cloudera®

Cloudera Data Science Workbench

Important Notice

Cloudera, Inc.
1001 Page Mill Road, Bldg 3
Palo Alto, CA 94304
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Data Science Workbench 1.0.x
Date: July 17, 2017

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 (incubating)

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

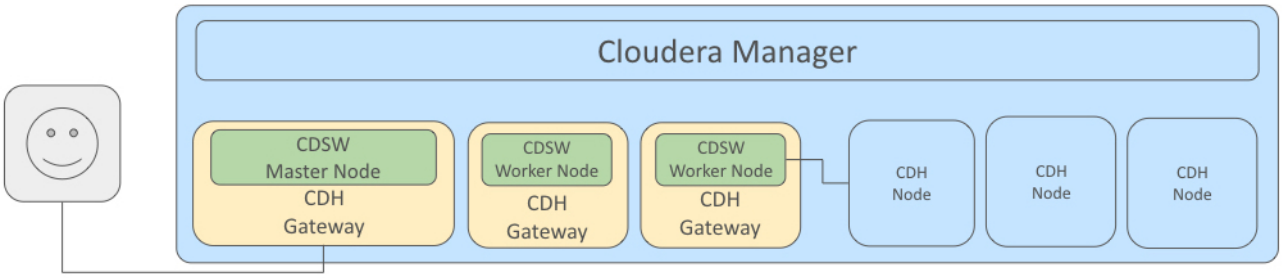
A single Cloudera Data Science Workbench deployment can support different business groups sharing common infrastructure without interfering with one another, or placing additional demands on IT.

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

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`.

Each session or job gets its own container that lives as long as the analysis. After a job or session 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

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.

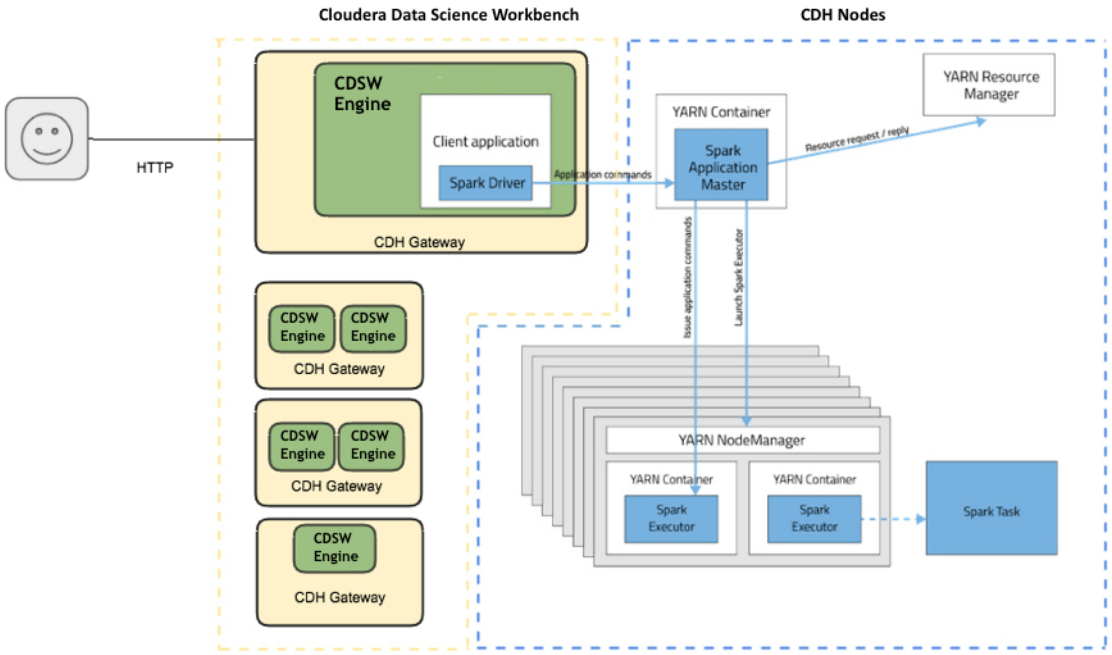
Cloudera's Distribution of Apache Spark 2

> **Important:** The rest of this topic assumes you are familiar with Apache Spark and Cloudera's 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 he the following figure:



More resources:

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

Cloudera Data Science Workbench Release Notes

These release notes provide information on fixed issues, known issues, and limitations for all generally-available (GA) versions of Cloudera Data Science Workbench.

Cloudera Data Science Workbench 1.0.1

This section lists the issues fixed in Cloudera Data Science Workbench 1.0.1. For a list of known issues and limitations, see Known Issues and Limitations in Cloudera Data Science Workbench 1.0.x on page 12.

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.

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 8.

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

This section lists the currently known issues and limitations in Cloudera Data Science Workbench.

- **Upgrading to Version 1.0.1**
  - If JAVA_HOME is in a non-standard location that cannot automatically be detected by Cloudera Data Science Workbench, sessions will fail to launch after the upgrade is complete.

    Workaround: Go to the Site Administration panel and click the **Engines** tab. Under the Environment Variables section, delete the custom value you previously set for JAVA_HOME. Now install Java in a location that Cloudera Data Science Workbench can detect automatically using the `bigtop-detect-javahome` utility.

- **Crashes and Hangs**
  - 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.

- **Security**
  - 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.
  - Self-signed certificates are not supported for TLS termination.
  - The TLS_KEY is not password protected.
  - PowerBroker-equipped Active Directory is not supported.
  - Using Kerberos plugin modules in `krb5.conf` is not supported.
  - 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.

- **Usability**
  - 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.
  - 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.
  - Some Jupyter visualization libraries such as Bokeh do not always render as expected. Generating a static HTML visualization and using the engines /cdn feature can help work around the issue.
  - Spark SQL syntax does not work correctly in the Scala Spark shell. Using `import spark.implicits_` to enable the use of Spark SQL syntax in the shell fails with the following error:

```
import spark.implicits._

Name: Compile Error
Message: <console>:19: error: stable identifier required, but
this.$line7$read.spark.implicits found.
```

    This is because `spark` is defined as a `var` which is not a stable identifier. You can work around this issue by overwriting the shell's `spark` definition and defining it as a `val`. You should then be able to import the `implicits` object:

```
val spark = SparkSession.builder().getOrCreate()
import spark.implicits._
```

Cloudera Data Science Workbench 1.0.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

- CDH 5.7 or higher.
- Cloudera Manager 5.11 or higher. All cluster hosts must be managed by Cloudera Manager.

  Important: All Cloudera Data Science Workbench administration tasks require root access to the cluster hosts. Currently, Cloudera Data Science Workbench does not support single-user mode installations.

- Cloudera Data Science Workbench requires Cloudera's Distribution of Apache Spark 2.1.

Operating System Requirements

Cloudera Data Science Workbench is currently supported only on ***RHEL/CentOS 7.2***.

A gateway node that is dedicated to running Cloudera Data Science Workbench must use RHEL/CentOS 7.2 even if the remaining hosts in your cluster are running any of the other supported operating systems. At this time, other distributions are not supported with Cloudera Data Science Workbench.

JDK Requirements

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

Networking and Security Requirements

- A wildcard subdomain such as `*.cdsw.`*`company`*`.com`. Wildcard subdomains are used to provide isolation for user-generated content.
- No pre-existing `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 disable all pre-existing rules before you proceed with the installation.

- SELinux must be disabled.
- No firewall restrictions across Cloudera Data Science Workbench or CDH hosts.
- No multi-homed networks.
- Non-root SSH access is not allowed on Cloudera Data Science Workbench hosts.

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
  - 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 unreliable execution of user workloads and out-of-memory errors.

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 50.

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+

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 AWS Reference Architecture document.

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 corporate network for browser access. There cannot be 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

Installing Cloudera Data Science Workbench 1.0.x

This topic describes how to install the Cloudera Data Science Workbench package on a CDH cluster managed by Cloudera Manager. Currently, we do not support a Custom Service Descriptor (CSD) or parcel-based installs.

Prerequisites

Review the complete list of prerequisites at Cloudera Data Science Workbench 1.0.x Requirements and Supported Platforms on page 14 before you proceed with the installation.

Installing Cloudera Data Science Workbench 1.0.x from Packages

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

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 62.

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 15.
2. To support workloads running on Cloudera's 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
```

Configure Block Devices

Docker Block Device

Cloudera Data Science Workbench installer will format and mount Docker on each gateway host. 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.

All application data will be located at `/var/lib/cdsw` on the master node. If an application block device is specified during initialization, Cloudera Data Science Workbench will format it as `ext4` and mount it to `/var/lib/cdsw`. If no device is explicitly specified during initialization, Cloudera Data Science Workbench will store all data at `/var/lib/cdsw` and assume the system administrator has formatted and mounted one or more block devices to this location. The second option is recommended for production installations.

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 61 for more information.

Install Cloudera Data Science Workbench on the Master Host

To install Cloudera Data Science Workbench and its dependencies:

1. Download the Cloudera Data Science Workbench repository installer (`cloudera-cdsw.repo`) from the following table and save it to `/etc/yum.repos.d/`.

Red Hat 7 Repository File	Link to RPM
cloudera-cdsw.repo	Cloudera Data Science Workbench 1.0.1
	Cloudera Data Science Workbench 1.0.0

Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

2. Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/cdsw/1/redhat/7/x86_64/cdsw/RPM-GPG-KEY-cloudera
```

3. Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

For guidance on any warnings displayed during the installation process, see Understanding Installation Warnings on page 70.

4. Edit the configuration file at `/etc/cdsw/config/cdsw.conf`. The following table lists the configuration properties that can be configured in `cdsw.conf`.

Properties	Description
Required Configuration	
`DOMAIN`	Wildcard DNS domain configured to point to the master node. If the wildcard DNS entries are configured as `cdsw.<company>.com` and `*.cdsw.<company>.com`, then DOMAIN should be set to `cdsw.<company>.com`. Users' browsers should then contact Cloudera Data Science Workbench at `http://cdsw.<company>.com`. This domain for DNS and is unrelated to Kerberos or LDAP domains.
`MASTER_IP`	IPv4 address for the master node that is reachable from the worker nodes. Within an AWS VPC, `MASTER_IP` should be set 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`.
`DOCKER_BLOCK_DEVICES`	Block device(s) for Docker images (space separated if there are multiple). Use the full path to specify the image(s), for instance, `/dev/xvde`.
Optional Configuration	
`APPLICATION_BLOCK_DEVICE`	**(Master Node Only)** Configure a block device for application state. If this property is left blank, the filesystem mounted at `/var/lib/cdsw` on the master node will be used to store all user data. For production deployments, Cloudera recommends you use this option with a dedicated SSD block device for the `/var/lib/cdsw` mount. (*Not recommended*) If set, Cloudera Data Science Workbench will format the provided block device as `ext4`, mount it to `/var/lib/cdsw`, and store all user data on it. This option has only been provided for proof-of-concept setups, and Cloudera is not responsible for any data loss. Use the full path to specify the mount point, for instance, `/dev/xvdf`.
`TLS_ENABLE`	Enable and enforce HTTPS (TLS/SSL) for web access. Set to `true` to enable and enforce HTTPS access to the web application.

Properties	Description
	You can also set this property to `true` to enable external TLS termination. For more details on TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 62.

`TLS_CERT`
 	
`TLS_KEY`
	Certificate and private key for internal TLS termination.
	
	Setting `TLS_CERT` and `TLS_KEY` will enable internal TLS termination. You must also set `TLS_ENABLE` to `true` above to enable and enforce internal termination. Set these only if you are not terminating TLS externally.
	
	Make sure you specify the full path to the certificate and key files, which must be in `PEM` format.
	
	For details on certificate requirements and enabling TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 62.

`HTTP_PROXY`
 	
`HTTPS_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, 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`	If a SOCKS proxy is in use, set to `socks5://<host>:<port>/`.

`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 recommends the following `NO_PROXY` configuration.
	
	`NO_PROXY="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,`

Properties	Description
	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"

5. Initialize and start Cloudera Data Science Workbench.

```
cdsw init
```

This initialization process requires Internet access to download the Docker image dependencies. The application can take 20-30 minutes to initially download and bootstrap. You can watch the status of application installation and startup with `watch cdsw status`.

(Optional) Install Cloudera Data Science Workbench on Worker Hosts

Cloudera Data Science Workbench supports adding and removing additional worker hosts at any time. Worker hosts allow you to transparently scale the number of concurrent workloads users can run.

To add worker hosts:

1. Download the Cloudera Data Science Workbench repository installer (`cloudera-cdsw.repo`) from the following table and save it to `/etc/yum.repos.d/`.

Red Hat 7 Repository File	Link to RPM
cloudera-cdsw.repo	Cloudera Data Science Workbench 1.0.1
	Cloudera Data Science Workbench 1.0.0

Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

2. Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/cdsw/1/redhat/7/x86_64/cdsw/RPM-GPG-KEY-cloudera
```

3. Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

For guidance on any warnings displayed during the installation process, see Understanding Installation Warnings on page 70.

4. Copy `cdsw.conf` file from the master host:

```
scp root@cdsw-host-1.<company>.com:/etc/cdsw/config/cdsw.conf /etc/cdsw/config/cdsw.conf
```

After initialization, the `cdsw.conf` file includes a generated bootstrap token that allows worker hosts to securely join the cluster. You can get this token by copying the configuration file from master and ensuring it has 644 permissions.

If your hosts have heterogeneous block device configurations, modify the Docker block device settings in the worker host configuration file after you copy it. Worker hosts do not need application block devices, which store the project files and database state, and this configuration option is ignored.

5. On the *master* node, whitelist the IPv4 address of the worker node for the NFS server.

```
cdsw enable <IPv4_address_of_worker>
```

6. On the *worker* node, run the following command to add the host to the cluster:

```
cdsw join
```

   This causes the worker nodes to register themselves with the Cloudera Data Science Workbench master node and increase the available pool of resources for workloads.

7. Return to the master node and verify the host is registered with this command:

```
cdsw status
```

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 24. For more details on collaborating with teams, working on projects, and sharing results, see the Cloudera Data Science Workbench User Guide on page 27.

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.0.x

Important: Before upgrading Cloudera Data Science Workbench, read the Cloudera Data Science Workbench Release Notes on page 12 relevant to the version you are upgrading to.

1. Reset the state of *every* worker node.

```
cdsw reset
```

2. Reset the state of the master node.

```
cdsw reset
```

3. **(Optional)** On the master node, backup the contents of the `/var/lib/cdsw` directory. This is the directory that stores all your application data.

4. 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
```

5. Install the latest version of Cloudera Data Science Workbench on the master node and on all the worker nodes. Follow the same process as you would for a fresh installation. However, note that even though you have installed the latest RPM, your previous configuration settings in `cdsw.conf` will remain unchanged.

   - Install Cloudera Data Science Workbench on the Master Host on page 18
   - (Optional) Install Cloudera Data Science Workbench on Worker Hosts on page 21.

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 27.

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 project workbench 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.

Launch a Session

To launch a session:

1. Click **Open Workbench** in the project overview.
2. Use **Select Engine Kernel** to choose your language.
3. Use **Select Engine Profile** to select the number of CPU cores and memory.
4. 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 the **Terminal Access** tab 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

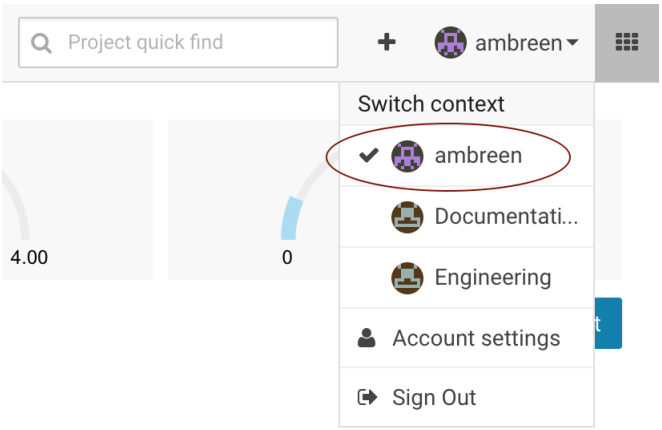
Now that you have successfully run a sample workload with the Cloudera Data Science Workbench, further acquaint yourself with Cloudera Data Science Workbench by reading the User, Administration, and Security guides to learn more about the types of users, how to collaborate on projects, how to use Spark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

- Cloudera Data Science Workbench User Guide on page 27
- Cloudera Data Science Workbench Administration Guide on page 56
- Cloudera Data Science Workbench Security Guide on page 62
- Using Cloudera's Distribution of Apache Spark 2 on page 45

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 the **Settings** tab.

   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 68.

   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 63.

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 the **Settings** tab 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 68. Generally, team SSH keys should not be used to authenticate against Git repositories. Use your personal key instead.

Managing Projects

Projects form the heart of Cloudera Science 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 the **Projects** tab.

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 tab.
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** - Can view 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.

Important: Remember that you, the *contributor*, and the *admin* collaborators edit the same files and have full access to all running sessions and jobs.. A collaborator with *contributor* privileges can modify your project files and any currently running sessions. Therefore, it is important to restrict project access to trusted collaborators only. A malicious user can easily impersonate you within a shared project. For more secure collaboration, and to work on multiple projects, using Git is strongly recommended. This will also help avoid file modification conflicts when doing more serious work.

For more information on collaborating effectively, see Sharing Projects and Analysis Results on page 42.

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 the **Projects** tab.
3. From the list of projects, select the one you want to modify.
4. Click the **Settings** tab 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 Customizing Engine Images on page 59.

   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 35.

   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 69.

   Git

   This page lists a webhook that can be added to your Git configuration to ensure that your project files are updated with the latest changes from the remote repository.

   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.

Using the Workbench

The project workbench 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.

Launch a Session

To launch a session:

1. Click **Open Workbench** in the project overview.
2. Use **Select Engine Kernel** to choose your language.
3. Use **Select Engine Profile** to select the number of CPU cores and memory.
4. 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 the **Terminal Access** tab 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.

Accessing Cloudera Manager and Hue from Cloudera Data Science Workbench

Cloudera Data Science Workbench gives you a way to access your cluster's Cloudera Manager and Hue web UIs from within the Cloudera Data Science Workbench application.. To access these applications, click ⁞⁞⁞ 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 menu.

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 (incubating), 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 53 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.

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 to version control your projects. 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 68.

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. 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.

To install a package:

1. Launch a session in your favorite language.
2. At the command prompt in the bottom right, enter the command to install:

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
```

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 Customizing Engine Images on page 59.

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 by default 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 users' browsers. HTTP services that bind CDSW_PUBLIC_PORT will be available in users' browsers at: http(s)://*<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>*.
CDSW_DOMAIN	The domain on which Cloudera Data Science Workbench is being served. This can be useful for iframing services, as 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.
CDSW_IP_ADDRESS	Other engines in the Cloudera Data Science Workbench cluster can contact this engine on this IP address.

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
```

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

```
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

```
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

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

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

```
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

```
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.

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!")
```

Making Web Services Available

Every console has an environment variable called `CDSW_PUBLIC_PORT`. Applications can contact any service that listens on that port over the Internet at `https://<console-id>.company.com`. You can get the `<console-id>` the environmental variable `CDSW_DASHBOARD_ID` or the string of random letters and numbers from the console URL.

For example, services in console `https://cdsw.company.com/user/project/consoles/xv30miihscnv947b` can be reached at `https://xv30miihscnv947b.company.com`.

Example: A Shiny Application

> **Note:** The Shiny example does not work in Beta 1 due to a known issue.

Use the following steps to create a new Shiny application.

Create a new, blank project and run an R console. Use the following command to install Shiny to the project.

R

```
install.packages('shiny')
```

Create files ui.R and server.R in the project, and copy and paste the contents of the example files provided by Shiny by RStudio:

R

```
# ui.R

shinyUI(bootstrapPage(

 selectInput(inputId = "n_breaks",
   label = "Number of bins in histogram (approximate):",
   choices = c(10, 20, 35, 50),
   selected = 20),

checkboxInput(inputId = "individual_obs",
   label = strong("Show individual observations"),
   value = FALSE),

checkboxInput(inputId = "density",
   label = strong("Show density estimate"),
   value = FALSE),

plotOutput(outputId = "main_plot", height = "300px"),

# Display this only if the density is shown
conditionalPanel(condition = "input.density == true",
   sliderInput(inputId = "bw_adjust",
      label = "Bandwidth adjustment:",
      min = 0.2, max = 2, value = 1, step = 0.2)
    )
 ))
```

R

```
# server.R

shinyServer(function(input, output) {

 output$main_plot <- renderPlot({

   hist(faithful$eruptions,
      probability = TRUE,
      breaks = as.numeric(input$n_breaks),
      xlab = "Duration (minutes)",
      main = "Geyser eruption duration")

   if (input$individual_obs) {
      rug(faithful$eruptions)
   }

   if (input$density) {
      dens <- density(faithful$eruptions,
         adjust = input$bw_adjust)
      lines(dens, col = "blue")
   }
```

```
 })
})
```

Run the following code in the console to load the libraries you will need.

R

```
library('cdsw')
library('shiny')
library('parallel')
```

Now start the Shiny server. Shiny blocks the R process it runs in, so use the parallel package to run it in a separate process.

R

```
mcparallel(runApp(host="0.0.0.0", port=8080, launch.browser=FALSE,
    appDir="/home/cdsw", display.mode="auto"))
```

Finally, create an IFrame widget in the console and point it at the Shiny server.

R

```
service.url <- paste("http://", Sys.getenv("CDSW_ENGINE_ID"), ".",
Sys.getenv("CDSW_DOMAIN"), sep="")
Sys.sleep(5)
iframe(src=service.url, width="640px", height="480px")
```

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 29.

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 30.

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 Results Externally

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.

To share results, 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 a single console output to anybody with access to the link. You can revoke access at any time. 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.

This method of sharing shows colleagues and collaborators your progress without your having to spend time creating a report.

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.

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.

Using Cloudera's Distribution of Apache Spark 2

For an architectural overview of how Cloudera's Distribution of Apache Spark 2 works with Cloudera Data Science Workbench, see Overview: Cloudera Distribution of Apache Spark 2. 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.

Configuring Cloudera's 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 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>
```

Accessing Spark 2 Web UIs from Cloudera Data Science Workbench

Spark 2 provides a web interface for running Spark applications to monitor and track progress of executions in real time. There will be one web UI for each Spark application driver running in Cloudera Data Science Workbench. To access this page, point your browser to `spark-<session_ID>.<cdsw.company.com>`. *session_ID* is the ID of the Cloudera Data Science Workbench session running the Spark application, and *cdsw.company.com* is the domain of the Cloudera Data Science Workbench instance you are accessing.

The *session_ID* is the alphanumeric string at the end of a session URL. For example if the URL of a running session is `cdsw.my.company.com/user1/project1/engines/`**`tb89zxn1078xqulm`**, then the Web UI for this session's Spark driver will be at, `spark-`**`tb89zxn1078xqulm`**`.cdsw.my.company.com`.

You can also use environmental variables to generate the HTML link for the session's web UI.

Shell

```
echo spark-${CDSW_ENGINE_ID}.${CDSW_DOMAIN}
```

Python

```
import os, IPython
url = "spark-%s.%s" % (os.environ["CDSW_ENGINE_ID"], os.environ["CDSW_DOMAIN"])
IPython.display.HTML("<a href=http://%s>Spark UI</a>" % url)
```

R

```
library("cdsw")
url = paste("spark-", Sys.getenv("CDSW_ENGINE_ID"), ".", Sys.getenv("CDSW_DOMAIN"),
sep="")
html(paste("<a href=http://", url, ">Spark UI</a>", sep=""))
```

Scala

```
val id = sys.env("CDSW_ENGINE_ID")
val domain = sys.env("CDSW_DOMAIN")
val url = s"http://spark-$id.$domain"
println(url)
```

Spark History Server

Spark 2 also 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. You can learn more about using the Spark History Server in the Apache Spark 2 monitoring documentation.

Cloudera Data Science Workbench gives you a way to access the Spark History Server from within the Cloudera Data Science Workbench application. Click ⠿ in the upper right hand corner of the Cloudera Data Science Workbench web application, and select **Spark History** from the dropdown menu.

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** > **Environment**. 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 SparkR or sparklyr. Although Cloudera does not ship or support SparkR or sparklyr, we recommend using sparklyr as the R interface for Cloudera Data Science Workbench.

Installing sparklyr

You can install sparklyr from GitHub.

```
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's 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.

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 58.

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.

On the **Admin** > **Activity** tab of the dashboard, you can see the status for all the previous and current sessions running on the workbench.

Managing Engine Profiles

On the **Admin** > **Engines** page, under **Engines Profiles**, you can provide default engine profiles to users. These engine profiles provide default CPU 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.

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, see Customizing Engine Images on page 59.

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.

Setting JAVA_HOME

By default, Cloudera Data Science Workbench will attempt to find your Java home. However, if you are using a non-standard path for Java, add `JAVA_HOME` under the Environmental Variables section. This will set `JAVA_HOME` and mount the `JAVA_HOME` directory into engines so that the Java version matches other CDH services.

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 specify folders that should be mounted through to all the engines. Cloudera Data Science Workbench will automatically mount CDH parcels and client configuration.

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 58.

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.

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**.

Customizing 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. Currently, Cloudera Data Science Workbench only supports *public* Docker images in registries accessible to the Cloudera Data Science Workbench nodes.

Site administrators can whitelist images for use in projects, and project administrators can select which of these white-listed images is installed for their projects.

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:1
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 your image with the Dockerfile.

```
docker build -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

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

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. On the master node, run the following command to disable the worker node on the NFS server:

```
cdsw disable <IP_address_worker_node>
```

3. Reset the worker node.

```
cdsw reset
```

Changing the Domain Name

Cloudera Data Science Workbench allows you to change the domain of the web console.

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.

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. Understanding your users' concurrent workload requirements or observing actual usage is the best approach to scaling Cloudera Data Science Workbench.

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.

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.

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 secure connections between your browser and the Cloudera Data Science Workbench web application.

Transport Layer Security (TLS) is an industry standard set of cryptographic protocols for securing communications over a network. TLS evolved from Secure Sockets Layer (SSL, which remains part of the name for historical reasons). TLS/SSL provides privacy and data integrity between applications communicating over a network by encrypting the packets transmitted between endpoints.

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. 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 backends. Once configured, as shown in the following instructions, it will start terminating HTTPS traffic as well. External termination can be done at an external load balancer such as the AWS Elastic Load Balancer.

Certificate Requirements

- The TLS certificate must list both, the Cloudera Data Science Workbench `DOMAIN` (set in `cdsw.conf`), as well as a wildcard for all first-level subdomains. For example, if `DOMAIN` is set to `cdsw.company.com`, then the TLS certificate must include both `cdsw.company.com` and `*.cdsw.company.com`. To verify this, run the following command and ensure that both domains are listed under `X509v3 Subject Alternative Name`.

```
openssl x509 -in <your_tls_cert>.crt -noout -text
```

- Many browsers no longer recognize SHA1 certificate signatures. If your certificate is signed by an internal Certificate Authority, make sure you use at least SHA-256. You can verify this in the output of the previous `openssl` command, under the Signature Algorithm field. For SHA-256, the value under Signature Algorithm will be `sha256WithRSAEncryption`.

Internal Termination

Internal TLS termination must be configured during the installation process and is governed by the following variables in `cdsw.conf`.

- `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.

External Termination

External TLS termination must be configured during the installation process and is governed by the `TLS_ENABLE` variable in `cdsw.conf`.

- `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.

Many load balancers and proxies require an 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.

Limitations

- Communication within the Cloudera Data Science Workbench cluster is not encrypted.
- Troubleshooting can be difficult because browsers do not typically display helpful security errors with WebSockets. Often they will just silently fail to connect.
- 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 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.

  If your browser asks if you want to trust the certificate provided by Cloudera Data Science Workbench, that means you are using a self-signed certificate, and WSS connections will likely be aborted silently, regardless of your response to the dialog.

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.

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

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.

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`.

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.
- Kerberos principals are case sensitive when stored as keytabs, even though interactive kinit is case-insensitive. Always use your full Kerberos principal, such as `username@EDH.COMPANY.COM`. If your deployment has a case mismatch between the Kerberos principals and the Linux usernames, you will need further configuration to successfully authenticate to the KDC. See Case Mismatch Between Kerberos Principals and Linux Usernames on page 64.
- Cloudera Data Science Workbench does not support the use of Kerberos plugin modules in `krb5.conf`.

Case Mismatch Between Kerberos Principals and Linux Usernames

Cloudera Data Science Workbench typically only accepts Kerberos principals whose primary precisely matches the user principal name (UPN) in Active Directory. This is because Kerberos keytabs require principals to have the correct case. For example, if the user's UPN is `UPPERCASE@edh.company.com`, the principal must be `UPPERCASE@EDH.COMPANY.COM`.

In contrast, password-based `kinit` is not case-sensitive and will accept even `uppercase@EDH.COMPANY.COM` as a valid principal. This difference becomes problematic when the Linux usernames corresponding to upper-cased Kerberos principals are lower-cased; for example, if the Kerberos principal is `UPPERCASE@EDH.COMPANY.COM`, but the Linux username is `uppercase`, CDH services will not be able to correctly infer the Linux username from a Kerberos TGT.

Depending on the encryption type, you can use one of the following ways to ensure CDH services are able to infer the Linux username from a Kerberos TGT:

- The `krb5.conf` files on the CDH nodes can be modified to include only `rc4-hmac` or `arcfour-hmac` in the `permitted_enctypes` field. These encryption types cause Cloudera Data Science Workbench to be case insensitive with respect to the principal's primary when used with Active Directory.

  AES encryption types cause the principal to become case-sensitive when used with keytabs but not with passwords. While this is not a bug, some users might be unaware of the actual case of their principal because `kinit` with password is not case-sensitive.

- If you want to use AES encryption types that are stronger than `rc4-hmac`, use `auth_to_local` to map your Kerberos principal to a differently-cased Linux username. For example, if your Kerberos principals are of the form, `UPPERCASE@EDH.COMPANY.COM`, but the corresponding Linux usernames are `uppercase`, you can configure an `auth_to_local` mapping in Cloudera Manager as follows:

  1. Go to the Cloudera Manager Admin console. You can do this by clicking ⁞⁞⁞ > **Cloudera Manager** in the upper right hand corner of the Cloudera Data Science Workbench web application.
  2. Go to the service you want to configure.

3. Click **Configuration**.
4. Search for the **Advanced Configuration Snippet (Safety Valve) for core-site.xml** property and add an `auth_to_local` rule to map the service's principal name to the Linux username. For example, the following rule maps a principal's primary to its lower-cased version.

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[1:$1]/L
    DEFAULT
  </value>
</property>
```

5. Click **Save Changes**.
6. Repeat steps 2-6 for each CDH service that requires the mapping.
7. Deploy client configuration and restart the relevant services. You should now be able to authenticate with your correctly-cased Kerberos principal, and CDH services will be able to associate your uppercase principals with the lowercase Linux usernames.

   You can test the `auth_to_local` rule on a Cloudera Data Science Workbench host or from a session terminal as follows:

```
hadoop org.apache.hadoop.security.HadoopKerberosName <username@EDH.COMPANY.COM>
```

For more details on this method, see the Configuring the Mapping from Kerberos Principals to Short Names.

Configuring External Authentication

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. This recommendation applies to 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 68.

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.

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.

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 68.

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. There is also an example of the Cloudera Data Science Workbench configuration file for your reference.

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 disable all pre-existing rules before you proceed with the installation.

Please 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'
Setting 'net/ipv6/conf/default/disable_ipv6' to '0'
Setting 'kernel/sysrq' to '16'
...
```

`/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 57.
- 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 17.

Java is not installed or is installed in a non-standard location.

If you have not already installed Java, exit the installer and install Oracle JDK on the cluster.

If Java has been installed, but is in a non-standard location, once installation is complete, set JAVA_HOME in the Cloudera Data Science Workbench site administrator dashboard. See Setting JAVA_HOME on page 57.

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 console.

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 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 17.

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.

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.

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 24.

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 1.0.x Requirements and Supported Platforms on page 14.

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.

Does Cloudera Data Science Workbench support operating systems besides RHEL/CentOS?

Cloudera Data Science Workbench runs workloads within Docker containers, which are officially supported only on ***RHEL/CentOS 7.2***.

- Docker RHEL Support
- RHEL 7 Support
- RHEL 6 Non-Support

At this time, Cloudera is focusing exclusively on providing support for RHEL/CentOS.

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?

Direct API access to Cloudera Data Science Workbench is not supported at this time.

How do I contact Cloudera Support for issues regarding Cloudera Data Science Workbench?

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.