# Enterprise Steam User Guide

*Release 1.4.1.74*

**H2O.ai**

**Dec 03, 2018**

# CONTENTS

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 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.

CHAPTER

ONE

# ENTERPRISE STEAM RELEASE NOTES

## 1.1 Change Log

### 1.1.1 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

### 1.1.2 Version 1.4.0 (Nov 23, 2018)

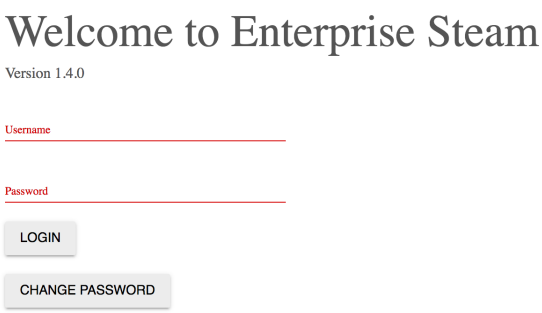
- Add Sparkling Water integration
- Add SAML authentication
- Add More detailed cluster profiles
- New cluster overview
- New launch cluster page
- Add option to generate Personal access tokens

CHAPTER

TWO

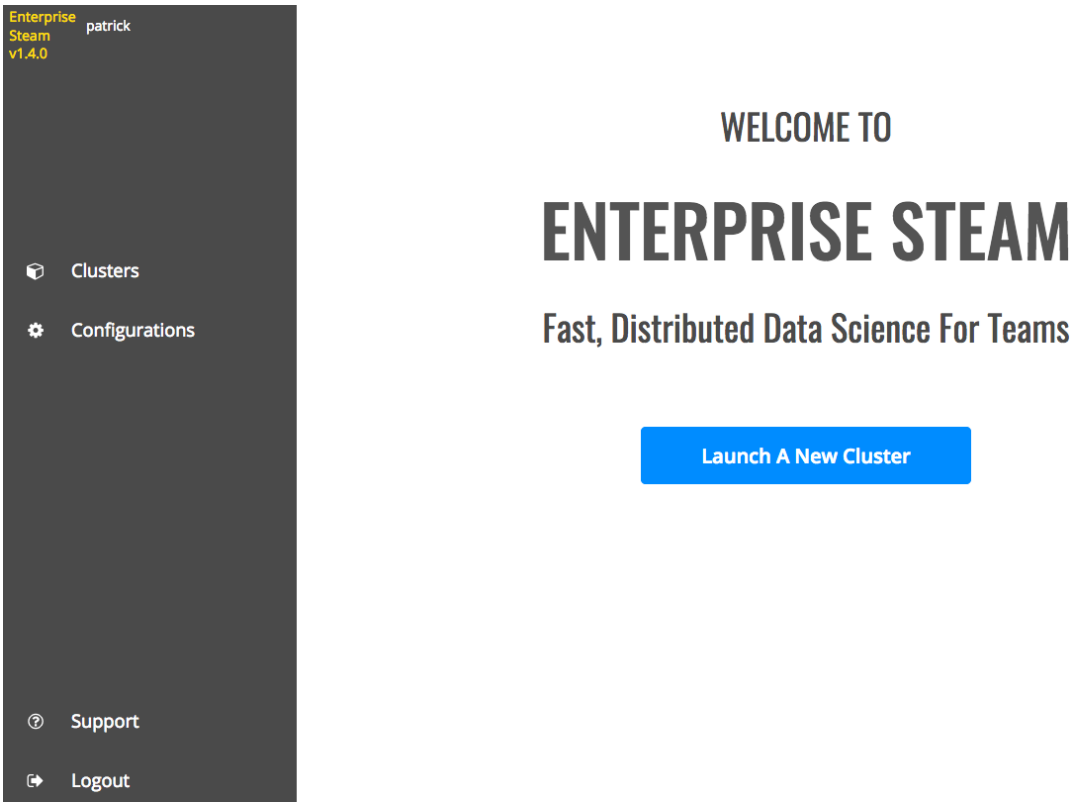
# LOGGING IN TO ENTERPRISE STEAM

In a Chrome web browser, 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

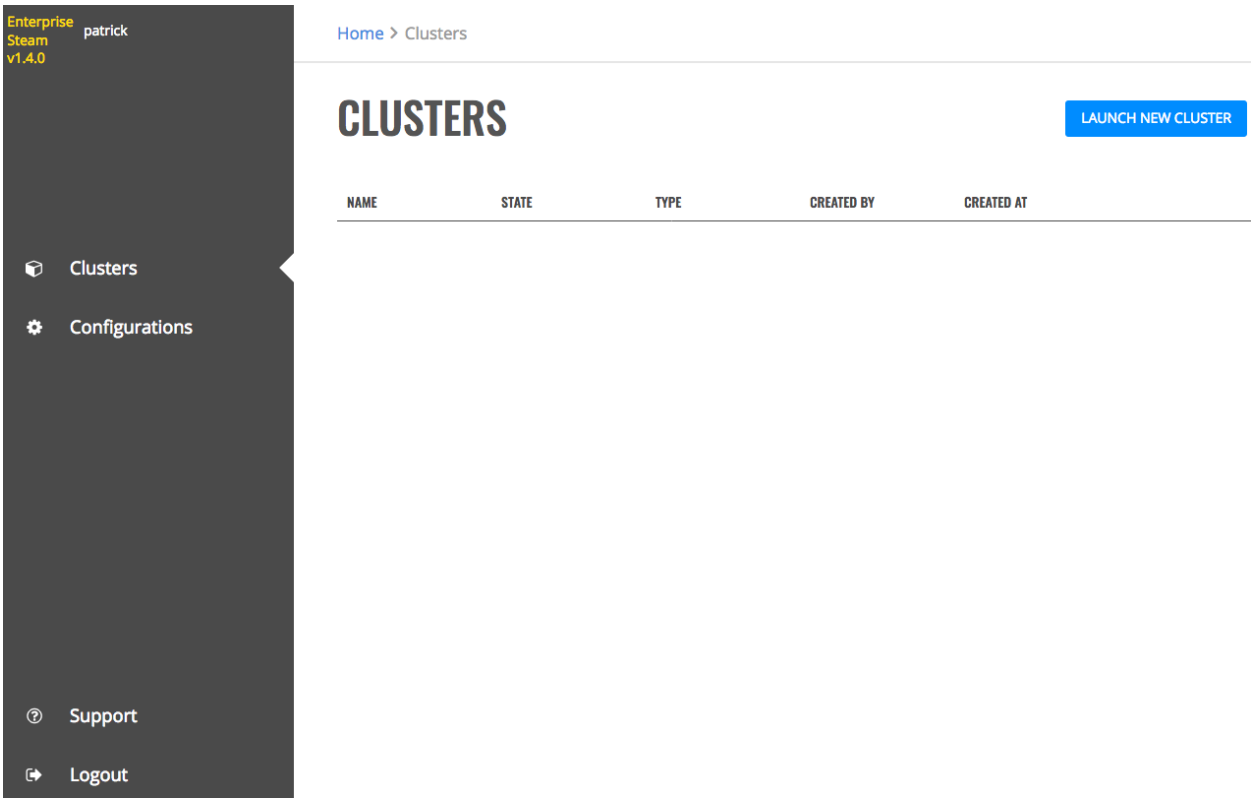
CHAPTER

THREE

# CLUSTERS

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.

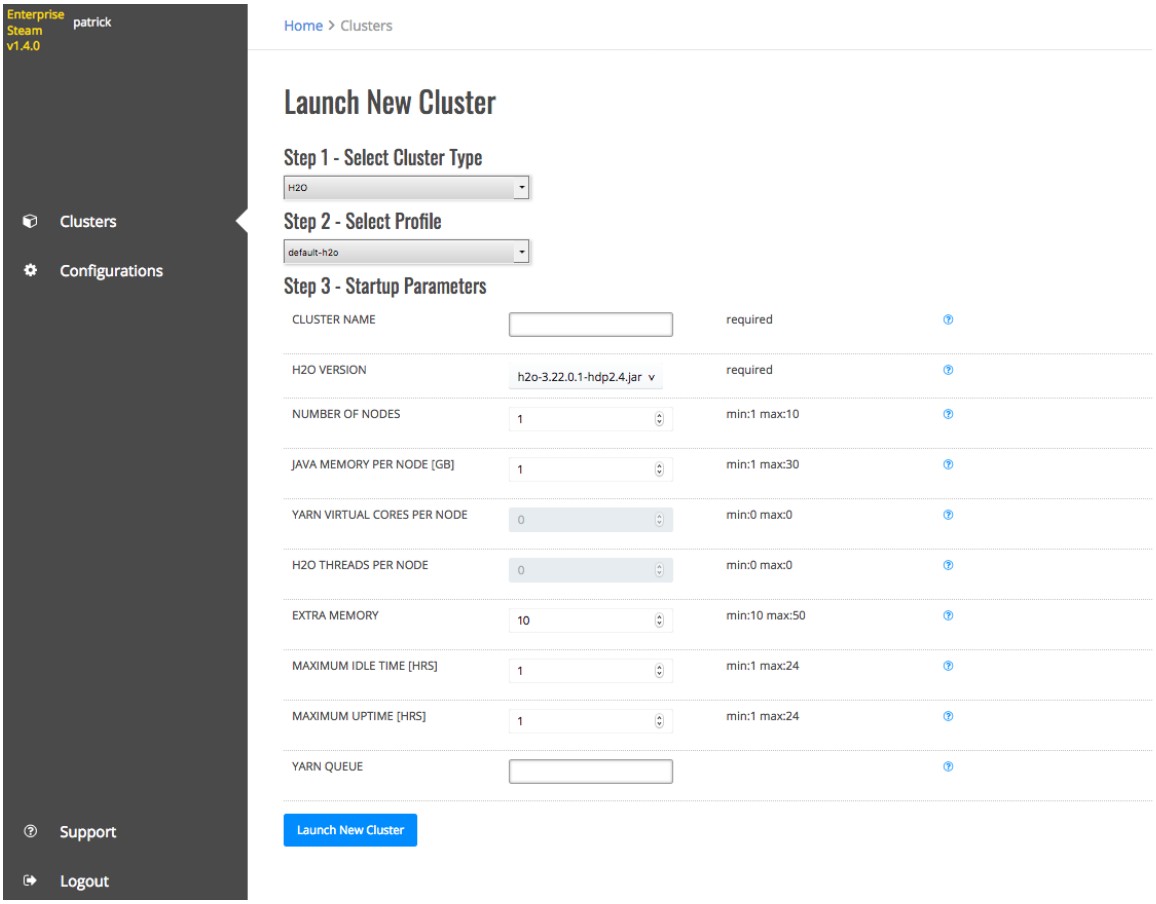



## 3.1 Launch a New Cluster

1. In the Enterprise Steam UI, navigate to the **Clusters** page and select **Launch New Cluster**.
2. Select the Cluster Type from the 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 options 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**: 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.
- **YARN Queue**: If your cluster contains queues for allocating cluster resources, optioanlly specify a queue for this cluster. Note that the YARN Queue cannot contain spaces. Leave this empty to use the default YARN queue.

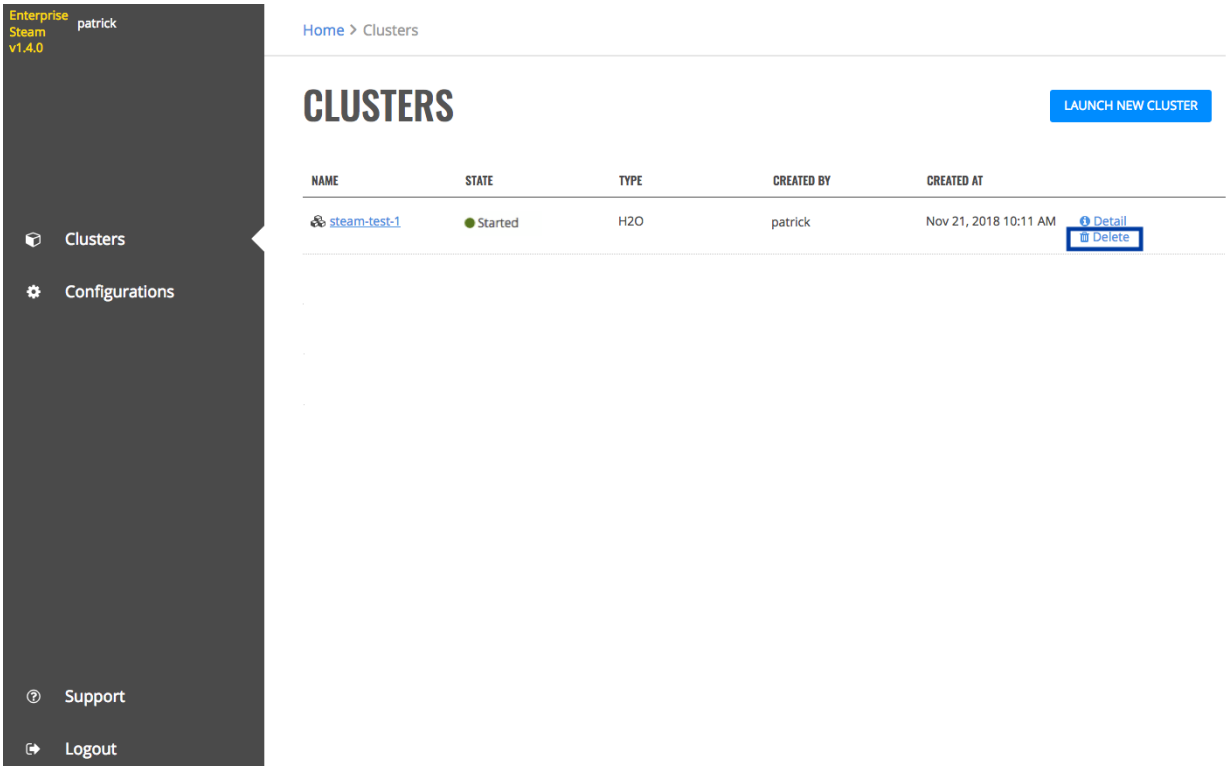


5. Click the **Launch New Cluster** button to start the new cluster.

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

## 3.2 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.



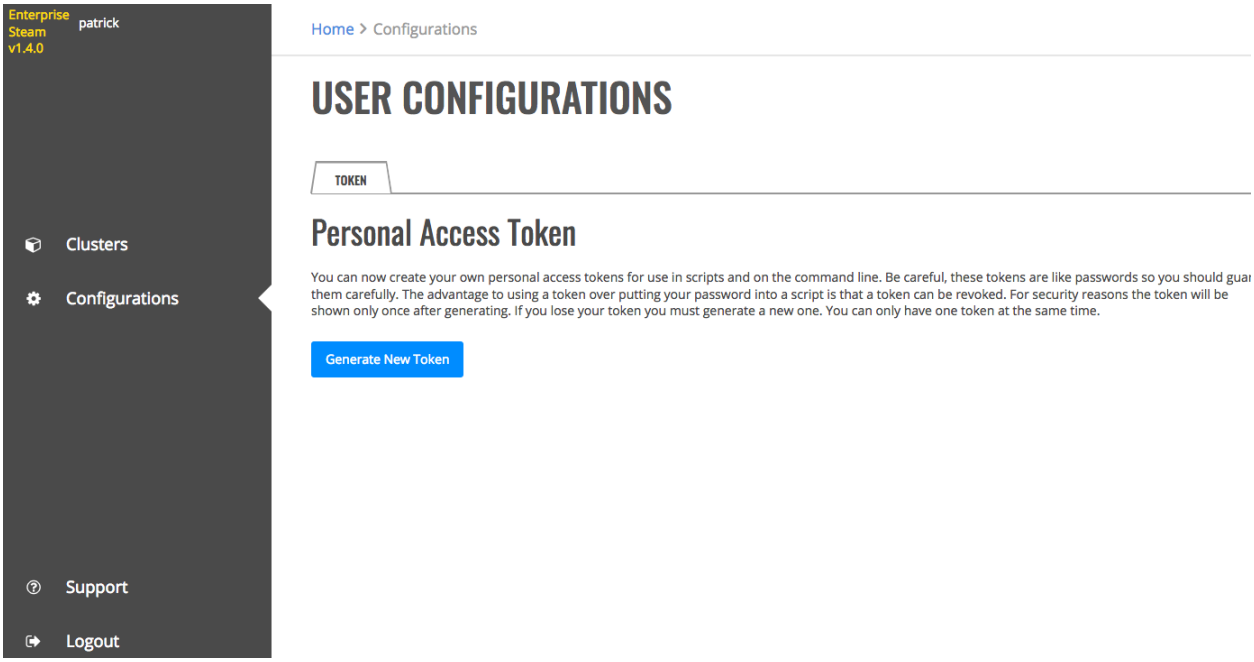
CHAPTER

FOUR

# CONFIGURATIONS

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 generaate 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.

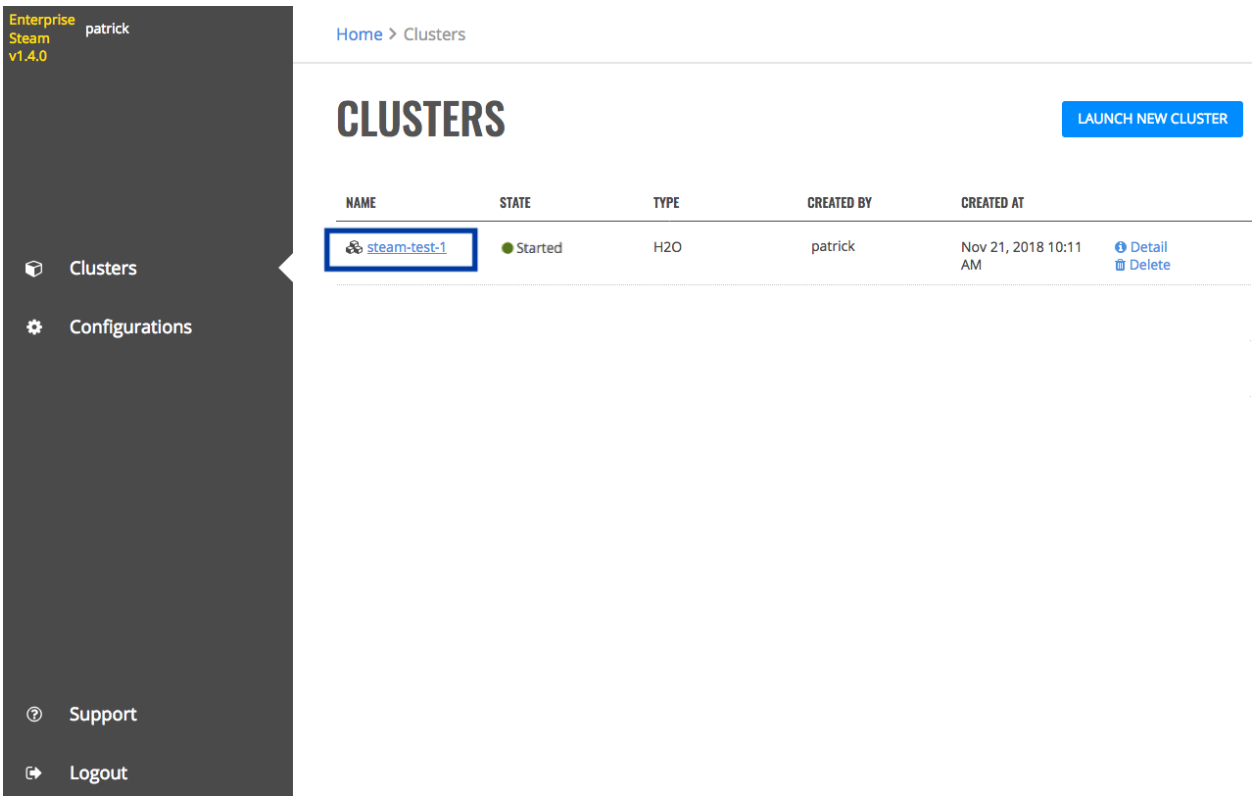


CHAPTER

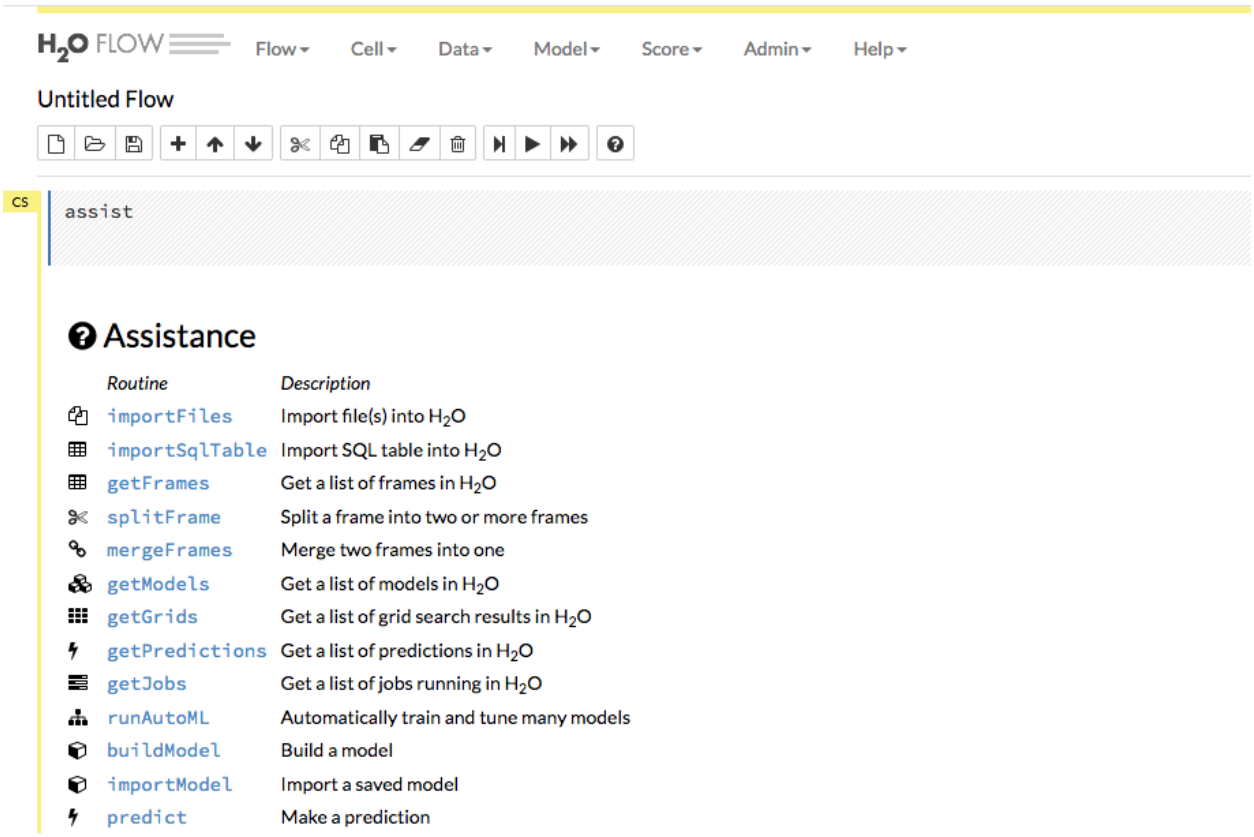
FIVE

# USING ENTERPRISE STEAM WITH H2O FLOW

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.



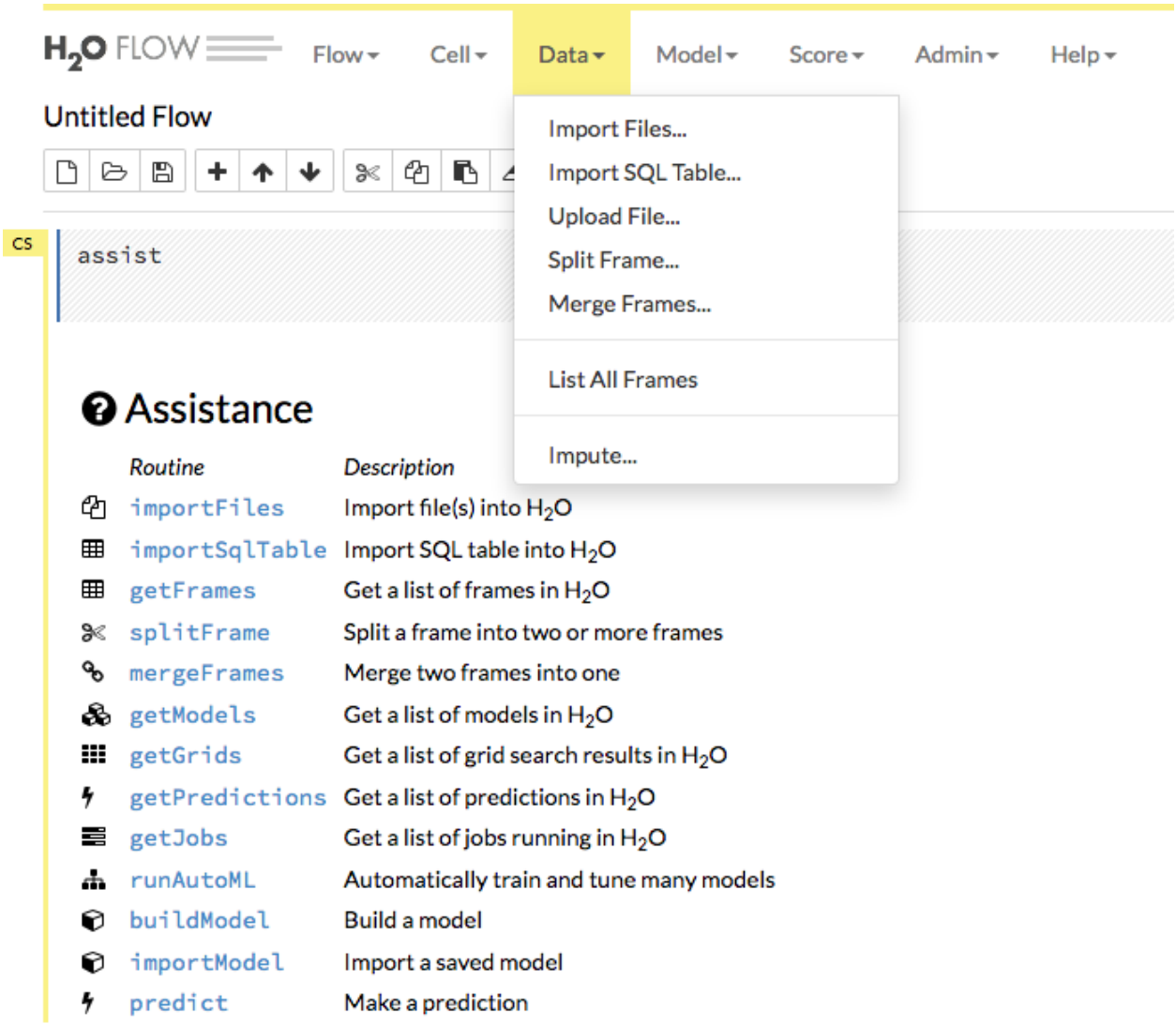

This opens H2O Flow in a new tab.



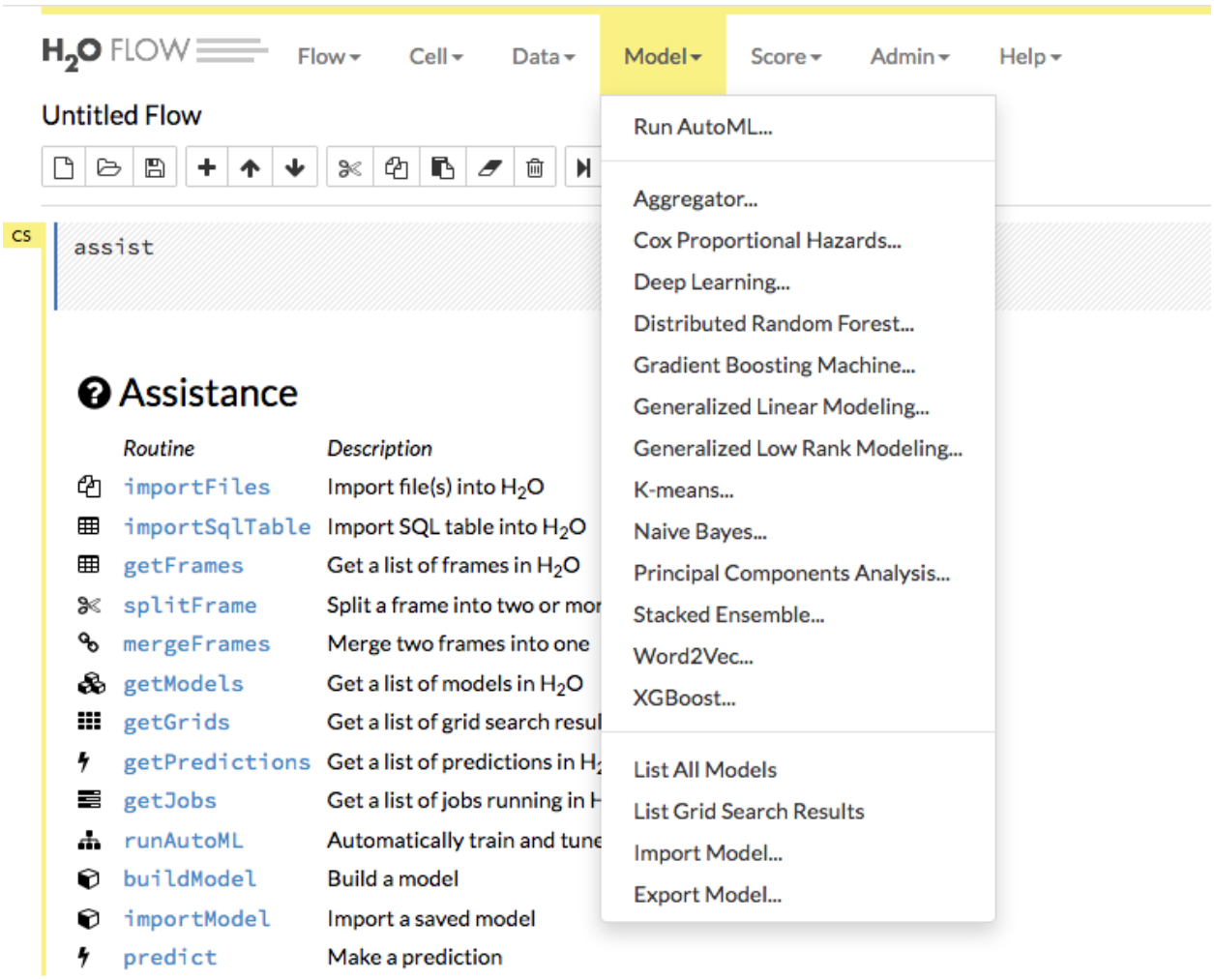

## 5.1 The H2O Flow UI

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.

CHAPTER

SIX

# USING ENTERPRISE STEAM WITH PYTHON/R

Enterprise Steam provides several Python and R functions that can be used for logging in, creating new clusters, ane 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

1. Go to https://s3.amazonaws.com/steam-release/enterprise-steam/latest-stable.html to retrieve the latest version of Enterprise Steam.
2. On the Steam API tab, download the Python package.
3. Open a Terminal window, and navigate to the location where the Python .whl file was downloaded. For example:

```
cd ~/Downloads
```

4. Install Enterprise Steam for Python using `pip install <file_name>`. For example:

```
pip install h2osteam-1.2.0-py2.py3-none-any.whl
```

### 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
>>> import h2osteam
>>> conn = h2osteam.login(url = "https://steam.0xdata.loc",
                          verify_ssl = False,
                          username="jsmith",
                          password="jsmith")
```

### 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).

```
>>> 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")

# Call the cluster to retrieve its ID and configuration params.
>>> cluster_config
{'id': 107, 'connect_params': {'cookies': [u'first-cluster-from-
↪Python=YW5nZWxhOmdrZm53aGJsdWY='], 'ip': 'steam.0xdata.loc', 'context_path': u
↪'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")
```

(continues on next page)

(continued from previous page)

```
# 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` 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=YW5nZWxhOnA1bHRreHN5amo='], '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` 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.

```
>>> 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.

```
>>> cluster = conn.start_internal_sparkling_cluster(cluster_name="test",
                                                    profile_name="default-sparkling-
↪internal",
                                                    h2o_version="3.20.0.9",
                                                    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)
```

### 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.

```
>>> cluster = conn.start_external_sparkling_cluster(cluster_name="test",
                                                    profile_name="default-sparkling-
↪external",
                                                    h2o_version="3.20.0.9",
                                                    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)
```

### 6.1.10 sparkling_cluster.session

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

::

```
>>> 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.

::

```
>>> sparkling_cluster = conn.start_internal_sparkling_cluster(.......)
>>> sparkling_cluster.send_statement()
```

### 6.1.12 sparkling_cluster.detail

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

::

```
>>> 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.

::

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

## 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

1. Go to https://s3.amazonaws.com/steam-release/enterprise-steam/latest-stable.html to retrieve the latest version of Enterprise Steam.
2. On the Steam API tab, download the R package.
3. Open a Terminal window, and navigate to the location where the Enterprse Steam file was downloaded. For example:

```
cd ~/Downloads
```

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

```
R CMD INSTALL h2osteam_1.2.0.tar.gz
```

### 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 provicded 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
> 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")

# 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"
```

(continues on next page)

(continued from previous page)

```
$connect_params$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.

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