# CONTENTS

1 Enterprise Steam Release Notes ................................................. 3  
  1.1 Change Log ............................................................... 3  

2 Logging in to Enterprise Steam .................................................. 7  
  2.1 The Enterprise Steam UI .................................................. 7  

3 Clusters and Notebooks ............................................................. 9  
  3.1 Launching a New Cluster ................................................... 9  
  3.2 Deleting Clusters ........................................................... 13  
  3.3 Notebooks .................................................................. 13  

4 Configurations ................................................................... 15  

5 Using Enterprise Steam with H2O Flow ........................................... 17  
  5.1 The H2O Flow UI .............................................................. 18  

6 Using Enterprise Steam with Python/R .......................................... 21  
  6.1 Using Enterprise Steam with Python ....................................... 21  
  6.2 Using Enterprise Steam with R .............................................. 26
Enterprise Steam is a service for securely starting and connecting to H2O YARN jobs in a Hadoop environment. Enterprise Steam offers security, resource control, and resource monitoring out of the box in a multi-tenant architecture so that organizations can focus on the core of their data science practice. Enterprise Steam enables streamlined H2O adoption in a secure manner that complies with company policies.

For data scientists, Enterprise Steam provides easy R/Python APIs and a Web UI for starting H2O YARN jobs and allows data scientists to practice data science in their own H2O cluster.

For Admins, Enterprise Steam provides control over which H2O versions are available and the YARN queues to use. Admins can also cap resources that data scientists can use.

This document describes how to start and use Enterprise Steam. Note that this document assumes that an Admin has successfully installed and started Enterprise Steam on a YARN edge node using the instructions provided in the Enterprise Steam Installation and Setup steps.

**Note:** Before you begin using Enterprise Steam, be sure that you have a Chrome browser version 50+ and that your minimum version of H2O is 3.10.4.1. If necessary, follow the instructions on the H2O Download page for your platform to upgrade H2O. For current customers with enterprise support, earlier versions can be supported. Contact H2O.ai if you require support for an earlier version.
1.1 Change Log

1.1.1 Version 1.4.10 (Jul 19, 2019)

- Added an ability to download H2O logs from Steam UI
- Added a config option for custom KRB5_CONFIG (STEAM_KRB5_CONFIG)
- Added support for templating in YARN queue names
- Added a check to ensure the Spark version and Sparkling Water version match
- Added a check to warn admin if Steam port is occupied
- Improved fetching of H2O YARN job status to now use impersonation
- Fixed h2osteam Conda package
- Fixed issue when Sparkling Water cluster failed to start when SPARK_YARN_MODE property was set

1.1.2 Version 1.4.9 (May 21, 2019)

- Added Conda packages for Python client
- Added ability to delete multiple stopped clusters at the same time
- Added Python PySpark path configuration to profile
- Added YARN queue configuration to user setting
- Removed the need for secondary external port
- Configuration fields are now protected from trailing and leading whitespace characters
- Updated pysparkling dependencies for Python2
- Fixed an issue with YARN usernames being lowercased when they were not configured to do so

1.1.3 Version 1.4.8 (Apr 15, 2019)

- Add support for RSparkling through Jupyter notebook
- Add option to disable Jupyterhub
- Expose internal address and node ID for H2O cluster in the Steam Web UI
- Add option to specify path to custom Pyspark Python for Sparkling Water
• Fix pysparkling dependencies for Python3
• Updated engine upload backend

1.1.4 Version 1.4.7 (Apr 3, 2019)

• Fix Sparkling Water proxy issue with uppercase usernames
• Improve uploading h2o-3 engines
• Set SPARK_YARN_MODE correctly based on the hadoop distribution

1.1.5 Version 1.4.6 (Apr 1, 2019)

• Added ability to choose H2O-3 Leader Node when starting a cluster
• Added ability to control the number of clusters a user can spin per cluster profile
• Added option to select default Sparkling Water backend
• Added automatic redirection back to login with an expired session cookie
• Added an ability to auto-assign Steam profiles according to SAML profiles
• Docs: Add “Before you begin installation” section
• Docs: Documented steam.yaml configuration options
• Docs: Updated documentation
• Fix an issue when Steam was hitting API endpoints of dead clusters
• Fix and issue when hadoop-unjar files were not deleted from temp directory
• Fix issue with uppercase usernames and Sparkling Water on Hadoop

1.1.6 Version 1.4.5 (Mar 22, 2019)

• Added Configurable Steam Web UI timeout (STEAM_WEB_UI_TIMEOUT_MIN)

1.1.7 Version 1.4.4 (Feb 20, 2019)

• Make log file permissions configurable (STEAM_LOG_PERMISSIONS)
• H2O: Communicate with cluster using leader node only
• SW: Added support for Hive tables
• SW: Disable Spark dynamic allocation for internal backend
• SW: Bundle and distribute all pysparkling dependencies
• LDAP group configuration is no longer mandatory
• Bug fixes for Jupyterhub
• Bug fixes for Sparkling Water params
• Bug fixes for CDH5
1.1.8 Version 1.4.3 (Jan 29, 2019)

- Added ability to provide secure HTTP headers (SERVER STRICT TRANSPORT, SERVER_X_XSS_PROTECTION, SERVER_CONTENT_SECURITY_POLICY)
- Improved logging output, time in UTC
- Accept encrypted TLS private keys
- Log files permissions elevated for the group
- Bug fixes for HDP Spark
- Bug fixes for Sparkling Water integration
- Bug fixes for Steam Python API
- Bug fixes for LDAP connector
- Bug fixes for Jupyterhub
- Bug fixes for Steam Web UI

1.1.9 Version 1.4.2 (Jan 1, 2019)

- Add ability to upload H2O engine jar from Python/R API
- Remove cluster dialog waits until cluster has been shut down
- Log successful logins
- Log files permissions reduced
- Steam nbextension shows “please wait” when changing clusters
- Document setup and troubleshooting steps for Hadoop/Kerberos configuration
- Bug fixes

1.1.10 Version 1.4.1 (Dec 3, 2018)

- Enable Sparkling Water API from API
- Hide disabled user with a checkbox
- Display message if there are no clusters to show
- Fix glibc dependency for steam binary
- Fix documentation version
- Add release notes to documentation

1.1.11 Version 1.4.0 (Nov 23, 2018)

- Add Sparkling Water integration
- Add SAML authentication
- Add More detailed cluster profiles
- New cluster overview
• New launch cluster page
• Add option to generate Personal access tokens
In a Chrome web browser (version 50+), navigate to the Enterprise Steam web server using the login credentials provided by your Admin and/or Enterprise Steam Admin. This Enterprise Steam web server is the server on which an admin has installed Enterprise Steam (for example, http://192.16.2.182:9000). Contact your Admin for the IP address and for your login credentials.

2.1 The Enterprise Steam UI

The first time you log in to Enterprise Steam, an empty Enterprise Steam page will display.
The left navigation provides quick links for all the following:

- Cluster details
- Configurations (for generating a Personal Access Token)
- An e-mail link to Enterprise Steam support at H2O
- A logout button
CLUSTERS AND NOTEBOOKS

The **Clusters** page shows clusters created by the current user, the state of the cluster, the cluster type, and the cluster creation date. From this page, you can launch a new cluster, view details of a cluster, or delete a cluster. You can also click the cluster name beside a “Started” cluster to access H2O Flow (see Using Enterprise Steam with H2O Flow).

**Note:** When Enterprise Steam is started for the first time, no clusters will appear in the UI.

### 3.1 Launching a New Cluster

The options for launching a new cluster vary based on the selected cluster Type.

#### 3.1.1 Launch a New H2O Cluster

1. In the Enterprise Steam UI, navigate to the **Clusters** page and select **Launch New Cluster**.
2. Select **H2O** from the Cluster Type dropdown menu.
3. Select a Cluster Profile from the dropdown menu to use when setting up the new cluster. Cluster profiles are configured by the Admin on the Configurations page and provide the allowed min and max values for each option in a cluster profile.
4. Specify values for the options below. Once added, other Enterprise Steam users will be able to connect to this cluster.

- **Cluster Name**: Specify a name for this cluster.
- **H2O Version**: For new H2O clusters, specify the H2O version to use.
- **Number of Nodes**: Specify the number of nodes for the cluster.
- **Java Memory per Node [GB]**: Specify the amount of memory that should be available on each node.
- **YARN Virtual Cores per Node**: Specify the number of virtual cores.
- **H2O Threads per Node**: Specify the number of threads (CPUs) to use in the cluster. Leave this blank to use all available threads.
- **Extra Memory**: Specify the amount of extra memory for internal JVM use outside of the Java heap. This is a percentage of memory per node. The default (and recommended) value is 10%.
- **Maximum Idle Time [HRS]**: Specify the maximum number of hours that the cluster can be idle before gracefully shutting down. Leave this blank to turn off this setting and allow the cluster to remain idle for an unlimited amount of time.
- **Maximum Uptime [HRS]**: Specify the maximum number of hours that the cluster can be running. Leave this blank to turn off this setting and allow the cluster to remain up for an unlimited amount of time.
- **Leader Node ID**: Optionally specify whether to connect to a different leader node.
- **YARN Queue**: If your cluster contains queues for allocating cluster resources, optionally specify a queue for this cluster. Note that the YARN Queue cannot contain spaces. Leave this empty to use the default YARN queue.
5. Click the Launch New Cluster button to start the new cluster. Upon successful completion, the cluster will appear on the Clusters page.

### 3.1.2 Launch a New Sparkling Water - Internal Backend Cluster

1. In the Enterprise Steam UI, navigate to the Clusters page and select Launch New Cluster.
2. Select Sparkling Water - Internal Backend from the Cluster Type dropdown menu.
3. Select a Cluster Profile from the dropdown menu to use when setting up the new cluster. Cluster profiles are configured by the Admin on the Configurations page and provide the allowed min and max values for each option in a cluster profile.
4. Specify values for the options below. Once added, other Enterprise Steam users will be able to connect to this cluster.
   - **Cluster Name**: Specify a name for this cluster.
   - **Sparkling Water Version**: For new Sparkling Water clusters, specify the Sparkling Water version to use.
   - **Driver Cores**: Specify the number of driver cores for the cluster.
• **Driver Memory [GB]:** Specify the amount of driver memory that should be available on each core.
• **Number of Executors:** Specify the number of cores per executor.
• **Executor Cores:** Specify the number of cores per executor.
• **Executor Memory:** Specify the amount of executor memory per node (in GB).
• **H2O Threads per Node:** Specify the number of threads (CPUs) to use in the cluster. 0 indicates to use all available threads.
• **Startup Timeout [SEC]:** Specify the startup timeout in seconds. The cluster will terminate if it cannot start within this time.
• **YARN Queue:** If your cluster contains queues for allocating cluster resources, optionally specify a queue for this cluster. Note that the YARN Queue cannot contain spaces. Leave this empty to use the default YARN queue.
• **PySpark Python Path:** If you have a custom Python environment, you can specify the path here; otherwise, leave empty to use the default Python path.
• **Spark Properties:** This shows a list of additional Spark properties for the cluster. This is maintained on by Admins from the Profiles tab.

5. Click the **Launch New Cluster** button to start the new cluster.

Upon successful completion, the cluster will appear on the **Clusters** page.

### 3.1.3 Launch a New Sparkling Water - External Backend Cluster

1. In the Enterprise Steam UI, navigate to the **Clusters** page and select **Launch New Cluster**.
2. Select **Sparkling Water - External Backend** from the Cluster Type dropdown menu.
3. Select a Cluster Profile from the dropdown menu to use when setting up the new cluster. Cluster profiles are configured by the Admin on the Configurations page and provide the allowed min and max values for each option in a cluster profile.
4. Specify values for the options below. Once added, other Enterprise Steam users will be able to connect to this cluster.
   • **Cluster Name:** Specify a name for this cluster.
   • **Sparkling Water Version:** For new Sparkling Water clusters, specify the Sparkling Water version to use.
   • **Driver Cores:** Specify the number of driver cores for the cluster.
   • **Driver Memory [GB]:** Specify the amount of driver memory that should be available on each core.
   • **Number of Executors:** Specify the number of cores per executor.
   • **Executor Cores:** Specify the number of cores per executor.
   • **Executor Memory:** Specify the amount of executor memory per node (in GB).
   • **H2O Nodes:** Specify the number of H2O nodes.
   • **H2O Memory Per Node [GB]:** Specify the amount of memory in GB to allocate to H2O for each node.
   • **H2O Threads per Node:** Specify the number of threads (CPUs) to use in the cluster. 0 indicates to use all available threads.
   • **Startup Timeout [SEC]:** Specify the startup timeout in seconds. The cluster will terminate if it cannot start within this time.
• **YARN Queue**: If your cluster contains queues for allocating cluster resources, optionally specify a queue for this cluster. Note that the YARN Queue cannot contain spaces. Leave this empty to use the default YARN queue.

• **PySpark Python Path**: If you have a custom Python environment, you can specify the path here; otherwise, leave empty to use the default Python path.

• **Spark Properties**: This shows a list of additional Spark properties for the cluster. This is maintained on by Admins from the Profiles tabl.

5. Click the **Launch New Cluster** button to start the new cluster.

Upon successful completion, the cluster will appear on the **Clusters** page.

### 3.2 Deleting Clusters

Click the **Delete** icon beside the cluster that you want to delete. A confirmation message will display. Click **Confirm** to continue. This action stops and then removes the cluster.

### 3.3 Notebooks

Enterprise Steam allows you to upload and run PySparkling and RSparkling Jupyter Notebooks from within the cluster. More information about Jupyter Notebooks is available here: [https://jupyter.org/](https://jupyter.org/).

#### 3.3.1 Requirements for RSparkling Jupyter Notebooks

• The Hadoop cluster must have R installed along with devtools and sparklyr libraries.

• In Steam you must have an H2O engine uploaded as well as Sparkling Water engine that was built for the same H2O version. The supported minimum Sparkling Water versions for this feature include:
3.3.2 Launching Notebooks

In the Enterprise Steam UI, navigate to the Clusters page and click the My Notebooks button in the upper-right corner to view available notebooks.

3.3.3 Creating New Notebooks

On the Jupyter Notebook Files tab, click New dropdown and select the type of notebook or other file that you want to create. Then create and save the new notebook.

3.3.4 Adding Notebooks

1. On the Jupyter Notebook Files tab, click the Upload button.
2. Browse to the location on your local machine where your notebooks is stored.
3. Click Upload to complete the notebook upload process.
The Configurations page allows you to create your own personal access tokens for use in scripts and on the command line. **Note:** Be careful, these tokens are like passwords so you should guard them carefully. The advantage to using a token over putting your password into a script is that a token can be revoked.

Click **Generate New Token** to generate and retrieve your token. **Note:** For security reasons the token will be shown only once after generating. If you lose your token, you must generate a new one. You can only have one token at a time.
As with other H2O products, Flow can be used alongside Enterprise Steam when performing machine learning tasks. On the Clusters page, click the cluster name of the H2O cluster that you want to open.

This opens H2O Flow in a new tab.
5.1 The H2O Flow UI

Use the menu items at the top to import/upload your data into Flow and to build and score models.

- The **Data** dropdown allows you to import or upload a dataset, import SQL table, split or merge frames, and impute data.
• Use the Model dropdown to select an algorithm and begin building models or to import/export models.
Refer to the H2O Flow documentation for more information on how to use Flow.
Enterprise Steam provides several Python and R functions that can be used for logging in, creating new clusters, and connecting to existing clusters. Select one of the topics below to view an end-to-end example.

### 6.1 Using Enterprise Steam with Python

This section describes how to use the Enterprise Steam for Python. Note that each Python request will result in a warning message. These warnings can be ignored.

#### 6.1.1 Downloading and Installing

2. On the Steam API tab, select Python package that you want to download.
3. Open a Terminal window, and navigate to the location where the Python package file was downloaded. For example:
   ```bash
   cd ~/Downloads
   ```
4. Install Enterprise Steam for Python using one of the following methods:
   ```bash
   # Install Python whl
   pip install h2osteam-1.4.9-py2.py3-none-any.whl
   
   # Install Conda tar.bz2
   # Replace version below with your desired Conda package/Python version
   conda install h2osteam-1.4.9-py27_0.tar.bz2
   ```

#### 6.1.2 login

In Python, use the `login` function to log in to your Enterprise Steam web server. Note that you must already have a username and a password. The web server and your username and password are provided to you by your Enterprise Steam Admin.

```python
$ python
>>> import h2osteam
>>> conn = h2osteam.login(url = "https://steam.0xdata.loc",
                        verify_ssl = False,
                        (continues on next page)
```
6.1.3 start_h2o_cluster

Use the `start_h2o_cluster` function to create a new cluster. This function takes the following parameters:

- **cluster_name**: Specify a name for this cluster.
- **profile_name**: Specify the profile to use for this cluster.
- **num_nodes**: Specify the number of nodes for the cluster.
- **node_memory**: Specify the amount of memory that should be available on each node.
- **v_cores**: Specify the number of virtual cores.
- **n_threads**: Specify the number of threads (CPUs) to use in the cluster. Specify 0 to use all available threads.
- **max_idle_time**: Specify the maximum number of hours that the cluster can be idle before gracefully shutting down. Specify 0 to turn off this setting and allow the cluster to remain idle for an unlimited amount of time.
- **max_uptime**: Specify the maximum number of hours that the cluster can be running. Specify 0 to turn off this setting and allow the cluster to remain up for an unlimited amount of time.
- **extramempercent**: Specify the amount of extra memory for internal JVM use outside of the Java heap. This is a percentage of memory per node. The default (and recommended) value is 10%.
- **h2o_version**: The H2O engine version that this cluster will use. Note that the Enterprise Steam Admin is responsible for adding engines to Enterprise Steam.
- **yarn_queue**: If your cluster contains queues for allocating cluster resources, specify the queue for this cluster. Note that the YARN Queue cannot contain spaces.
- **callback_ip**: Optionally specify the IP address for callback messages from the mapper to the driver (driverif).

```python
>>> cluster_config = conn.start_h2o_cluster(cluster_name = 'first-cluster-from-Python',
                                           profile_name = 'default',
                                           num_nodes = 2,
                                           node_memory = '30g',
                                           h2o_version = '3.22.0.1',
                                           max_idle_time = 1,
                                           max_uptime = 1)
```

# Call the cluster to retrieve its ID and configuration params.
```python
>>> cluster_config
{'id': 107, 'connect_params': {'cookies': ['first-cluster-from-Python=YW5nZWxhOm9mdrZm53aGJsdWY=', 'ip': 'steam.0xdata.loc', 'context_path': 'jsmit_first-cluster-from-Python', 'verify_ssl_certificates': False, 'https': True, 'port': 9999}}
```

Note that after you create a cluster, you can immediately connect to that cluster and begin using H2O. Refer to the following for a complete Python example.
>>> import h2o
>>> from h2o.estimators.gbm import H2OGradientBoostingEstimator
>>> h2o.connect(config = cluster_config)

# import the cars dataset
# this dataset is used to classify whether or not a car is economical based on
# the car's displacement, power, weight, and acceleration, and the year it was made
>>> cars = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/
˓→junit/cars_20mpg.csv")

# convert response column to a factor
>>> cars["economy_20mpg"] = cars["economy_20mpg"].asfactor()

# set the predictor names and the response column name
>>> predictors = ["displacement","power","weight","acceleration","year"]
>>> response = "economy_20mpg"

# split into train and validation sets
>>> train, valid = cars.split_frame(ratios = [.8], seed = 1234)

# initialize your estimator
>>> cars_gbm = H2OGradientBoostingEstimator(seed = 1234)

# train your model, specifying your 'x' predictors,
# your 'y' the response column, training_frame, and validation_frame
>>> cars_gbm.train(x = predictors, y = response, training_frame = train, validation_˓→frame = valid)

# print the auc for the validation data
>>> cars_gbm.auc(valid=True)

6.1.4 get_h2o_cluster

Use the get_h2o_cluster function to retrieve information about a specific cluster using the cluster name.

>>> conn.get_h2o_cluster('first-cluster-from-Python')
{'id': 108, 'connect_params': {'cookies': [u'first-cluster-from-Python=YW5nZWxhOnA1bHRreHN5am0=', 'ip': 'steam.0xdata.loc', 'context_path': u˓→jsmith_first-cluster-from-Python', 'verify_ssl_certificates': False, 'https': True, ˓→'port': 9999}}

6.1.5 get_h2o_clusters

Use the get_h2o_clusters function to retrieve all running H2O clusters accessible to current user

>>> conn.get_h2o_clusters()

6.1.6 stop_h2o_cluster

Use the stop_h2o_cluster function to stop a cluster.

>>> conn.stop_h2o_cluster(cluster_config)
6.1.7 show_profiles

Use the `show_profiles` to show available profiles.

```python
>>> conn.show_profiles(cluster_config)
```

6.1.8 start_internal_sparkling_cluster

Use the `start_internal_sparkling_cluster` function to create a new sparkling water cluster using internal backend. This function takes the following parameters:

- `cluster_name`: Specify a name for this cluster.
- `profile_name`: Specify the profile to use for this cluster.
- `h2o_version`: The H2O engine version that this cluster will use. Note that the Enterprise Steam Admin is responsible for adding engines to Enterprise Steam.
- `driver_cores`: Number of Spark driver cores
- `driver_memory_gb`: Amount of Spark driver memory in GB
- `num_executors`: Number of Spark executors
- `executor_cores`: Number of Spark executor cores
- `executor_memory_gb`: Amount of Spark executor memory in GB
- `h2o_node_threads`: Specify the number of threads (CPUs) to use per node. Specify 0 to use all available threads.
- `start_timeout_sec`: Specify start timeout in seconds
- `yarn_queue`: If your cluster contains queues for allocating cluster resources, specify the queue for this cluster. Note that the YARN Queue cannot contain spaces.
- `spark_properties`: Specify additional spark properties as a Python dictionary.

```python
>>> cluster = conn.start_internal_sparkling_cluster(cluster_name="test",
                      profile_name="default-sparkling-internal",
                      h2o_version="3.22.0.1",
                      driver_cores=1,
                      driver_memory_gb=1,
                      num_executors=1,
                      executor_cores=1,
                      executor_memory_gb=1,
                      h2o_node_threads=0,
                      start_timeout_sec=90,
                      yarn_queue=None,
                      spark_properties={"spark.python.worker.reuse": 'true', 'key': 'val'}
```

6.1.9 start_external_sparkling_cluster

Use the `start_external_sparkling_cluster` function to create a new sparkling water cluster using external backend. This function takes the following parameters:

- `cluster_name`: Specify a name for this cluster.
- **profile_name**: Specify the profile to use for this cluster.
- **h2o_version**: The H2O engine version that this cluster will use. Note that the Enterprise Steam Admin is responsible for adding engines to Enterprise Steam.
- **driver_cores**: Number of Spark driver cores
- **driver_memory_gb**: Amount of Spark driver memory in GB
- **num_executors**: Number of Spark executors
- **executor_cores**: Number of Spark executor cores
- **executor_memory_gb**: Amount of Spark executor memory in GB
- **h2o_nodes**: Specify the number of H2O nodes for the cluster.
- **h2o_node_memory_gb**: Specify the amount of memory that should be available on each H2O node.
- **h2o_node_threads**: Specify the number of threads (CPUs) to use per node. Specify 0 to use all available threads.
- **start_timeout_sec**: Specify start timeout in seconds
- **yarn_queue**: If your cluster contains queues for allocating cluster resources, specify the queue for this cluster. Note that the YARN Queue cannot contain spaces.
- **spark_properties**: Specify additional spark properties as a Python dictionary.

```python
>>> cluster = conn.start_external_sparkling_cluster(cluster_name="test",
profile_name="default-sparkling-external",
h2o_version="3.22.0.1",
driver_cores=1,
driver_memory_gb=1,
num_executors=1,
executor_cores=1,
executor_memory_gb=1,
h2o_nodes=1,
h2o_node_memory_gb=1,
h2o_node_threads=0,
start_timeout_sec=90,
yarn_queue=None,
spark_properties={'spark.python.worker.reuse': 'true', 'key': 'val'})
```

### 6.1.10 sparkling_cluster.session

Use the `session` function of sparkling water cluster to connect to the remote spark session and issue commands.

```python
>>> sparkling_cluster = conn.start_internal_sparkling_cluster(.....)
>>> sparkling_cluster.session()
```

### 6.1.11 sparkling_cluster.send_statement

Use the `send_statement` function of sparkling water cluster to send a single statement to the remote spark session.

```python
>>> sparkling_cluster = conn.start_internal_sparkling_cluster(.....)
>>> sparkling_cluster.send_statement("f_crimes = h2o.import_file(path = "/data/ChicagoCrimes10k.csv", col_types =column_type")
```

(continues on next page)
6.1.12 **sparkling_cluster.detail**

Use the `detail` function of sparkling water cluster to get an information about that sparkling water cluster.

```python
>>> sparkling_cluster = conn.start_internal_sparkling_cluster(.......)
>>> sparkling_cluster.detail()
```

6.1.13 **sparkling_cluster.stop**

Use the `stop` function of sparkling water cluster to stop the cluster.

```python
>>> sparkling_cluster = conn.start_internal_sparkling_cluster(.......)
>>> sparkling_cluster.stop()
```

6.1.14 **upload_engine**

Use the `upload_engine` function to upload H2O engine to Steam.

```python
>>> conn.upload_engine("~/Downloads/h2o-3.22.0.1-hdp2.4.zip")
```

6.1.15 **upload_sparkling_engine**

Use the `upload_sparkling_engine` function to upload Sparkling Water engine to Steam.

```python
>>> conn.upload_sparkling_engine("~/Downloads/sparkling-water-2.3.17.zip")
```

6.2 Using Enterprise Steam with R

This section describes how to use the Enterprise Steam for R. Note that this requires “urltools”. Refer to https://github.com/Ironholds/urltools/ for more information.

6.2.1 Downloading and Installing

2. On the Steam API tab, download the R package.
3. Open a Terminal window, and navigate to the location where the Enterprise Steam file was downloaded. For example:

   ```
   cd ~/Downloads
   ```

4. Install Enterprise Steam for R using `R CMD INSTALL <file_name>`. For example:
6.2.2 login

Use the `login` function to log in to your Enterprise Steam web server. Note that you must already have a username and a password. The web server and your username and password are provided to you by your Enterprise Steam Admin. This function takes the following parameters:

- `url`: The URL of the Enterprise Steam instance
- `verify_ssl`: Specify True or False to verify SSL certificate
- `username`: Your username as provided by your Enterprise Steam Admin
- `password`: Your password as provided by your Enterprise Steam Admin
- `login_file`: A login file where user information is stored.
- `login_file_passphrase`: A login file where user passphrase information is stored.

```r
$r
> library(h2osteam)
> conn <- h2osteam.login(url = "https://steam.0xdata.loc",
                       verify_ssl = F,
                       username="jsmith",
                       password="jsmith")
```

6.2.3 start_h2o_cluster

Use the `start_h2o_cluster` function to create a new cluster. This function takes the following parameters:

- `cluster_name`: Specify a name for this cluster.
- `profile_name`: Specify the profile to use for this cluster.
- `num_nodes`: Specify the number of nodes for the cluster.
- `node_memory`: Specify the amount of memory that should be available on each node.
- `v_cores`: Specify the number of virtual cores.
- `n_threads`: Specify the number of threads (CPUs) to use in the cluster. Specify 0 to use all available threads.
- `max_idle_time`: Specify the maximum number of hours that the cluster can be idle before gracefully shutting down. Specify 0 to turn off this setting and allow the cluster to remain idle for an unlimited amount of time.
- `max_uptime`: Specify the maximum number of hours that the cluster can be running. Specify 0 to turn off this setting and allow the cluster to remain up for an unlimited amount of time.
- `extramempercent`: Specify the amount of extra memory for internal JVM use outside of the Java heap. This is a percentage of memory per node. The default (and recommended) value is 10%.
- `h2o_engine_id`: The H2O engine version that this cluster will use. Note that the Enterprise Steam Admin is responsible for adding engines to Enterprise Steam.
- `yarn_queue`: If your cluster contains queues for allocating cluster resources, specify the queue for this cluster. Note that the YARN Queue cannot contain spaces.
> cluster_config <- h2osteam.start_h2o_cluster(conn = conn,
cluster_name = "first-cluster-from-R",
profile_name = "default",
num_nodes = 2,
node_memory = "30g",
h2o_version = "3.22.0.1",
max_idle_time = 1,
max_uptime = 1)

# Call the cluster to retrieve its ID and configuration params.
> cluster_config
$id
[1] 109

$connect_params
$connect_params$ip
[1] "steam.0xdata.loc"

$connect_params$port
[1] 9999

$connect_params$cookies
[1] "first-cluster-from-R=YW5nZWxhOnVoYzdyeTNtM3g="

$connect_params$context_path
[1] "jsmith_first-cluster-from-R"

$connect_params$https
[1] TRUE

$connect_params$insecure
[1] TRUE

Note that after you create a cluster, you can immediately connect to that cluster and begin using H2O. Refer to the following for a complete R example.

> library(h2o)
> h2o.connect(config = cluster_config)

# import the cars dataset
# this dataset is used to classify whether or not a car is economical based on
# the car’s displacement, power, weight, and acceleration, and the year it was made
> cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/
˓
→junit/cars_20mpg.csv")

# convert response column to a factor
> cars["economy_20mpg"] <- as.factor(cars["economy_20mpg"])

# set the predictor names and the response column name
> predictors <- c("displacement","power","weight","acceleration","year")
> response <- "economy_20mpg"

# split into train and validation sets
> cars.split <- h2o.splitFrame(data = cars,ratios = 0.8, seed = 1234)
> train <- cars.split[[1]]
> valid <- cars.split[[2]]
# train your model, specifying your 'x' predictors, # your 'y' the response column, training_frame, and validation_frame
> cars_gbm <- h2o.gbm(x = predictors,
    y = response,
    training_frame = train,
    validation_frame = valid,
    seed = 1234)

# print the auc for your model
> print(h2o.auc(cars_gbm, valid = TRUE))

6.2.4 get_h2o_cluster

Use the get_h2o_cluster to retrieve information about a specific cluster using the cluster name.

> h2osteam.get_h2o_cluster(conn, 'first-cluster-from-R')
$id
[1] 109

$connect_params
$connect_params$ip
[1] "steam.0xdata.loc"

$connect_params$port
[1] 9999

$connect_params$cookies
[1] "first-cluster-from-R=YW5nZWxhOnVoYzdyeTNtM3g="

$connect_params$context_path
[1] "jsmith_first-cluster-from-R"

$connect_params$https
[1] TRUE

$connect_params$insecure
[1] TRUE

6.2.5 get_h2o_clusters

Use the get_h2o_clusters to retrieve all running H2O clusters accessible to current user

> h2osteam.get_h2o_clusters(conn)

6.2.6 stop_h2o_cluster

Use the stop_h2o_cluster function to stop a cluster.

> h2osteam.stop_h2o_cluster(conn, cluster_config)
6.2.7 show_profiles

Use the `show_profiles` to show available profiles.

```python
> h2osteam.show_profiles(conn)
```