970-Nixon.txt')

def word_tokenize(x):
    import nltk
    return nltk.word_tokenize(x)

def pos_tag(x):
    import nltk
    return nltk.pos_tag([x])

words = data.flatMap(word_tokenize)
words.saveAsTextFile('nixon_tokens')

pos_word = words.map(pos_tag)
pos_word.saveAsTextFile('nixon_token_pos')
```

1. Pack the Python environment into conda.

```
conda create -n nltk_env --copy -y -q python=2.7.11 nltk numpy
```

   The `--copy` option allows you to copy whole dependent packages into certain directory of a conda environment. If you want to add extra pip packages without conda, you should copy packages manually after using `pip install`. In Cloudera Data Science Workbench, pip will install packages into `~/.local`.

```
pip install some-awesome-package
cp -r ~/.local/lib ~/.conda/envs/nltk_env/
```

   Zip the conda environment for shipping on PySpark cluster.

```
cd ~/.conda/envs
zip -r ../../nltk_env.zip nltk_env
```

2. (Specific to NLTK) For this example, you can use NLTK data as input.

```
cd ~/
source activate nltk_env

# download nltk data
(nltk_env)$ python -m nltk.downloader -d nltk_data all
(nltk_env)$ hdfs dfs -put nltk_data/corpora/state_union/1970-Nixon.txt ./

# archive nltk data for distribution
cd ~/nltk_data/tokenizers/
zip -r ../../tokenizers.zip *
cd ~/nltk_data/taggers/
zip -r ../../taggers.zip *
```

3. Set `spark-submit` options in `spark-defaults.conf`.

```
spark.yarn.appMasterEnv.PYSPARK_PYTHON=./NLTK/nltk_env/bin/python
spark.yarn.appMasterEnv.NLTK_DATA=./
spark.executorEnv.NLTK_DATA=./
spark.yarn.dist.archives=nltk_env.zip#NLTK,tokenizers.zip#tokenizers,taggers.zip#taggers
```

   With these settings, PySpark unzips `nltk_env.zip` into the NLTK directory. NLTK_DATA is the environmental variable where NLTK data is stored.

4. Set the PYSPARK_PYTHON environment variable in Cloudera Data Science Workbench. To set this, go to the project page and click **Settings** > **Engine** > **Environment Variables**. Set PYSPARK_PYTHON to `./NLTK/nltk_env/bin/python` and click **Save Environment**.
5. Restart your project session and run the `nltk_sample.py` script in the workbench. You can test whether the script ran successfully using the following command:

```
!hdfs dfs -cat ./nixon_tokens/* | head -n 20
Annual
Message
to
the
Congress
on
the
State
```

```
of
the
Union
.
January
22
,
1970
Mr.
Speaker
,
Mr.

! hdfs dfs -cat nixon_token_pos/* | head -n 20
[(u'Annual', 'JJ')]
[(u'Message', 'NN')]
[(u'to', 'TO')]
[(u'the', 'DT')]
[(u'Congress', 'NNP')]
[(u'on', 'IN')]
[(u'the', 'DT')]
[(u'State', 'NNP')]
[(u'of', 'IN')]
[(u'the', 'DT')]
[(u'Union', 'NN')]
[(u'.', '.')]
[(u'January', 'NNP')]
[(u'22', 'CD')]
[(u',', ',')]
[(u'1970', 'CD')]
[(u'Mr.', 'NNP')]
[(u'Speaker', 'NN')]
[(u',', ',')]
[(u'Mr.', 'NNP')]
```

Using Spark 2 from R

R users can access Spark 2 using sparklyr. Although Cloudera does not ship or support sparklyr, we do recommend using sparklyr as the R interface for Cloudera Data Science Workbench.

Installing sparklyr

Install the latest version of sparklyr as follows.

```
install.packages("sparklyr")
```

Connecting to Spark 2

You can connect to local instances of Spark 2 as well as remote clusters.

```
## Connecting to Spark 2
# Connect to an existing Spark 2 cluster in YARN client mode using the spark_connect
# function.
library(sparklyr)
system.time(sc <- spark_connect(master = "yarn-client"))
# The returned Spark 2 connection (sc) provides a remote dplyr data source to the Spark
 2 cluster.
```

Using Spark 2 from Scala

This topic describes how to set up a Scala project for Cloudera Distribution of Apache Spark 2 along with a few associated tasks. Cloudera Data Science Workbench provides an interface to the Spark 2 shell (v 2.0+) that works with Scala 2.11.

Setting Up a Scala Project

1. Open Cloudera Data Science Workbench.
2. Select **New Project**.
3. Enter a **Project Name**.
4. Choose whether the **Project Visibility** is *Private* or *Public*.
5. Under **Initial Setup**, choose the **Template** tab.
6. Select **Scala**.
7. Click **Create Project**. Your new project displays sample files.
8. Click **Open Workbench**.
9. Select **Scala** engine.
10. Click **Launch Session**.

The Scala engine typically takes 30 seconds to become ready. This increased startup time is due to the Scala engine automatically creating a Spark context in Apache YARN.

The examples that follow are included under the Scala templates.

Getting Started and Accessing Spark 2

Unlike PySpark or Sparklyr, you can access a SparkContext assigned to the *spark* (SparkSession) and *sc* (SparkContext) objects on console startup, just as when using the Spark shell. By default, the application name will be set to CDSW_*sessionID*, where sessionId is the id of the session running your Spark code. To customize this, set the `spark.app.name` property to the desired application name in a `spark-defaults.conf` file.

`Pi.scala` is a classic starting point for calculating Pi using Montecarlo Estimation.

This is the full, annotated code sample.

```
// Calculate pi with Monte Carlo estimation
import scala.math.random

// Make a very large unique set of 1 -> n
val partitions = 2
val n = math.min(100000L * partitions, Int.MaxValue).toInt
val xs = 1 until n

// Split n into the number of partitions we can use
val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")

// Generate a random set of points within a 2x2 square
val sample = rdd.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  (x, y)
}.setName("'Random points rdd'")

// points w/in the square also w/in the center circle of r=1
val inside = sample
    .filter { case (x, y) => (x * x + y * y < 1) }
    .setName("'Random points inside circle'")

val count = inside.count()

// Area(circle)/Area(square) = inside/n => pi=4*inside/n
println("Pi is roughly " + 4.0 * count / n)
```

Key points to note:

- `import scala.math.random`

  Importing included packages works just as in the shell, and need only be done once.

- **Spark context (*sc*).**

You can access a SparkContext assigned to the variable *sc* on console startup.

```
val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")
```

Example: Reading Data from HDFS (Wordcount)

Since HDFS is the configured filesystem, Spark jobs on Cloudera Data Science Workbench read from HDFS by default.

The file you use must already exist in HDFS. For example, you might load the `jedi_wisdom.txt` file using the terminal. Click the **Terminal** link above your Cloudera Data Science Workbench console and enter the following command:

```
hdfs dfs -put data/jedi_wisdom.txt /tmp
```

This example reads data from HDFS.

`wordcount.scala`:

```
//count lower bound
val threshold = 2

// this file must already exist in hdfs, add a
// local version by dropping into the terminal.
val tokenized = sc.textFile("/tmp/data/jedi_wisdom.txt").flatMap(_.split(" "))

// count the occurrence of each word
val wordCounts = tokenized.map((_ , 1)).reduceByKey(_ + _)

// filter out words with fewer than threshold occurrences
val filtered = wordCounts.filter(_._2 >= threshold)

System.out.println(filtered.collect().mkString(","))
```

Click **Run**.

Key points to note:

- You add files to HDFS by opening the terminal and running the following command:

```
hdfs dfs -put data/jedi_wisdom.txt /tmp
```

- You access the file from Wordcount scala using the following command:

```
sc.textFile("tmp/jedi_wisdom.txt")
```

Example: Read Files from the Cluster Local Filesystem

Use the following command in the terminal to read text from the local filesystem. The file must exist on all nodes, and the same path for the driver and executors. In this example you are reading the file `ebay-xbox.csv`.

```
sc.textFile(“file:///tmp/ebay-xbox.csv”)
```

Example: Using External Packages by Adding Jars or Dependencies

External libraries are handled through line magics. Line magics in the Toree kernel are prefixed with *%*.

Adding Remote Packages

You can use Apache Toree's `AddDeps` magic to add dependencies from Maven central. You must specify the company name, artifact ID, and version. To resolve any transitive dependencies, you must explicitly specify the `--transitive` flag.

```
%AddDeps org.scalaj scalaj-http_2.11 2.3.0
import scalaj.http._
val response: HttpResponse[String] = Http("http://www.omdbapi.com/").param("t","crimson
 tide").asString
response.body
response.code
response.headers
response.cookies
```

Adding Remote or Local JARs

You can use the `AddJars` magic to distribute local or remote JARs to the kernel and the cluster. Using the *-f* option ignores cached JARs and reloads.

```
%AddJar http://example.com/some_lib.jar -f
%AddJar file:/path/to/some/lib.jar
```

Setting Up an HTTP Proxy for Spark 2

In Cloudera Data Science Workbench clusters that use an HTTP proxy, follow these steps to support web-related actions in Spark. You must set the Spark configuration parameter `extraJavaOptions` on your gateway nodes.

To set up a Spark proxy:

1. Log in to Cloudera Manager.
2. Go to **Spark2** > **Configuration**.
3. Filter the properties with **Scope** > **Gateway** and **Category** > **Advanced**.
4. Scroll down to **Spark 2 Client Advanced Configuration Snippet (Safety Valve) for spark2-conf/spark-defaults.conf**.
5. Enter the following configuration code, substituting your proxy host and port values:

```
spark.driver.extraJavaOptions= \
-Dhttp.proxyHost=<YOUR HTTP PROXY HOST> \
-Dhttp.proxyPort=<HTTP PORT> \
-Dhttps.proxyHost=<YOUR HTTPS PROXY HOST> \
-Dhttps.proxyPort=<HTTPS PORT>
```

6. Click **Save Changes**.
7. Choose **Actions** > **Deploy Client Configuration**.

Cloudera Data Science Workbench Administration Guide

This topic describes how to configure and manage a Cloudera Data Science Workbench deployment as a site administrator. By default, the first user account that signs up for the Cloudera Data Science Workbench becomes a site administrator. Site administrators can manage other users, monitor resources, secure access to the deployment, and upload license keys for the product.

Important: Site administrators have complete access to *all* activity on the deployment. This includes access to all teams and projects on the deployment, even if they have not been explicitly added as team members or collaborators.

To access the site administrator dashboard:

1. Go to the Cloudera Data Science Workbench web application (`http://cdsw.`*`company`*`.com`) and log in as a site administrator.
2. On the left sidebar, click **Admin**. You will see an array of tabs for all the tasks you can perform as a site administrator.

The rest of this topic describes some common tasks for a Cloudera Data Science Workbench site administrator.

Related Topics:

Managing Users

As a site administrator you can add new users, assign or modify privileges for existing users, and monitor user activity on the Cloudera Data Science Workbench deployment.

Adding New Users

To invite new users, navigate to the **Admin** > **Users** tab. Under Invitations, enter the name or email ID of the person you want to invite and click **Invite**. This tab will show you a list of all outstanding invitations. Once an invitation has been accepted, the record will no longer show up on this page. The **Users** tab also displays a list of users of the application. Click on a username to see more details about the user.

If you want new users to join by invitation only, go to the **Admin** > **Settings** tab and check the **Require invitation to sign up** checkbox to require invitation tokens for account creation. By default, invitations are sent from noreply@*your-cdsw-domain*. To modify this default, see Setting up Email Notifications on page 90.

Assigning the Site Administrator role to an Existing User

To make a regular user a site administrator:

1. Sign in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Click the **Users** tab.
4. Click on the username of the user who you want to make a site administrator.
5. Select the **Site Administrator** checkbox.
6. Click **Update**.

Disabling User Accounts

Use the following instructions to disable user accounts. Note that disabled users cannot login and do not count towards named users for licensing.

1. Sign in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.

3. Click the **Users** tab.
4. Click on the username of the user who you want to disable.
5. Select the **Disabled** checkbox.
6. Click **Update**.

Monitoring Users

The **Users** tab on the admin dashboard displays the complete list of users. You can see which users are currently active, and when a user last logged in to the Cloudera Data Science Workbench. To modify a user's username, email or permissions, click the **Edit** button under the Action column.

Monitoring Site Usage

The **Admin** > **Overview** tab displays basic information about your deployment, such as the number of users signed up, the number of teams and projects created, memory used, and some average job scheduling and run times. You can also see the version of Cloudera Data Science Workbench you are currently running.

The **Admin** > **Activity** tab of the dashboard displays the following time series charts. These graphs should help site administrators identify basic usage patterns, understand how cluster resources are being utilized over time, and how they are being distributed among teams and users.

- **CPU** - Total number of CPUs requested by sessions running at this time.
- **Memory** - Total amount of memory (in GiB) requested by sessions running at this time.
- **GPU** - Total Number of GPUs requested by sessions running at this time.
- **Runs** - Total number of sessions and jobs running at this time.
- **Lag** - Depicts session scheduling and startup times.
  - **Scheduling Duration:** The amount of time it took for a session pod to be scheduled on the cluster.
  - **Starting Duration:** The amount of time it took for a session to be ready for user input. This is the amount of time since a pod was scheduled on the cluster until code could be executed.

Managing Engine Profiles

On the **Admin** > **Engines** page, under **Engines Profiles**, you can provide default engine profiles to users. These engine profiles define the default vCPU, GPU, and memory configurations for sessions and jobs. Cloudera recommends that all profiles include at least 2 GB of RAM to avoid out of memory errors for common user operations.

You will see the option to add GPUs to the engine profiles only if your Cloudera Data Science Workbench nodes are equipped with GPUs, and you have enabled them for usage by setting the relevant properties in `cdsw.conf`.

Adding Custom Engine Images

Cloudera Data Science Workbench site administrators can add libraries and other dependencies to the Docker image in which their engines run. You can whitelist images for use in projects on the **Admin** > **Engines** page, under the **Engine Images** section. Currently, Cloudera Data Science Workbench only supports *public* Docker images in registries accessible to the Cloudera Data Science Workbench nodes.

Project administrators will need to explicitly select which of these white-listed images is installed for their projects. For an example on how to add more libraries and dependencies to your Docker image, see Creating Extended Engine Images on page 97.

Customizing the Engine Environment

On the **Admin** > **Engines** page, go to the **Environmental Variables** section to provide values for global environment variables that must be injected into all engines across the deployment. Environmental variables set here at the global level by site administrators can be overridden by per-project values set by project administrators. For a list of environmental variables that can be set in every engine, see Project Environment Variables on page 60.

Non-standard CDH Parcel Location

By default, Cloudera Data Science Workbench looks for the CDH parcel at `/opt/cloudera/parcels`. If your CDH parcel is at another location, add that path to the Parcel directory section.

Mounts

You can use this section to specify folders that should be mounted through to all the engines. Cloudera Data Science Workbench will automatically mount CDH parcels and client configuration.

Note that all Cloudera Data Science Workbench projects run inside a Docker container. If your project is referencing any files/folders on the host, you must use this option to explicitly load them into the container.

Setting Time Zones for Sessions

The default time zone for Cloudera Data Science Workbench sessions is UTC. This is the default regardless of the time zone setting on the Master node.

To change to your preferred time zone, for example, Pacific Standard Time (PST), navigate to **Admin** > **Engines**. Under the **Environmental Variables** section, add a new variable with the name set to TZ and value set to `America/Los_Angeles`, and click **Add**.

Configuring External Authentication

Cloudera Data Science Workbench supports external authentication using LDAP and SAML protocols. You can configure LDAP or SAML authentication under the **Admin** > **Security** tab by following the instructions at Configuring External Authentication.

Setting up Email Notifications

Go to the **Admin** > **Settings** tab to specify an email address for outbound invitations and job notifications.

By default, all emails are sent from noreply@*your-cdsw-domain*. However, if your SMTP domain is different from the Cloudera Data Science Workbench domain, or it does not allow spoofing, you will need to explicitly specify the email address at the No Reply Email field.

Currently, Cloudera Data Science Workbench only sends email notifications when you add teammates to a project, not when you create a new project. Email preferences cannot currently be configured at an individual user level.

Managing License Keys

Cloudera Data Science Workbench is subject to the same license and subscription model as Cloudera Enterprise. To upload a license key, go to the **Admin** > **License** tab. For details on the types of licenses and detailed instructions for how to upload a new license key, see Managing License Keys for Cloudera Data Science Workbench on page 91.

Disabling Analytics Tracking

To help improve the product, Cloudera Data Science Workbench by default collects aggregate usage data by sending limited tracking events to Google Analytics and Cloudera servers. No customer data or personal information is sent as part of these bundles. To disable analytics tracking, go to **Admin** > **Settings** and uncheck the **Send usage data to Cloudera** checkbox.

For more details on the usage and diagnostic data collected by Cloudera Data Science Workbench, see Data Collection in Cloudera Data Science Workbench on page 101.

Managing License Keys for Cloudera Data Science Workbench

Cloudera Data Science Workbench requires a Cloudera Enterprise license. To obtain a Cloudera Enterprise license, either fill in this form, or call 866-843-7207. Note that only one license key can be used at a time.

After an initial trial period of 60 days, you must upload a license key to continue to use Cloudera Data Science Workbench.

Trial License

Cloudera Data Science Workbench is fully functional during a 60-day, non-renewable trial period. The trial period starts when you create your first user.

If 60 days or fewer remain on the license, a badge in the lower left corner of the dashboard displays the number of days remaining. The initial trial period is 60 days, so the remaining days are always displayed during the trial period.

When a trial license expires, functionality is limited as follows:

- A warning banner notifies you that the license has expired and suggests you contact the site administrator or upload a license key.
- You cannot create new users, projects, sessions, or jobs.
- Existing users can log in and view their projects and files.
- You cannot run existing jobs.

Cloudera Enterprise License

When an Enterprise license expires, a warning banner displays, but all product features remain fully functional.

Contact Cloudera Support to receive an updated license.

Uploading License Keys

To upload the license key:

1. Go to **Admin** > **License**.
2. Click **Upload License**.
3. Select the license file to be uploaded and click **Upload**.

Using GPUs for Cloudera Data Science Workbench Workloads

A GPU is a specialized processor that can be used to accelerate highly parallelized computationally-intensive workloads. Because of their computational power, GPUs have been found to be particularly well-suited to deep learning workloads. Ideally, CPUs and GPUs should be used in tandem for data engineering and data science workloads. A typical machine learning workflow involves data preparation, model training, model scoring, and model fitting. You can use existing general-purpose CPUs for each stage of the workflow, and optionally accelerate the math-intensive steps with the selective application of special-purpose GPUs. For example, GPUs allow you to accelerate model fitting using frameworks such as Tensorflow, PyTorch, Keras, MXNet, and Microsoft Cognitive Toolkit (CNTK).

By enabling GPU support, data scientists can share GPU resources available on Cloudera Data Science Workbench nodes. Users can requests a specific number of GPU instances, up to the total number available on a node, which are then allocated to the running session or job for the duration of the run. Projects can use isolated versions of libraries, and even different CUDA and cuDNN versions via Cloudera Data Science Workbench's extensible engine feature.

Prerequisite

GPU support for workloads was added in Cloudera Data Science Workbench 1.1.0. The rest of this topic assumes you have already installed or upgraded to the latest version of Cloudera Data Science Workbench.

Key Points to Note

- Cloudera Data Science Workbench only supports CUDA-enabled NVIDIA GPU cards.
- Cloudera Data Science Workbench does not support heterogeneous GPU hardware in a single deployment.
- Cloudera Data Science Workbench does not include an engine image that supports NVIDIA libraries. Create your own custom CUDA-capable engine image using the instructions described in this topic.
- Cloudera Data Science Workbench also does not install or configure the NVIDIA drivers on the Cloudera Data Science Workbench gateway nodes. These depend on your GPU hardware and will have to be installed by your system administrator. The steps provided in this topic are generic guidelines that will help you evaluate your setup.
- The instructions described in this topic require Internet access. If you have an airgapped deployment, you will be required to manually download and load the resources onto your nodes.
- For a list of known issues associated with this feature, refer Known Issues - GPU Support on page 25.

Enabling Cloudera Data Science Workbench to use GPUs

To enable GPU usage on Cloudera Data Science Workbench, perform the following steps to provision the Cloudera Data Science Workbench gateway hosts and install Cloudera Data Science Workbench on these hosts. As noted in the following instructions, certain steps must be repeated on all gateway nodes that have GPU hardware installed on them.

Set Up the Operating System and Kernel

Use the following commands to update and reboot your system.

```
sudo yum update -y
sudo reboot
```

Install the Development Tools and `kernel-devel` packages.

```
sudo yum groupinstall -y "Development tools"
sudo yum install -y kernel-devel-`uname -r`
```

Perform these steps on all nodes with GPU hardware installed on them.

Install the NVIDIA Driver on GPU Nodes

Cloudera Data Science Workbench does not ship with any of the NVIDIA drivers needed to enable GPUs for general purpose processing. System administrators are expected to install the version of the drivers that are compatible with the CUDA libraries that will be consumed on each node. Use the NVIDIA UNIX Driver archive to find out which driver is compatible with your GPU cards.

Important:

Cloudera Data Science Workbench has been tested against NVIDIA's driver version 381.22, and it has shown to work well with deep learning frameworks such as Pytorch, Tensorflow, and Keras. For a list of all the GPU devices that support version 381.22, see NVIDIA Supported Chips - 381.22.

Use the following commands to download and install the NVIDIA driver for the GPU card you are using. The version of the driver depends on the GPU and the operating system in use. It is crucial that you download the correct version. To install the NVIDIA driver, follow the instructions on the respective driver's download page. For instance, if you use the `.run` file method (Linux 64 bit), you can download and install the driver with the following sample commands. Modify the value for the `NVIDIA_DRIVER_VERSION` parameter as needed.

```
wget http://us.download.nvidia.com/.../NVIDIA-Linux-x86_64-<driver_version>.run
export NVIDIA_DRIVER_VERSION=<driver_version>
chmod 755 ./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run
./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run -asq
```

Once the installation is complete, run the following command to verify that the driver was installed correctly:

```
/usr/bin/nvidia-smi
```

Perform this step on all nodes with GPU hardware installed on them.

Enable Docker NVIDIA Volumes on GPU Nodes

To enable Docker containers to use the GPUs, the previously installed NVIDIA driver libraries must be consolidated in a single directory named after the *<driver_version>*, and mounted into the containers. This is done using the `nvidia-docker` package, which is a thin wrapper around the Docker CLI and a Docker plugin.

The following sample steps demonstrate how to use `nvidia-docker` to set up the directory structure for the drivers so that they can be easily consumed by the Docker containers that will leverage the GPU. *Perform these steps on all nodes with GPU hardware installed on them.*

1. Download and install `nvidia-docker`. Use a version that is suitable for your deployment.

```
wget
https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.1/nvidia-docker-1.0.1-1.x86_64.rpm
sudo yum install -y nvidia-docker-1.0.1-1.x86_64.rpm
```

2. Start the necessary services and plugins:

```
systemctl start nvidia-docker
systemctl enable nvidia-docker
```

3. Run a small container to create the Docker volume structure:

```
sudo nvidia-docker run --rm nvidia/cuda:8.0 nvidia-smi
```

4. Verify that the `/var/lib/nvidia-docker/volumes/nvidia_driver/$NVIDIA_DRIVER_VERSION/` directory was created.

5. Use the following Docker command to verify that Cloudera Data Science Workbench can access the GPU.

```
sudo docker run --net host \
    --device=/dev/nvidiactl \
    --device=/dev/nvidia-uvm \
    --device=/dev/nvidia0 \
    -v
/var/lib/nvidia-docker/volumes/nvidia_driver/$NVIDIA_DRIVER_VERSION/:/usr/local/nvidia/
 \
    -it nvidia/cuda:8.0 \
    /usr/local/nvidia/bin/nvidia-smi
```

On a multi-GPU machine the output of this command will show exactly one GPU. This is because we have run this sample Docker container with only one device (`/dev/nvidia0`).

Important: Cloudera Data Science Workbench does not detect GPUs after a machine reboot. This is because certain NVIDIA modules do not load automatically after a reboot. Review the associated Known Issue and workaround here.

Enable GPU Support in Cloudera Data Science Workbench

Minimum Required Cloudera Manager Role: Cluster Administrator

Depending on your deployment, use one of the following sets of steps to enable Cloudera Data Science Workbench to identify the GPUs installed:

CSD Deployments

1. Go to the CDSW service in Cloudera Manager. Click **Configuration**. Set the following parameters as directed in the following table.

Enable GPU Support	Use the checkbox to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a node that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.
NVIDIA GPU Driver Library Path	Complete path to the NVIDIA driver libraries. In this example, the path would be, `/var/lib/nvidia-docker/volumes/nvidia_driver/`*`$NVIDIA_DRIVER_VERSION`*`/`

2. Restart the CDSW service in Cloudera Manager.
3. Test whether Cloudera Data Science Workbench is detecting GPUs.

RPM Deployments

1. Set the following parameters in `/etc/cdsw/config/cdsw.conf` on *all* Cloudera Data Science Workbench nodes. You must make sure that `cdsw.conf` is consistent across all nodes, irrespective of whether they have GPU hardware installed on them.

`NVIDIA_GPU_ENABLE`	Set this property to `true` to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a node that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.
`NVIDIA_LIBRARY_PATH`	Complete path to the NVIDIA driver libraries. In this example, the path would be, `"/var/lib/nvidia-docker/volumes/nvidia_driver/`*`$NVIDIA_DRIVER_VERSION`*`/"`

2. On the *master* node, run the following command to restart Cloudera Data Science Workbench.

```
cdsw restart
```

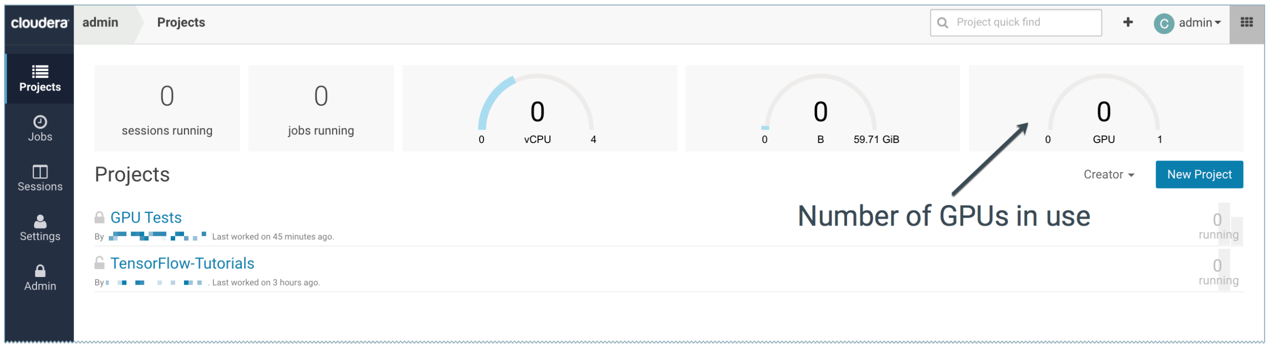
If you modified `cdsw.conf` on a *worker* node, run the following commands to make sure the changes go into effect:

```
cdsw reset
cdsw join
```

3. Use the following section to test whether Cloudera Data Science Workbench can now detect GPUs.

Test whether Cloudera Data Science Workbench can detect GPUs

Once Cloudera Data Science Workbench has successfully restarted, if NVIDIA drivers have been installed on the Cloudera Data Science Workbench hosts, Cloudera Data Science Workbench will now be able to detect the GPUs available on its hosts.



Additionally, the output of this command will also indicate that there are nodes with GPUs present.

```
cdsw status
```

Create a Custom CUDA-capable Engine Image

The base Ubuntu 16.04 engine image (`docker.repository.cloudera.com/cdsw/engine:2`) that ships with Cloudera Data Science Workbench will need to be extended with CUDA libraries to make it possible to use GPUs in jobs and sessions.

The following sample Dockerfile illustrates an engine on top of which machine learning frameworks such as Tensorflow and PyTorch can be used. This Dockerfile uses a deep learning library from NVIDIA called NVIDIA CUDA Deep Neural Network (cuDNN). Make sure you check with the machine learning framework that you intend to use in order to know which version of cuDNN is needed. As an example, Tensorflow uses CUDA 8.0 and requires cuDNN 6.0.

The following sample Dockerfile uses NVIDIA's official Dockerfiles for CUDA and cuDNN images.

cuda.Dockerfile

```
FROM  docker.repository.cloudera.com/cdsw/engine:2

RUN NVIDIA_GPGKEY_SUM=d1be581509378368edeec8c1eb2958702feedf3bc3d17011adbf24efacce4ab5
 && \
    NVIDIA_GPGKEY_FPR=ae09fe4bbd223a84b2ccfce3f60f4b3d7fa2af80 && \
    apt-key adv --fetch-keys
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub
 && \
    apt-key adv --export --no-emit-version -a $NVIDIA_GPGKEY_FPR | tail -n +5 >
cudasign.pub && \
   echo "$NVIDIA_GPGKEY_SUM  cudasign.pub" | sha256sum -c --strict - && rm cudasign.pub
 && \
   echo "deb http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64
 /" > /etc/apt/sources.list.d/cuda.list

ENV CUDA_VERSION 8.0.61
LABEL com.nvidia.cuda.version="${CUDA_VERSION}"

ENV CUDA_PKG_VERSION 8-0=$CUDA_VERSION-1
RUN apt-get update && apt-get install -y --no-install-recommends \
        cuda-nvrtc-$CUDA_PKG_VERSION \
        cuda-nvgraph-$CUDA_PKG_VERSION \
        cuda-cusolver-$CUDA_PKG_VERSION \
        cuda-cublas-8-0=8.0.61.2-1 \
        cuda-cufft-$CUDA_PKG_VERSION \
        cuda-curand-$CUDA_PKG_VERSION \
        cuda-cusparse-$CUDA_PKG_VERSION \
        cuda-npp-$CUDA_PKG_VERSION \
        cuda-cudart-$CUDA_PKG_VERSION && \
    ln -s cuda-8.0 /usr/local/cuda && \
    rm -rf /var/lib/apt/lists/*
```

```
RUN echo "/usr/local/cuda/lib64" >> /etc/ld.so.conf.d/cuda.conf && \
    ldconfig

RUN echo "/usr/local/nvidia/lib" >> /etc/ld.so.conf.d/nvidia.conf && \
    echo "/usr/local/nvidia/lib64" >> /etc/ld.so.conf.d/nvidia.conf

ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64

RUN echo "deb
http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64
/" > /etc/apt/sources.list.d/nvidia-ml.list

ENV CUDNN_VERSION 6.0.21
LABEL com.nvidia.cudnn.version="${CUDNN_VERSION}"

RUN apt-get update && apt-get install -y --no-install-recommends \
            libcudnn6=$CUDNN_VERSION-1+cuda8.0 && \
    rm -rf /var/lib/apt/lists/*
```

You can now build a custom engine image out of `cuda.Dockerfile` using the following sample command:

```
docker build --network host -t <company-registry>/cdsw-cuda:2 . -f cuda.Dockerfile
```

Push this new engine image to a public Docker registry so that it can be made available for Cloudera Data Science Workbench workloads. For example:

```
docker push <company-registry>/cdsw-cuda:2
```

Allocate GPUs for Sessions and Jobs

Once Cloudera Data Science Workbench has been enabled to use GPUs, a site administrator must whitelist the CUDA-capable engine image created in the previous step. Site administrators can also set a limit on the maximum number of GPUs that can be allocated per session or job.

1. Sign in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Go to the **Engines** tab.
4. From the **Maximum GPUs per Session/Job** dropdown, select the maximum number of GPUs that can be used by an engine.
5. Under **Engine Images**, add the custom CUDA-capable engine image created in the previous step. This whitelists the image and allows project administrators to use the engine in their jobs and sessions.
6. Click **Update**.

Project administrators can now whitelist the CUDA engine image to make it available for sessions and jobs within a particular project.

1. Navigate to the project's **Overview** page.
2. Click **Settings**.
3. Go to the **Engines** tab.
4. Under **Engine Image**, select the CUDA-capable engine image from the dropdown.

Note: Cloudera Data Science Workbench **1.2.0** projects need additional configuration to ensure that CUDA libraries are accessible by the CUDA engine. Review the associated Known Issue and workaround here. This issue has been fixed in Cloudera Data Science Workbench 1.2.1.

Example: Tensorflow

This is a simple example that walks you through a simple Tensorflow workload that uses GPUs.

1. Open the workbench console and start a Python engine.
2. Install Tensorflow.

   Python 2

```
!pip install tensorflow-gpu
```

   Python 3

```
!pip3 install tensorflow-gpu
```

3. Restart the session. This requirement of a session restart after installation is a known issue specific to Tensorflow.
4. Create a new file with the following sample code. The code first performs a multiplication operation and prints the session output, which should mention the GPU that was used for the computation. The latter half of the example prints a list of all available GPUs for this engine.

```
import tensorflow as tf
a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
c = tf.matmul(a, b)

# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

# Runs the operation.
print(sess.run(c))

# Prints a list of GPUs available
from tensorflow.python.client import device_lib
def get_available_gpus():
    local_device_protos = device_lib.list_local_devices()
    return [x.name for x in local_device_protos if x.device_type == 'GPU']

print get_available_gpus()
```

Creating Extended Engine Images

Cloudera Data Science Workbench site administrators and project administrators can add libraries and other dependencies to the Docker image in which their engines run. Site administrators can whitelist specific images for use in projects, and project administrators can select which of these white-listed images is installed for their projects.

Note: Currently, Cloudera Data Science Workbench does not support pulling images from registries that require Docker credentials.

Use the following basic MeCab example as a guide on how you can extend the Cloudera Data Science Workbench base engine image to include the libraries you want.

Related Resources:

- This Cloudera Engineering Blog post on Customizing Docker Images in Cloudera Data Science Workbench describes an end-to-end example on how to build and publish a customized Docker image and use it as an engine in Cloudera Data Science Workbench.
- For an example of how to extend the base engine image to include Conda, see Creating an Extensible Engine With Conda on page 60.

Example: MeCab

The following Dockerfile shows how to add [MeCab](), a Japanese text tokenizer, to the base Cloudera Data Science Workbench engine.

```
# Dockerfile

FROM docker.repository.cloudera.com/cdsw/engine:3
RUN apt-get update && \
    apt-get install -y -q mecab \
                          libmecab-dev \
                          mecab-ipadic-utf8 && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN cd /tmp && \
    git clone --depth 1 https://github.com/neologd/mecab-ipadic-neologd.git && \
    /tmp/mecab-ipadic-neologd/bin/install-mecab-ipadic-neologd -y -n -p
/var/lib/mecab/dic/neologd && \
    rm -rf /tmp/mecab-ipadic-neologd
RUN pip install --upgrade pip
RUN pip install mecab-python==0.996
```

To use this image on your Cloudera Data Science Workbench project, perform the following steps.

1. Build a new image with the Dockerfile.

```
docker build --network=host -t <company-registry>/user/cdsw-mecab:latest . -f Dockerfile
```

2. Push the image to your company's Docker registry.

```
docker push <company-registry>/user/cdsw-mecab:latest
```

3. Whitelist the image, `<company-registry>/user/cdsw-mecab:latest`. Only a site administrator can do this.
   a. Log in as a site administrator.
   b. Click **Admin**.
   c. Go to the **Engines** tab.
   d. Add `<company-registry>/user/cdsw-mecab:latest` to the list of whitelisted engine images.
4. Make the whitelisted image available to your project. Only a project administrator can do this.
   a. Go to the project **Settings** page.
   b. Click **Engines**.
   c. Select `company-registry/user/cdsw-mecab:latest` from the dropdown list of available Docker images. Sessions and jobs you run in your project will now have access to this custom image.

Managing Cloudera Data Science Workbench Hosts

This topic describes how to add and remove hosts on a Cloudera Data Science Workbench deployment.

Adding a Worker Node

Important: If you are using Spark 2.2, review [JDK 8 Requirement for Spark 2.2]() on page 31 for any additional steps you might need to perform to configure `JAVA_HOME` on the new nodes .

Using Cloudera Manager

Perform the following steps to add a new worker node to Cloudera Data Science Workbench.

1. Add a new host to your cluster. Make sure this is a gateway host and you are not running any services on this host. You might choose to create a host template for worker nodes that will be added to Cloudera Data Science Workbench in the future. The host template must include the Docker Daemon, Worker, and Spark 2 gateway roles.
2. Assign the Spark 2 gateway role to the new host. For instructions, refer the Cloudera Manager documentation at Adding a Role Instance.
3. Go to the Cloudera Data Science Workbench service.
4. Click the **Instances** tab.
5. Click **Add Role Instances**.
6. Assign the Worker and Docker Daemon roles to the new host. Click **Continue**.
7. Review your changes and click **Continue**. The wizard finishes by performing any actions necessary to prepare the cluster for the new role instances. *Do not start the new roles at this point.*
8. The new host must have the following packages installed on it.

```
nfs-utils
libseccomp
lvm2
bridge-utils
libtool-ltdl
iptables
rsync
policycoreutils-python
selinux-policy-base
selinux-policy-targeted
ntp
ebtables
bind-utils
nmap-ncat
openssl
e2fsprogs
redhat-lsb-core
```

   You can either manually install these packages now, *or*, go to the **Configuration** page and enable the **Install Required Packages** property to allow Cloudera Manager to install them for you as part of the **Prepare Node** command run in the next step.
9. Go to the **Instances** page and select the new host. From the list of available actions, select the **Prepare Node** command to install the required packages on the new node.
10. On the **Instances** page, select the new role instances and click **Actions for Selected** > **Start**.

Using Packages

For instructions on how to add a worker node to Cloudera Data Science Workbench, see Installing Cloudera Data Science Workbench on a Worker Node.

Removing a Worker Node

Using Cloudera Manager

Perform the following steps to remove a worker node from Cloudera Data Science Workbench.

1. Log into the Cloudera Manager Admin Console.
2. Click the **Instances** tab.
3. Select the Docker Daemon and Worker roles on the node to be removed from Cloudera Data Science Workbench.
4. Select **Actions for Selected** > **Stop** and click **Stop** to confirm the action. Click **Close** when the process is complete.
5. On the **Instances** page, re-select the Docker Daemon and Worker roles that were stopped in the previous step.
6. Select **Actions for Selected** > **Delete** and then click **Delete** to confirm the action.

Using Packages

To remove a worker node:

1. On the master node, run the following command to delete the worker node:

```
kubectl delete node <worker_node_domain_name>
```

2. Reset the worker node.

```
cdsw reset
```

Changing the Domain Name

Cloudera Data Science Workbench allows you to change the domain of the web console.

Using Cloudera Manager

1. Log into the Cloudera Manager Admin Console.
2. Go to the Cloudera Data Science Workbench service.
3. Click the **Configuration** tab.
4. Search for the **Wildcard DNS Domain** property and modify the value to reflect the new domain.
5. Click **Save Changes**.
6. Restart the Cloudera Data Science Workbench service to have the changes go into effect.

Using Packages

1. Open `/etc/cdsw/config/cdsw.conf` and set the `DOMAIN` variable to the new domain name.

```
DOMAIN="new-cdsw.company.com"
```

2. Run the following commands to have the new domain name go into effect.

```
cdsw reset
cdsw init
```

Cloudera Data Science Workbench Scaling Guidelines

New nodes can be added and removed from a Cloudera Data Science Workbench deployment without interrupting any jobs already scheduled on existing hosts. Therefore, it is rather straightforward to increase capacity based on observed usage. At a minimum, Cloudera recommends you allocate at least 1 CPU core and 2 GB of RAM per concurrent session or job. CPU can burst above a 1 CPU core share when spare resources are available. Therefore, a 1 CPU core allocation is often adequate for light workloads. Allocating less than 2 GB of RAM can lead to out-of-memory errors for many applications.

As a general guideline, Cloudera recommends nodes with RAM between 60GB and 256GB, and between 16 and 48 cores. This provides a useful range of options for end users. SSDs are strongly recommended for application data storage.

For some data science and machine learning applications, users can collect a significant amount of data in memory within a single R or Python process, or use a significant amount of CPU resources that cannot be easily distributed into the CDH cluster. If individual users frequently run larger workloads or run workloads in parallel over long durations, increase the total resources accordingly. Understanding your users' concurrent workload requirements or observing actual usage is the best approach to scaling Cloudera Data Science Workbench.

Backup and Disaster Recovery for Cloudera Data Science Workbench

All application data for Cloudera Data Science Workbench, including project files and database state, is stored at `/var/lib/cdsw`. Given typical access patterns, it is strongly recommended that `/var/lib/cdsw` be stored on a dedicated SSD block device or SSD RAID configuration. Because application data is not replicated to HDFS or backed up by default, site administrators must enable a backup strategy to meet any disaster recovery scenarios.

Cloudera strongly recommends both regular backups and backups before upgrades and is not responsible for any data loss.

Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the cdsw_protect_stop_restart.sh script. This is to help avoid the data loss issue detailed in TSB-346.

Data Collection in Cloudera Data Science Workbench

Cloudera Data Science Workbench collects usage and diagnostic data in two ways, both of which can be controlled by system administrators.

Usage Tracking

Cloudera Data Science Workbench collects aggregate usage data by sending limited tracking events to Google Analytics and Cloudera servers. No customer data or personal information is sent as part of these bundles.

Disable Usage Tracking

1. Log in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Click the **Settings** tab.
4. Uncheck **Send usage data to Cloudera**.

Cloudera Manager also collects and sends anonymous usage information using Google Analytics to Cloudera. If you are running a CSD-based deployment and want to disable data collection by Cloudera Manager, see Managing Anonymous Usage Data Collection.

Diagnostic Bundles

Diagnostic bundles are used to aid in debugging issues filed with Cloudera Support. They can be created using either Cloudera Manager (only for CSD-based deployments), or the command line. This section also provides details on the information collected by Cloudera Data Science workbench as part of these bundles.

Using Cloudera Manager

If you are working on a CSD-based deployment, Cloudera Data Science Workbench logs and diagnostic data are available as part of the diagnostic bundles created by Cloudera Manager. By default, Cloudera Manager is configured to collect diagnostic data weekly and to send it to Cloudera *automatically*. You can schedule the frequency of data collection on a daily, weekly, or monthly schedule, or even disable the scheduled collection of data. To learn how to configure the frequency of data collection or disable it entirely, see Diagnostic Data Collection by Cloudera Manager.

You can also manually trigger a collection and transfer of diagnostic data to Cloudera at any time. For instructions, see Manually Collecting and Sending Diagnostic Data to Cloudera.

Using the Command Line

Diagnostic bundles can be created by system administrators using the `cdsw logs` command. Two archives result from running the `cdsw logs` command:

- **Redacted -** Sensitive information is redacted from this archive. This is the bundle that you should attach to any case opened with Cloudera Support. The file name will be of the form, `cdsw-logs-`*`$hostname`*`-`*`$date`*`-`*`$time`*`.redacted.tar.gz`.

- **Original -** The original archive is meant for internal use. It should be retained at least for the duration of the support case, in case any critical information was redacted. This archive can be shared with Cloudera at your discretion. The file name will be of the form, `cdsw-logs-$hostname-$date-$time.tar.gz`.

  The contents of these archives are stored in text and can easily be inspected by system administrators. The original and the redacted forms are designed to be easily diff-able.

Information Collected in Diagnostic Bundles

Cloudera Data Science Workbench diagnostic bundles collect the following information:

- System information such as hostnames, operating system, kernel modules and settings, available hardware, and system logs.
- Cloudera Data Science Workbench version, status information, configuration, and the results of install-time validation checks.
- Details about file systems, devices, and mounts in use.
- CDH cluster configuration, including information about Java, Kerberos, installed parcels, and CDH services such as Spark 2.
- Network configuration and status, including interfaces, routing configuration, and reachability.
- Status information for system services such as Docker, Kubernetes, NFS, and NTP.
- Listings for processes, open files, and network sockets.
- Reachability, configuration, and logs for Cloudera Data Science Workbench application components.
- Hashed Cloudera Data Science Workbench user names.
- Information about Cloudera Data Science Workbench sessions, including type, engine, ownership, termination status, and performance data.

Ports Used By Cloudera Data Science Workbench

Cloudera Data Science Workbench runs on gateway nodes in a CDH cluster. As such, Cloudera Data Science Workbench acts as a gateway and requires full connectivity to CDH services (Impala, Spark 2, etc.) running on the cluster. Additionally, in the case of Spark 2, CDH cluster nodes will require access to the Spark driver running on a set of random ports (20050-32767) on Cloudera Data Science Workbench nodes.

Internally, the Cloudera Data Science Workbench master and worker nodes require full connectivity with no firewalls. Externally, end users connect to Cloudera Data Science Workbench exclusively through a web server running on the master node, and therefore do not need direct access to any other internal Cloudera Data Science Workbench or CDH services.

This information has been summarized in the following table.

Components	Details
Communication with the CDH cluster	**CDH -> Cloudera Data Science Workbench** The CDH cluster must have access to the Spark driver that runs on Cloudera Data Science Workbench nodes, on a set of randomized ports in the range, **20050-32767**.
	Cloudera Data Science Workbench -> CDH As a gateway service, Cloudera Data Science Workbench must have access to all the ports used by CDH and Cloudera Manager.

Components	Details
Communication with the Web Browser	The Cloudera Data Science Workbench web application is available at port **80**. HTTPS access is available over port **443**.

Cloudera Data Science Workbench Security Guide

This guide is intended for site administrators who want to secure a Cloudera Data Science Workbench deployment using TLS/SSL encryption and Kerberos authentication. It also provides instructions on configuring external authentication using LDAP and SAML. This guide assumes that you have basic knowledge of Linux and systems administration practices, in general.

Enabling TLS/SSL for Cloudera Data Science Workbench

Cloudera Data Science Workbench uses HTTP and WebSockets (WS) to support interactive connections to the Cloudera Data Science Workbench web application. However, these connections are not secure by default. This topic describes how you can use TLS/SSL to enforce secure encrypted connections, using HTTPS and WSS (WebSockets over TLS), to the Cloudera Data Science Workbench web application.

Specifically, Cloudera Data Science Workbench can be configured to use a TLS termination proxy to handle incoming connection requests. The termination proxy server will decrypt incoming connection requests and forwards them to the Cloudera Data Science Workbench web application. A TLS termination proxy can be internal or external.

Internal Termination

An internal termination proxy will be run by Cloudera Data Science Workbench's built-in load balancer, called the ingress controller, on the master node. The ingress controller is primarily responsible for routing traffic and load balancing between Cloudera Data Science Workbench's web service backend. Once configured, as shown in the instructions that follow, it will start terminating HTTPS traffic as well. The primary advantage of internal termination approach is simplicity.

External Termination

External TLS termination can be provided through a number of different approaches. Common examples include:

- Load balancers, such as the AWS Elastic Load Balancer
- Modern firewalls
- Reverse web proxies, such as `nginx`
- VPN appliances supporting TLS/SSL VPN

Organizations that require external termination will often have standardized on single approach for TLS. The primary advantage of this approach is that it allows such organizations to integrate with Cloudera Data Science Workbench without violating their IT department's policies for TLS. For example, with an external termination proxy, Cloudera Data Science Workbench does not need access to the TLS private key.

Load balancers and proxies often require a URL they can ping to validate the status of the web service backend. For instance, you can configure a load balancer to send an HTTP GET request to `/internal/load-balancer/health-ping`. If the response is 200 (OK), that means the backend is healthy. Note that, as with all communication to the web backend from the load balancer when TLS is terminated externally, this request should be sent over HTTP and not HTTPS.

Related topic: Troubleshooting TLS/SSL Errors on page 117

Private Key and Certificate Requirements

The TLS certificate issued by your CA must list both, the Cloudera Data Science Workbench, as well as a wildcard for all first-level subdomains. For example, if the Cloudera Data Science Workbench domain is `cdsw.company.com`, then the TLS certificate must include both `cdsw.company.com` and `*.cdsw.company.com`.

Creating a Certificate Signing Request (CSR)

Use the following steps to create a Certificate Signing Request (CSR) to submit to your CA.

> **Important:** Make sure you use `openssl`, and not `keytool`, to perform these steps. Keytool does not support a wildcard Subject Alternative Name (SAN) and cannot create flat files.

1. Create a `cdsw.cnf` file and populate it with the required configuration parameters including the SAN field values.

```
vi cdsw.cnf
```

2. Copy and paste the default `openssl.cnf` from: http://web.mit.edu/crypto/openssl.cnf.
3. Modify the following sections and save the `cdsw.cnf` file:

```
[ CA_default ]
default_md = sha2

[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext

[ req_distinguished_name ]
countryName                  = Country Name (2 letter code)
stateOrProvinceName          = State or Province Name (full name)
localityName                = Locality Name (eg, city)
organizationName            = Organization Name (eg, company)
commonName                  = Common Name (e.g. server FQDN or YOUR name)

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1   = *.cdsw.company.com
DNS.2   = cdsw.company.com
```

Key points to note:

- The domains set in the `DNS.1` and `DNS.2` entries above must match the `DOMAIN` set in `cdsw.conf`.
- The `default_md` parameter must be set to sha256 at a minimum. Older hash functions such as SHA1 are deprecated and will be rejected by browsers, either currently or in the very near future.
- The `commonName` (CN) parameter will be ignored by browsers. You must use Subject Alternative Names.

4. Run the following command to generate the CSR.

```
openssl req -out cert.csr -newkey rsa:2048 -nodes -keyout private.key -config cdsw.cnf
```

This command generates the private key and the CSR in one step. The `-nodes` switch disables encryption of the private key (which is not supported by Cloudera Data Science Workbench at this time).

5. Run the following command to verify that the certificate issued by the CA lists both the required domains, `cdsw.company.com` and `*.cdsw.company.com`, under `X509v3 Subject Alternative Name`.

```
openssl x509 -in <your_tls_cert>.crt -noout -text
```

You should also verify that a valid hash function is being used to create the certificate. For SHA-256, the value under Signature Algorithm will be `sha256WithRSAEncryption`.

Configuring Internal Termination

Depending on your deployment (CSD or RPM), use one of the following sets of instructions to configure internal termination.

CSD Deployments

To enable internal termination, configure the following properties in the CDSW service in Cloudera Manager.

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to the CDSW service and click **Configuration**.
3. Search for the following properties and configure as required.

    - **Enable TLS** - When enabled, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.
    - **TLS Key for Internal Termination** - Set to the path of the TLS private key.
    - **TLS Certificate for Internal Termination** - Set to the path of the TLS certificate.

    Certificates and keys must be in PEM format.

4. Click **Save Changes**.
5. Restart the CDSW service.

RPM Deployments

To enable internal termination, configure the following properties in `cdsw.conf` (on all Cloudera Data Science Workbench nodes).

- `TLS_ENABLE` - When set to `true`, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.
- `TLS_KEY` - Set to the path of the TLS private key.
- `TLS_CERT` - Set to the path of the TLS certificate.

  Certificates and keys must be in PEM format.

You can configure these properties either as part of the installation process or after the fact. If you make any changes to `cdsw.conf` after installation is complete, make sure to restart the master and worker nodes as needed.

Configuring External Termination

Depending on your deployment (CSD or RPM), use one of the following sets of instructions to configure external termination.

CSD Deployments

To enable external termination, configure the following property in the CDSW service in Cloudera Manager.

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to the CDSW service and click **Configuration**.
3. Search for the following properties and configure as required.

    - **Enable TLS** - When enabled, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.

      The **TLS Key for Internal Termination** and **TLS Certificate for Internal Termination** properties must be left blank.

4. Click **Save Changes**.
5. Restart the CDSW service.

RPM Deployments

To enable external termination, configure the following property in `cdsw.conf` (on all Cloudera Data Science Workbench nodes).

- `TLS_ENABLE` - When set to `true`, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.

  The `TLS_KEY` and `TLS_CERT` properties must be left blank.

You can configure this property either as part of the installation process or after the fact. If you make any changes to `cdsw.conf` after installation is complete, make sure to restart the master and worker nodes as needed.

Limitations

- Communication within the Cloudera Data Science Workbench cluster is not encrypted.
- Cloudera Data Science Workbench does not support encrypted private keys with internal TLS termination. If you require an encrypted private key, use external TLS termination with a terminating proxy that does support encrypted private keys.
- Troubleshooting can be difficult because browsers do not typically display helpful security errors with WebSockets. Often they will just silently fail to connect.
- **Self-signed certificates -** In general, browsers do not support self-signed certificates for WSS. Your certificate must be signed by a Certificate Authority (CA) that your users' browsers will trust. Cloudera Data Science Workbench will not function properly if browsers silently abort WebSockets connections.

  If you are using a TLS certificate that has been used to sign itself, and is not signed by a CA in the trust store, then the browser will display a dialog asking if you want to trust the certificate provided by Cloudera Data Science Workbench. This means you are using a self-signed certificate, which is *not supported and will not work*. In this case WSS connections will likely be aborted silently, regardless of your response (Ignore/Accept) to the dialog.

  As long as you have a TLS certificate signed by a CA certificate in the trust store, it will be supported and will work with Cloudera Data Science Workbench. For example, if you need to use a certificate signed by your organization's internal CA, make sure that all your users import your root CA certificate into their machine's trust store. This can be done using the Keychain Access application on Macs or the Microsoft Management Console on Windows.

Configuring Cloudera Data Science Workbench Deployments Behind a Proxy

If your deployment is behind an HTTP or HTTPS proxy, set the respective `HTTP_PROXY` or `HTTPS_PROXY` property in `/etc/cdsw/config/cdsw.conf` to the hostname of the proxy you are using.

```
HTTP_PROXY="<http://proxy_host>:<proxy-port>"
HTTPS_PROXY="<http://proxy_host>:<proxy_port>"
```

If you are using an intermediate proxy such as Cntlm to handle NTLM authentication, add the Cntlm proxy address to the `HTTP_PROXY` *or* `HTTPS_PROXY` fields in `cdsw.conf`.

```
HTTP_PROXY="http://localhost:3128"
HTTPS_PROXY="http://localhost:3128"
```

If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.

To enable trust, perform the following steps on the master and worker nodes.

1. Copy the proxy's root certificate to the trusted CA certificate store (`ca-trust`) on the Cloudera Data Science Workbench host.

```
cp /tmp/<proxy-root-certificate>.crt /etc/pki/ca-trust/source/anchors/
```

2. Use the following command to rebuild the trusted certificate store.

```
update-ca-trust extract
```

3. If you will be using custom engine images that will be pulled from a Docker repository, add the proxy's root certificates to a directory under `/etc/docker/certs.d`. For example, if your Docker repository is at `docker.repository.mycompany.com`, create the following directory structure:

```
/etc/docker/certs.d
|-- docker.repository.mycompany.com          # Directory named after Docker repository

    |-- <proxy-root-certificate>.crt         # Docker-related root CA certificates
```

   This step is not required with the standard engine images because they are included in the Cloudera Data Science Workbench RPM.

4. Re-initialize Cloudera Data Science Workbench to have this change go into effect.

```
cdsw init
```

Configure hostnames to be skipped from the proxy

Use the `NO_PROXY` field in `cdsw.conf` to include a comma-separated list of hostnames that should be skipped from the proxy. These typically include `127.0.0.1`, `localhost`, the value of `MASTER_IP`, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines. This change must be made on the master and on all the worker nodes.

At a minimum, Cloudera recommends the following `NO_PROXY` configuration.

```
127.0.0.1,localhost,<MASTER_IP>,100.66.0.1,100.66.0.2,
100.66.0.3,100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,
100.66.0.9,100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,
100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,100.66.0.20,
100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,100.66.0.25,100.66.0.26,
100.66.0.27,100.66.0.28,100.66.0.29,100.66.0.30,100.66.0.31,100.66.0.32,
100.66.0.33,100.66.0.34,100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,
100.66.0.39,100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,
100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,100.66.0.50,
100.77.0.129,100.77.0.130
```

Hadoop Authentication with Kerberos for Cloudera Data Science Workbench

Cloudera Data Science Workbench users can authenticate themselves using Kerberos against the cluster KDC defined in the host's `/etc/krb5.conf` file. Cloudera Data Science Workbench does not assume that your Kerberos principal is always the same as your login information. Therefore, you will need to make sure Cloudera Data Science Workbench knows your Kerberos identity when you sign in.

To authenticate against your cluster's Kerberos KDC, go to the top-right dropdown menu, click **Account settings** > **Hadoop Authentication**, and enter your Kerberos principal. To authenticate, either enter your password or click **Upload Keytab** to upload the keytab file directly to Cloudera Data Science Workbench. Once successfully authenticated, Cloudera Data Science Workbench uses your stored credentials to ensure that you are secure when running your workloads.

When you authenticate with Kerberos, Cloudera Data Science Workbench will store your keytab in an internal database. When you subsequently run an engine, the keytab is used by a Cloudera Data Science Workbench sidecar container to generate ticket-granting tickets for use by your code. Ticket-granting tickets allow you to access resources such as Spark, Hive, and Impala, on Kerberized CDH clusters.

While you can view your current ticket-granting ticket by typing `klist` in an engine terminal, there is no way for you or your code to view your keytab. This prevents malicious code and users from stealing your keytab.

Important:

- If the `/etc/krb5.conf` file is not available on all Cloudera Data Science Workbench nodes, authentication will fail.
- If you do not see the **Hadoop Authentication** tab, make sure you are accessing your personal account's settings from the top right menu. If you have selected a team account, the tab will not be visible when accessing the Team Settings from the left sidebar.
- When you upload a Kerberos keytab to authenticate yourself to the CDH cluster, Cloudera Data Science Workbench displays a fleeting error message ('cancelled') in the bottom right corner of the screen, even if authentication was successful. This error message can be ignored.

UI Behavior for Non-Kerberized Clusters

The contents of the **Hadoop Authentication** tab change depending on whether the cluster is kerberized. For a secure cluster with Kerberos enabled, the **Hadoop Authentication** tab displays a **Kerberos** section with fields to enter your Kerberos principal and username. However, if Cloudera Data Science Workbench cannot detect a `krb5.conf` file on the host, it will assume the cluster is not kerberized, and the **Hadoop Authentication** tab will display **Hadoop Username Override** configuration instead.

For a non-kerberized cluster, by default, your Hadoop username will be set to your Cloudera Data Science Workbench login username. To override this default and set an alternative `HADOOP_USER_NAME`, go to the **Hadoop Username Override** setting at **Account settings** > **Hadoop Authentication**.

If the **Hadoop Authentication** tab is incorrectly displaying Kerberos configuration fields for a non-kerberized cluster, make sure the `krb5.conf` file is not present on the host running Cloudera Data Science Workbench. If you do find any instances of `krb5.conf` on the host, run `cdsw stop`, remove the `krb5.conf` file(s), and run `cdsw start`. You should now see the expected **Hadoop Username Override** configuration field.

Limitations

- Cloudera Data Science Workbench only supports Active Directory and MIT KDCs. PowerBroker-equipped Active Directory is not supported.
- Cloudera Data Science Workbench does not support the use of Kerberos plugin modules in `krb5.conf`.

Configuring External Authentication with LDAP and SAML

Cloudera Data Science Workbench supports user authentication against its internal local database, and against external services such as Active Directory, OpenLDAP-compatible directory services, and SAML 2.0 Identity Providers. By default, Cloudera Data Science Workbench performs user authentication against its internal local database. This topic describes the signup process for the first user, how to configure authentication using LDAP, Active Directory or SAML 2.0, and an optional workaround that allows site administrators to bypass external authentication by logging in using the local database in case of misconfiguration.

User Signup Process

The first time you visit the Cloudera Data Science Workbench web console, the first account that you sign up with is a local administrator account. If in the future you intend to use external services for authentication, Cloudera recommends you use exclusive username & email combinations, rather than site administrators' work email addresses, for both the first site administrator account, and any other local accounts created before switching to external authentication. If the username/email combinations are not unique, an email address might end up being associated with different usernames, one for the external authentication service provider and one for a local Cloudera Data Science Workbench account. This will prevent the user from logging into Cloudera Data Science Workbench with their credentials for the external authentication service.

The link to the signup page is only visible on the login page when the authentication type is set to `local`. When you enable external services for authentication, signing up through the local database is disabled, and user accounts are automatically created upon their first successful login.

Optionally, site administrators can use a **Require invitation to sign up** flag under the **Admin** > **Settings** tab to require invitation tokens for account creation. When this flag is enabled, only users that are invited by site administrators can login to create an account, regardless of the authentication type.

Important: If you forget the original credentials, or make a mistake with LDAP or SAML configuration, you can use the workaround described in Debug Login URL on page 112.

When you switch to using external authentication methods such as LDAP or SAML 2.0, user accounts will be automatically created upon each user's first successful login. Cloudera Data Science Workbench will extract user attributes such as username, email address and full name from the authentication responses received from the LDAP server or SAML 2.0 Identity Provider and use them to create the user accounts.

Configuring LDAP/Active Directory Authentication

Cloudera Data Science Workbench supports both search bind and direct bind operations to authenticate against an LDAP or Active Directory directory service. The search bind authentication mechanism performs an ldapsearch against the directory service, and binds using the found Distinguished Name (DN) and password provided. The direct bind authentication mechanism binds to the LDAP server using a username and password provided at login.

You can configure Cloudera Data Science Workbench to use external authentication methods by clicking the **Admin** link on the left sidebar and selecting the **Security** tab. Select **LDAP** from the list to start configuring LDAP properties.

General Configuration

- **LDAP Server URI**: Required. The URI of the LDAP/Active Directory server against which Cloudera Data Science Workbench should authenticate. For example, `ldaps://ldap.`*`company`*`.com:636`.
- **Use Direct Bind**: If checked, the username and password provided at login are used with the LDAP Username Pattern for binding to the LDAP server. If unchecked, Cloudera Data Science Workbench uses the search bind mechanism and two configurations, LDAP Bind DN and LDAP Bind Password, are required to perform the ldapsearch against the LDAP server.
- **LDAP Bind DN**: Required when using search bind. The DN to bind to for performing ldapsearch. For example, `cn=admin,dc=company,dc=com`.
- **LDAP Bind Password**: Required when using search bind. This is the password for the LDAP Bind DN.
- **LDAP Username Pattern**: Required when using direct bind. Provides a template for the DN that will ultimately be sent to the directory service during authentication. For example, `sAMAccountName={0},ou=People,dc=company,dc=com`. The `{0}` parameter will be replaced with the username provided at login.
- **LDAP Search Base**: Required. The base DN from which to search for the provided LDAP credentials. For example, `ou=Engineering,dc=company,dc=com`.
- **LDAP User Filter**: Required. The LDAP filter for searching for users. For example, `(&(sAMAccountName={0})(objectclass=person))`. The `{0}` placeholder will be replaced with the username provided at login.
- **LDAP User Username Attribute**: Required. The case-sensitive username attribute of the LDAP directory service. This is used by Cloudera Data Science Workbench to perform the bind operation and extract the username from the response. Common values are `uid`, `sAMAccountName`, or `userPrincipalName`.

When you select **Use Direct Bind**, Cloudera Data Science Workbench performs a direct bind to the LDAP server using the LDAP Username Pattern with the credentials provided on login (not **LDAP Bind DN** and **LDAP Bind Password**).

By default, Cloudera Data Science Workbench performs an LDAP search using the bind DN and credentials specified for the **LDAP Bind DN** and **LDAP Bind Password** configurations. It searches the subtree, starting from the base DN specified for the **LDAP Search Base** field, for an entry whose attribute specified in **LDAP User Username Attribute**, has the same value as the username provided on login. Cloudera Data Science Workbench then validates the user-provided password against the DN found as a result of the search.

LDAPS Support

To support secure communication between Cloudera Data Science Workbench and the LDAP/Active Directory server, Cloudera Data Science Workbench needs to be able to validate the identity of the LDAP/Active Directory service. If the certificate of your LDAP/Active Directory service was signed by a trusted or commercial Certificate Authority (CA), it is not necessary to upload the CA certificate here. However, if your LDAP/Active Directory certificate was signed by a self-signed CA, you must upload the self-signed CA to the Cloudera Data Science Workbench in order to use LDAP over SSL (LDAPS).

- **CA Certificate**: Only required if your LDAP/Active Directory certificate was not signed by a trusted or commercial CA. If your LDAP/Active Directory certificate was signed by a trusted or commercial CA, there is no need to upload it here.

Test LDAP Configuration

You can test your LDAP/Active Directory configuration by entering your username and password in the **Test LDAP Configuration** section. Use this form to simulate the user login process and verify the validity of your LDAP/Active Directory configuration. Before using this form, make sure you click **Update** to save the LDAP configuration you want to test.

Configuring SAML Authentication

Cloudera Data Science Workbench supports the Security Assertion Markup Language (SAML) for Single Sign-on (SSO) authentication; in particular, between an identity provider (IDP) and a service provider (SP). The SAML specification defines three roles: the principal (typically a user), the IDP, and the SP. In the use case addressed by SAML, the principal (user agent) requests a service from the service provider. The service provider requests and obtains an identity assertion from the IDP. On the basis of this assertion, the SP can make an access control decision—in other words it can decide whether to perform some service for the connected principal.

The primary SAML use case is called web browser single sign-on (SSO). A user with a user agent (usually a web browser) requests a web resource protected by a SAML SP. The SP, wanting to know the identity of the requesting user, issues an authentication request to a SAML IDP through the user agent. In the context of this terminology, Cloudera Data Science Workbench operates as a SP.

Cloudera Data Science Workbench supports both SP- and IDP-initiated SAML 2.0-based SSO. Its Assertion Consumer Service (ACS) API endpoint is for consuming assertions received from the Identity Provider. If your Cloudera Data Science Workbench domain root were `cdsw.`*`company`*`.com`, then this endpoint would be available at `http://cdsw.`*`company`*`.com/api/v1/saml/acs`. SAML 2.0 metadata is available at `http://cdsw.`*`company`*`.com/api/v1/saml/metadata` for IDP-initiated SSO. Cloudera Data Science Workbench uses HTTP Redirect Binding for authentication requests and expects to receive responses from HTTP POST Binding.

When Cloudera Data Science Workbench receives the SAML responses from the Identity Provider, it expects to see at least the following user attributes in the SAML responses:

- The unique identifier or username. Valid attributes are:
  - `uid`
  - `urn:oid:0.9.2342.19200300.100.1.1`
- The email address. Valid attributes are:
  - `mail`
  - `email`
  - `urn:oid:0.9.2342.19200300.100.1.3`
- The common name or full name of the user. Valid attributes are:
  - `cn`
  - `urn:oid:2.5.4.3`

In the absence of the `cn` attribute, Cloudera Data Science Workbench will attempt to use the following user attributes, if they exist, as the full name of the user:

- The first name of the user. Valid attributes are:
  - `givenName`
  - `urn:oid:2.5.4.42`
- The last name of the user. Valid attributes are:
  - `sn`
  - `urn:oid:2.5.4.4`

Configuration Options

- **Cloudera Data Science Workbench Entity ID**: Required. A globally unique name for Cloudera Data Science Workbench as a Service Provider. This is typically the URI.
- **Cloudera Data Science Workbench NameID Format**: Optional. The name identifier format for both Cloudera Data Science Workbench and Identity Provider to communicate with each other regarding a user. Default: `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`.
- **Cloudera Data Science Workbench Authentication Context**: Optional. SAML authentication context classes are URIs that specify authentication methods used in SAML authentication requests and authentication statements. Default: `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`.
- **Cloudera Data Science Workbench Certificate**: Required if the Cloudera Data Science Workbench Private Key is set, otherwise optional. You can upload a certificate in the PEM format for the Identity Provider to verify the authenticity of the authentication requests generated by Cloudera Data Science Workbench. The uploaded certificate is made available at the `http://cdsw.`*`company`*`.com/api/v1/saml/metadata` endpoint.
- **Cloudera Data Science Workbench Private Key**: Optional. If you upload a private key, you must upload a corresponding certificate as well so that the Identity Provider can use the certificate to verify the authentication requests sent by Cloudera Data Science Workbench. You can upload the private key used for both signing authentication requests sent to Identity Provider and decrypting assertions received from the Identity Provider.
- **Identity Provider SSO URL**: Required. The entry point of the Identity Provider in the form of URI.
- **Identity Provider Signing Certificate**: Optional. Administrators can upload the X.509 certificate of the Identity Provider for Cloudera Data Science Workbench to validate the incoming SAML responses.

For on-premises deployment, you must provide a certificate and private key, generated and signed with your trusted Certificate Authority, for Cloudera Data Science Workbench to establish secure communication with the Identity Provider.

Cloudera Data Science Workbench extracts the Identity Provider SSO URL and Identity Provider Signing Certificate information from the uploaded Identity Provider Metadata file. Cloudera Data Science Workbench also expects all Identity Provider metadata to be defined in a `<md:EntityDescriptor>` XML element with the namespace "`urn:oasis:names:tc:SAML:2.0:metadata`", as defined in the SAMLMeta-xsd schema.

Debug Login URL

When using external authentication, such as LDAP, Active Directory or SAML 2.0, even a small mistake in authentication configurations in either Cloudera Data Science Workbench or the Identity Provider could potentially block all users from logging in.

Cloudera Data Science Workbench provides an optional fallback debug login URL for site administrators to log in against the local database with their username/password created during the signup process before changing the external authentication method. The debug login URL is `http://cdsw.`*`company`*`.com/`**`login?debug=1`**. If you do not remember the original password, you can reset it by going directly to `http://cdsw.company.com/forgot-password`. When configured to use external authentication, the link to the forgot password page is disabled on the login page for security reasons.

Disabling the Debug Login Route

Optionally, the debug login route can be disabled to prevent users from accessing Cloudera Data Science Workbench via local database when using external authentication. In case of external authentication failures, when the debug login route is disabled, root access to the master host is required to re-enable the debug login route.

Contact Cloudera Support for more information.

SSH Keys

This topic describes the different types of SSH keys used by Cloudera Data Science Workbench, and how you can use those keys to authenticate to an external service such as GitHub.

Personal Key

Cloudera Data Science Workbench automatically generates an SSH key pair for your user account. You can rotate the key pair and view your public key on your user settings page. It is not possible for anyone to view your private key.

Every console you run has your account's private key loaded into its SSH-agent. Your consoles can use the private key to authenticate to external services, such as GitHub. For instructions, see Adding SSH Key to GitHub on page 113.

Team Key

Like Cloudera Data Science Workbench users, each Cloudera Data Science Workbench team has an associated SSH key. You can access the public key from the team's account settings. Click **Account**, then select the team from the drop-down menu at the upper right corner of the page.

Team SSH keys provide a useful way to give an entire team access to external resources such as databases or GitHub repositories (as described in the next section). When you launch a console in a project owned by a team, you can use that team's SSH key from within the console.

Adding SSH Key to GitHub

If you want to use GitHub repositories to create new projects or collaborate on projects, use the following instructions to add your Cloudera Data Science Workbench SSH public key to your GitHub account:

1. Sign in to Cloudera Data Science Workbench.
2. Go to the upper right drop-down menu and switch context to the account whose key you want to add.
3. On the left sidebar, click **Settings**.
4. Go to the **SSH Keys** tab and copy your public SSH key.
5. Sign in to your GitHub account and add the Cloudera Data Science Workbench key copied in the previous step to your GitHub account. For instructions, refer the GitHub documentation on adding SSH keys to GitHub.

SSH Tunnels

In some environments, external databases and data sources reside behind restrictive firewalls. A common pattern is to provide access to these services using a bastion host with only the SSH port open. This introduces complexity for end users who must manually set up SSH tunnels to access data. Cloudera Data Science Workbench provides a convenient way to connect to such resources.

From the **Project** > **Settings** > **Tunnels** page, you can use your SSH key to connect Cloudera Data Science Workbench to an external database or cluster by creating an SSH tunnel. If you create an SSH tunnel to an external server in one of your projects, then all engines that you run in that project are able to connect securely to a port on that server by connecting to a local port. The encrypted tunnel is completely transparent to the user or code.

To create an automatic SSH tunnel:

1. Open the **Project Settings** page.
2. Open the **Tunnels** tab.
3. Click **New Tunnel**.
4. Enter the server IP Address or DNS hostname.
5. Enter your username on the server.
6. Enter the local port that should be proxied, and to which remote port on the server.

Then, on the remote server, configure SSH to accept password-less logins using your individual or team SSH key. Often, you can do so by appending the SSH key to the file `/home/username/.ssh/authorized_keys`.

Troubleshooting Cloudera Data Science Workbench

Use one or more of the following courses of action to start debugging issues with Cloudera Data Science Workbench.

- Check the status of the application.

```
cdsw status
```

- Make sure the contents of the configuration file are correct.

```
cat /etc/cdsw/config/cdsw.conf
```

- SSH to your master host and run the following node validation command to check that the key services are running:

```
cdsw validate
```

The following sections describe solutions to potential problems and error messages you may encounter while installing, configuring or using Cloudera Data Science Workbench.

Understanding Installation Warnings

This section describes solutions to some warnings you might encounter during the installation process.

Preexisting iptables rules not supported

```
WARNING: Cloudera Data Science Workbench requires iptables, but does not support
preexisting iptables rules.
```

Kubernetes makes extensive use of `iptables`. However, it's hard to know how pre-existing `iptables` rules will interact with the rules inserted by Kubernetes. Therefore, Cloudera recommends you run the following commands to clear all pre-existing rules before you proceed with the installation.

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

Remove the entry corresponding to /dev/xvdc from /etc/fstab

Cloudera Data Science Workbench installs a custom filesystem on its Application and Docker block devices. These filesystems will be used to store user project files and Docker engine images respectively. Therefore, Cloudera Data Science Workbench requires complete access to the block devices. To avoid losing any existing data, make sure the block devices allocated to Cloudera Data Science Workbench are reserved only for the workbench.

Linux sysctl kernel configuration errors

Kubernetes and Docker require non-standard kernel configuration. Depending on the existing state of your kernel, this might result in `sysctl` errors such as:

```
sysctl net.bridge.bridge-nf-call-iptables must be set to 1
```

This is because the settings in `/etc/sysctl.conf` conflict with the settings required by Cloudera Data Science Workbench. Cloudera cannot make a blanket recommendation on how to resolve such errors because they are specific to your deployment. Cluster administrators may choose to either remove or modify the conflicting value directly in `/etc/sysctl.conf`, remove the value from the conflicting configuration file, or even delete the module that is causing the conflict.

To start diagnosing the issue, run the following command to see the list of configuration files that are overwriting values in `/etc/sysctl.conf`.

```
SYSTEMD_LOG_LEVEL=debug /usr/lib/systemd/systemd-sysctl
```

You will see output similar to:

```
Parsing /usr/lib/sysctl.d/00-system.conf
Parsing /usr/lib/sysctl.d/50-default.conf
Parsing /etc/sysctl.d/99-sysctl.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Parsing /etc/sysctl.d/k8s.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-iptables in file
'/etc/sysctl.d/k8s.conf'.
Parsing /etc/sysctl.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.conf'.
Setting 'net/ipv4/conf/all/promote_secondaries' to '1'
Setting 'net/ipv4/conf/default/promote_secondaries' to '1'
...
```

`/etc/sysctl.d/k8s.conf` is the configuration added by Cloudera Data Science Workbench. Administrators must make sure that no other file is overwriting values set by `/etc/sysctl.d/k8s.conf`.

CDH parcels not found at /opt/cloudera/parcels

There are two possible reasons for this warning:

- If you are using a custom parcel directory, you can ignore the warning and proceed with the installation. Once the Cloudera Data Science Workbench is running, set the path to the CDH parcel in the admin dashboard. See Non-standard CDH Parcel Location on page 90.
- This warning can be an indication that you have not added gateway roles to the Cloudera Data Science Workbench nodes. In this case, do not ignore the warning. Exit the installer and go to Cloudera Manager to add gateway roles to the cluster. See Configure Gateway Hosts Using Cloudera Manager on page 42.

404 Not Found Error

The `404 Not Found` error might appear in the browser when you try to reach the Cloudera Data Science Workbench web application.

This error is an indication that your installation of Cloudera Data Science Workbench was successful, but there was a mismatch in the domain configured in `cdsw.conf` and the domain referenced in the browser. To fix the error, go to `/etc/cdsw/config/cdsw.conf` and check that the URL you supplied for the `DOMAIN` property matches the one you are trying to use to reach the web application. This is the wildcard domain dedicated to Cloudera Data Science Workbench, not the hostname of the master node.

If this requires a change to `cdsw.conf`, after saving the changes run `cdsw reset` followed by `cdsw init`.

Troubleshooting Kerberos Errors

HDFS commands fail with Kerberos errors even though Kerberos authentication is successful in the web application

If Kerberos authentication is successful in the web application, and the output of `klist` in the engine reveals a valid-looking TGT, but commands such as `hdfs dfs -ls /` still fail with a Kerberos error, it is possible that your cluster is missing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File. The JCE policy file is required when Red Hat uses AES-256 encryption. This library should be installed on each cluster host and will live under `$JAVA_HOME`. For more information, see Using AES-256 Encryption.

Cannot find renewable Kerberos TGT

Cloudera Data Science Workbench runs its own Kerberos TGT renewer which produces non-renewable TGT. However, this confuses Hadoop's renewer which looks for renewable TGTs. If the Spark 2 logging level is set to `WARN` or lower, you may see exceptions such as:

```
16/12/24 16:38:40 WARN security.UserGroupInformation: Exception encountered while running
 the renewal command. Aborting renew thread. ExitCodeException exitCode=1: kinit: Resource
 temporarily unavailable while renewing credentials

16/12/24 16:41:23 WARN security.UserGroupInformation: PriviledgedActionException
as:user@CLOUDERA.LOCAL (auth:KERBEROS) cause:javax.security.sasl.SaslException: GSS
initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level:
 Failed to find any Kerberos tgt)]
```

This is *not* a bug. Spark 2 workloads will not be affected by this. Access to Kerberized resources should also work as expected.

Troubleshooting TLS/SSL Errors

This section describes some common issues with TLS configuration on Cloudera Data Science Workbench. Common errors include:

- Cloudera Data Science Workbench initialisation fails with an error such as:

```
Error preparing server: tls: failed to parse private key
```

- Your browser reports that the Cloudera Data Science Workbench web application is not secure even though you have enabled TLS settings as per Enabling TLS/SSL for Cloudera Data Science Workbench on page 104.

Possible Causes and Solutions

- **Certificate does not include the wildcard domain** - Confirm that the TLS certificate issued by your CA lists both, the Cloudera Data Science Workbench domain, as well as a wildcard for all first-level subdomains. For example, if your Cloudera Data Science Workbench domain is `cdsw.company.com`, then the TLS certificate must include both `cdsw.company.com` and `*.cdsw.company.com`.
- **Path to the private key and/or certificate is incorrect** - Confirm that the path to the private key file is correct by comparing the path and file name to the values for `TLS_KEY` and/or `TLS_CERT` in `cdsw.conf` or Cloudera Manager. For example:

```
TLS_CERT="/path/to/cert.pem"
TLS_KEY="/path/to/private.key"
```

- **Private key file does not have the right permissions** - The private key file must have read-only permissions. Set it as follows:

```
chmod 444 private.key
```

- **Private key is encrypted** - Cloudera Data Science Workbench does not support encrypted private keys. Check to see if your private key is encrypted:

```
$ cat private.key

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,11556F53E4A2824A
```

  If the private key is encrypted as shown above, use the following steps to decrypt it:

  1. Make a backup of the private key file.

```
mv private.key private.key.encrypted
```

  2. Decrypt the backup private key and save the file to `private.key`. You will be asked to enter the private key password.

```
openssl rsa -in private.key.encrypted -out private.key
```

- **Private key and certificate are not related** - Check to see if the private key matches the public key in the certificate.

  1. Print a hash of the private key modulus.

```
openssl rsa -in private.key -noout -modulus | openssl md5
(stdin)= 7a8d72ed61bb4be3c1f59e4f0161c023
```

  2. Print a hash of the public key modulus.

```
openssl x509 -in cert.pem -noout -modulus | openssl md5
(stdin)= 7a8d72ed61bb4be3c1f59e4f0161c023
```

  If the `md5` hash output of both keys is different, they are not related to each other, and will not work. You must revoke the old certificate, regenerate a new private key and Certificate Signing Request (CSR), and then apply for a new certificate.

Troubleshooting Issues with Running Workloads

This section describes some potential issues data scientists might encounter once the application is running workloads.

404 error in Workbench after starting an engine

This is typically caused because a wildcard DNS subdomain was not set up before installation. While the application will largely work, the engine consoles are served on subdomains and will not be routed correctly unless a wildcard DNS entry pointing to the master node is properly configured. You might need to wait 30-60 minutes until the DNS entries propagate. For instructions, see Set Up a Wildcard DNS Subdomain on page 36.

Engines cannot be scheduled due to lack of CPU or memory

A symptom of this is the following error message in the Workbench: "Unschedulable: No node in the cluster currently has enough CPU or memory to run the engine."

Either shut down some running sessions or jobs or provision more nodes for Cloudera Data Science Workbench.

Workbench prompt flashes red and does not take input

The Workbench prompt flashing red indicates that the session is not currently ready to take input.

Cloudera Data Science Workbench does not currently support non-REPL interaction. One workaround is to skip the prompt using appropriate command-line arguments. Otherwise, consider using the terminal to answer interactive prompts.

PySpark jobs fail due to HDFS permission errors

```
: org.apache.hadoop.security.AccessControlException: Permission denied: user=alice,
access=WRITE, inode="/user":hdfs:supergroup:drwxr-xr-x
```

To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

PySpark jobs fail due to Python version mismatch

```
Exception: Python in worker has different version 2.6 than that in driver 2.7, PySpark
 cannot run with different minor versions
```

One solution is to install the matching Python 2.7 version on all the cluster hosts. Another, more recommended solution is to install the Anaconda parcel on all CDH cluster hosts. Cloudera Data Science Workbench Python engines will use the version of Python included in the Anaconda parcel which ensures Python versions between driver and workers will always match. Any library paths in workloads sent from drivers to workers will also match because Anaconda is present in the same location across all hosts. Once the parcel has been installed, set the `PYSPARK_PYTHON` environment variable in the Cloudera Data Science Workbench Admin dashboard. Alternatively, you can use Cloudera Manager to set the path.

Cloudera Data Science Workbench Command Line Reference

This topic describes the commands available with the Cloudera Data Science Workbench command line utility, `cdsw`. Running `cdsw` without any arguments will print a brief description of each command. Starting with Cloudera Data Science Workbench 1.2, the commands available for a CSD-based deployment will be a subset of those available for a package-based deployment. For a CSD-based deployment, common commands such as `cdsw start`, `stop`, and `restart`, must be run through the Cloudera Data Science Workbench service in Cloudera Manager. For instructions, see Starting, Stopping, and Restarting the Service on page 75.

Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the cdsw_protect_stop_restart.sh script. This is to help avoid the data loss issue detailed in TSB-346.

All of the following commands can be used in a package-based deployment. Those available for CSD-based deployments have been marked in the table.

Command	[illegible]	Description and Usage
`cdsw init`		Initializes and bootstraps the master node. Use this command to start Cloudera Data Science Workbench. Also see, Additional Usage Notes on page 121
`cdsw start`		Run on the master node to start application components.
`cdsw stop`		Run on the master node to stop application components.
`cdsw restart`		Run on the master node to restart application components. To restart a worker node, use `cdsw reset`, followed by `cdsw join`. These commands have been explained further in this topic.
`cdsw reset`		De-registers and resets a node. On a worker node, this command will remove the worker from the cluster. On the master node, this command will bring down the application and effectively tear down the Cloudera Data Workbench deployment.
`cdsw join`		Initializes a worker node. After a worker node has been added, run this command on the *worker* node to add it to the Cloudera Data Science Workbench cluster. This registers the worker nodes with the master, and increases the available pool of resources for workloads.
`cdsw status`	■	Displays the current status of the application.
`cdsw validate`	■	Performs diagnostic checks for common errors that might be preventing the application from running as expected. This command should typically be run as the first step to troubleshooting any problems with the application, as indicated by `cdsw status`.
`cdsw logs`	■	Creates a tarball with diagnostic information for your Cloudera Data Science Workbench installation. If you file a case with Cloudera Support, run this command on each node and attach the resulting bundle to the case.

Command	D	Description and Usage
		For more details on the information collected in these bundles, see Data Collection in Cloudera Data Science Workbench on page 101.
`cdsw version`	🟦	Displays the version number of Cloudera Data Science Workbench.
`cdsw help`	🟦	Displays the inline help for the Cloudera Data Science Workbench CLI.

Additional Usage Notes

Note: These notes apply only to package-based deployments. In case of CSD-based deployments where you cannot directly modify `cdsw.conf`, Cloudera Manager will prompt you if the Cloudera Data Science Workbench service needs to be restarted.

Changes to `cdsw.conf`: Make sure `cdsw.conf` is consistent across all Cloudera Data Science Workbench nodes. Any changes made to the file must be copied over to all the other nodes.

- **Master Node -** Changes to the `JAVA_HOME`, `MASTER_IP`, `DOCKER_BLOCK_DEVICES`, and `APPLICATION_BLOCK_DEVICE` parameters in `cdsw.conf` require a re-initialization of the master node.

```
cdsw reset
cdsw init
```

Changes to other `cdsw.conf` parameters such as domain name changes, or TLS and HTTP proxy changes, require a restart of the application components.

```
cdsw restart
```

- **Worker Node -** Changes to `cdsw.conf` on a worker node, require a restart of the worker node as follows:

```
cdsw reset
cdsw join
```

Cloudera Data Science Workbench FAQs

Where can I get a sample project to try out Cloudera Data Science Workbench?

Cloudera Data Science Workbench ships with sample project templates that you can use to try running workloads. These are currently available in Python, R, and Scala. See Create a Project from a Template on page 45.

What are the software and hardware requirements for Cloudera Data Science Workbench?

For detailed information on the software and hardware required to successfully run Cloudera Data Science Workbench, see Cloudera Data Science Workbench 1.2.x Requirements and Supported Platforms on page 30.

Can I run Cloudera Data Science Workbench on hosts shared with other Hadoop services?

No. Cloudera does not support running Cloudera Data Science Workbench on non-dedicated nodes. Running other services on Cloudera Data Science Workbench nodes can lead to unreliable execution of workloads and difficult to debug out-of-memory errors.

How does Cloudera Data Science Workbench use Docker and Kubernetes?

Cloudera Data Science Workbench uses Docker and Kubernetes to manage containers. Currently, Cloudera Data Science Workbench only supports the versions of Docker and Kubernetes that are shipped with each release. Upgrading Docker, or Kubernetes, or running on third-party Kubernetes clusters is not supported.

Cloudera does not support Kubernetes or Docker for running any other workloads beyond those on Cloudera Data Science Workbench.

Can I run Cloudera Data Science Workbench on my own Kubernetes cluster?

This is not supported.

Does Cloudera Data Science Workbench support REST API access?

Starting with version 1.1.0, Cloudera Data Science Workbench supports a Jobs REST API that lets you orchestrate jobs from 3rd party workflow tools. For more details, see Cloudera Data Science Workbench Jobs API on page 71.

Other means of API access to Cloudera Data Science Workbench are not supported at this time.

How do I contact Cloudera for issues regarding Cloudera Data Science Workbench?

Cloudera Support

If you are a Cloudera customer, you can register for an account to create a support ticket at the support portal.

Before you log a support ticket, run the following command on the master host to create a tarball with diagnostic information for your Cloudera Data Science Workbench installation.

```
cdsw logs
```

Attach the resulting bundle to the support case you create.

Cloudera Community

Register for the Cloudera Community forums and post your questions or feedback on the Cloudera Data Science Workbench board.

Appendix: Apache License, Version 2.0

licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```