

# H<sub>2</sub>O

---

# Steam

Preview Release

---

# Table of Contents

Introduction	1.1
Using Steam	1.2
Creating a Project	1.2.1
Comparing Models	1.2.2
Deploying a Model in Steam	1.2.3
Making Predictions	1.2.4
Using Steam with H2O Flow	1.3
Starting the Steam CLI	1.4
Stopping Steam	1.5
CLI Command Reference Appendix	1.6

# Introduction

Steam is an "instant on" platform that streamlines the entire process of building and deploying applications. It is the industry's first data science hub that lets data scientists and developers collaboratively build, deploy, and refine predictive applications across large scale datasets. Data scientists can publish Python and R code as REST APIs and easily integrate with production applications.

This document describes how to start and use Steam and the Steam Scoring Service. Note that this document assumes that an admin has successfully installed and started Steam on a YARN edge node using the instructions provided in the *Steam Installation and Setup* guide.

**Note:** Before you begin using Steam, be sure that your minimum version of H2O is 3.10.0.3. Earlier versions are not supported. If necessary, follow the instructions on the [H2O Download page](#) for your platform to upgrade H2O.

Refer to the following sections:

- [Starting Steam](#)
- [CLI Command Reference Appendix](#)

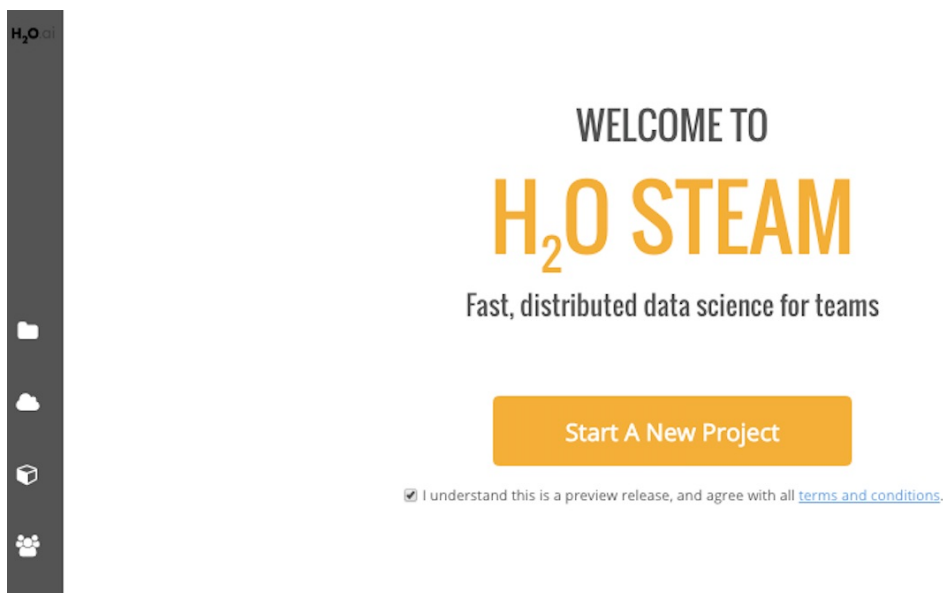
# Using Steam

In a Chrome web browser, navigate to the Steam web server using login credential provided by your admin and/or Steam superuser. This Steam web server is the server on which an admin has installed Steam (for example, <http://192.16.2.182:9000>). Contact your admin for the IP address and for credentials.

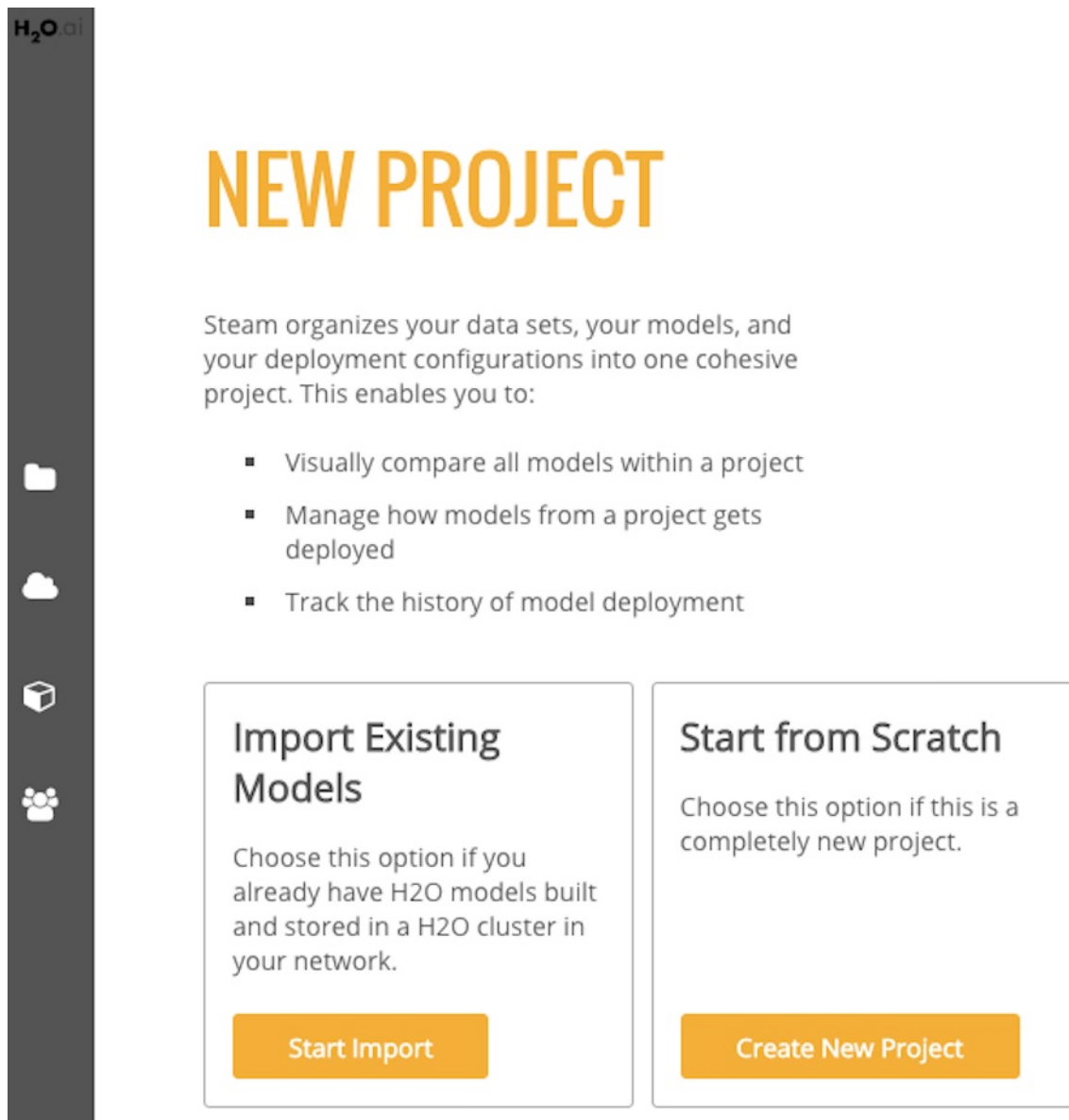
## Creating a Project

Steam makes use of project-based machine learning. Whether you are trying to detect fraud or predict user retention, the datasets, models, and test results are stored and saved in the individual projects. And all Steam users within your environment have access to these projects.

The first time you log in to Steam, an empty Steam page will display, prompting you to start a project. Be sure to accept the terms and conditions in order to continue.



1. Accept the terms of this preview release, then click **Start a New Project**. This opens a page allowing you to start a new project from scratch or to begin importing models into your Steam environment.



2. To start a new project from scratch, click **Create New Project**. This opens a page showing you the available H2O clusters. When you first log in to Steam, the list of clusters will be empty. Enter your cluster IP address, then click **Connect**. Once connected, this will immediately populate the current list of clusters.
3. Select the H2O frame from the Datasets dropdown, then select the Category.
4. Select the checkbox beside the model(s) to import into the Steam project. In this example, two models are available on the H2O cluster: one model built using GBM and one model built using GLM. Both models were built using the "DGA" dataset.
5. Specify a name for the project.

Home > New > Step 1

### 1. SELECT H2O CLUSTER

Select an H2O cluster to import models and datasets from.

CLUSTER	DATASETS	MODELS
technwriter	N/A	N/A

Connect

### ... OR CONNECT TO A NEW H2O CLUSTER

Connect to a H2O cluster where your existing models and data sets are located.

localhost 54321

Connect

### 2. SELECT DATASET

Frame: Key\_Frame\_DGA\_hex

Category: Regression

### 3. PICK MODELS TO IMPORT

Models in a project must share the same feature set and response column to enable comparison.

MODEL	RESPONSE COLUMN	CATEGORICAL	
glm-cb2bcc40-9564-4547-8958-5d10cd04ebe6	1	Regression	<input checked="" type="checkbox"/> Select for Import
gbm-9d972f9c-3e18-4e97-96ab-06b851cb741a	class	Regression	<input checked="" type="checkbox"/> Select for Import

### 4. NAME PROJECT

DGA\_test

Create Project

6. Click **Create Project** when you are done. Upon successful completion, the Models page will be populated with the model(s) that you added to your project.

Home > Projects > 1 > Models

## MODELS

Import Model

filter models

MODEL	TRAIN ROC	TEST ROC	ACTIONS
1st. glm-cb2bcc40-9564-4547-8958-5d10cd04ebe6 technwriter 0			<ul style="list-style-type: none"> <li>view model details</li> <li>label as test</li> <li>deploy model</li> </ul>
2nd. gbm-9d972f9c-3e18-4e97-96ab-06b851cb741a technwriter 0			<ul style="list-style-type: none"> <li>view model details</li> <li>label as test</li> <li>deploy model</li> </ul>

1 - 2 of 2 models

You can perform the following actions directly from the models page:

- Import a new model
- View model details and export the model as a java, jar, or war file
- Label a model as a test, staging, or production model
- Deploy the model

## Comparing Models

Following is an example of the Model Details page for the "GLM-CB2BCC40-9564-4547-8958-5D10CD04EBE6" model.



Home > Projects > 1 > Models > 1

### GLM-CB2BCC40-9564-4547-8958-5D10CD04EBE6

Export Model Deploy Model compared to: SELECT MODEL FOR COMPARISON

#### Model Overview

**Basics**

- Date: 2016-08-08 09:36
- Training Time: 0
- Model Type: Generalized Linear Modeling

**Model Parameters**

- Dataset Name: Key\_Frame\_DGAhex
- Response Column Name: 1

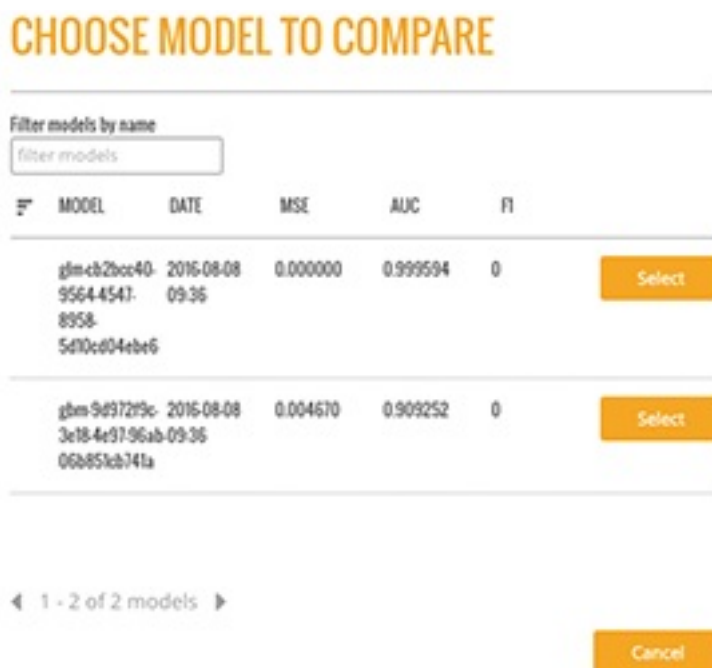
**Goodness of Fit**

**Metrics**

- Mean Squared Error: 3.248932242938112e-7
- R<sup>2</sup>: 0.9995944853732964

As indicated in the previous section, two models were added to this project. From this page, you can compare the GLM and GBM models that were built.

1. Click the **Compared To** field. This opens a popup showing all models available in the current project.



### CHOOSE MODEL TO COMPARE

Filter models by name  
filter models

MODEL	DATE	MSE	AUC	F1	
glm-cb2bcc40-9564-4547-8958-5d10cd04ebe6	2016-08-08 09:36	0.000000	0.999594	0	Select
gbm-9d972f9c-3e18-4e97-96ab-0936-06b85kb741a	2016-08-08 09:36	0.004670	0.909252	0	Select

1 - 2 of 2 models

Cancel

2. Select to compare the current GLM model with the GBM model. Once a model is selected, the Model Details page immediately populates with the comparison information.



## Deploying a Model in Steam

1. On the Models page, click the **deploy model** link for the model that you want to deploy.
2. Specify a service name for the deployment, then click **Deploy**.

**DEPLOY GLM-CB2BCC40-9564-4547-8958-5D10CD04EBE6**

**CONFIGURE SERVICE** Steam automatically selects a port that's not in use based on the port range set by your admin.

Service name

Deploy

Cancel

3. Upon successful completion, a scoring service will be created for this deployed model. Click the **Deployment** menu option on the left navigation to go to the Deployment page.



## Making Predictions

The Deployment page lists all available deployed service.

1. To reach the scoring service, click the IP address link listed under the Deployed Services. This opens Steam Prediction Service tool. The fields that display on the Prediction Service tool are automatically populated with field information from the deployed model.

2. Make predictions by specifying input values based on column data from the original dataset. This automatically populates the fields in the query string. (Note that you can optionally include input parameters directly in the query string instead of specifying parameters.)

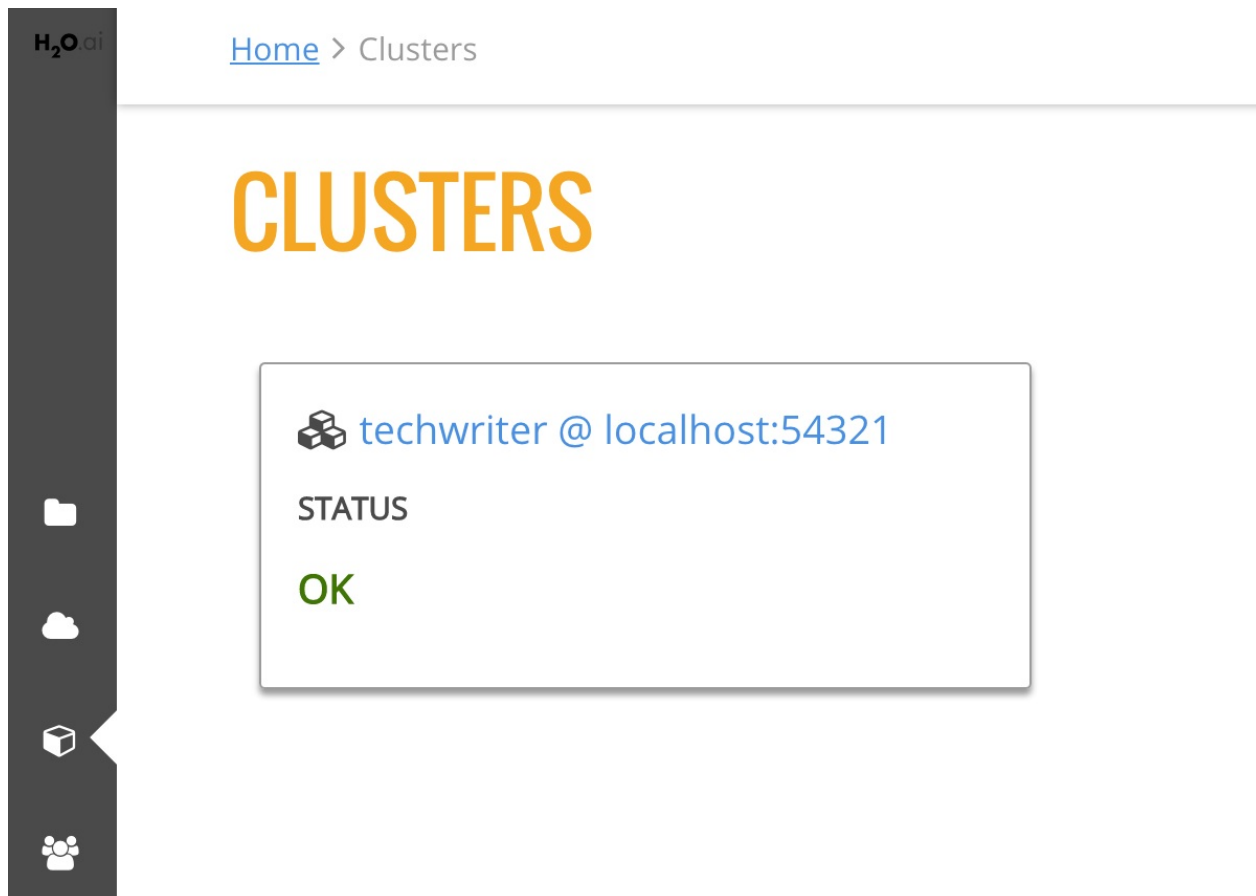
3. Click **Predict** when you are done.

**Note:** Use the **Clear** button to clear all entries and begin a new prediction. Use the **More Stats** button to view additional statistics about the scoring service results.

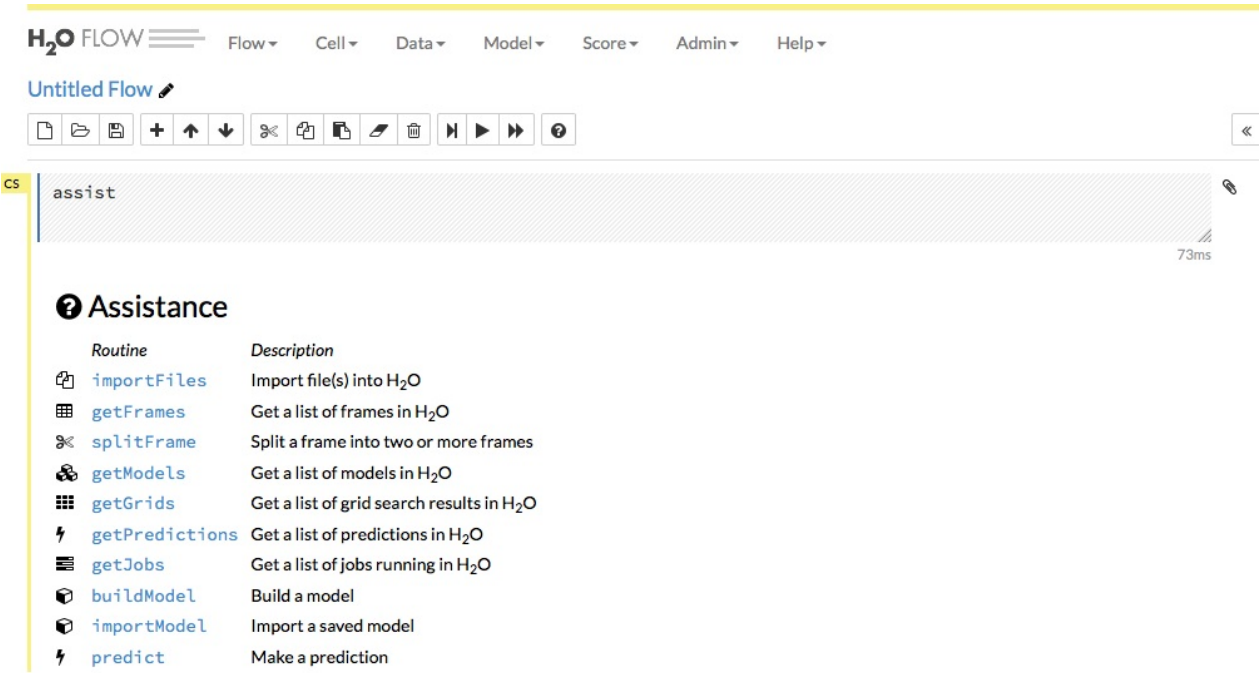
## Using Steam with H2O Flow

As with other H2O products, Flow can be used alongside Steam when performing machine learning tasks.

Navigate to the Clusters page in Steam and click the link for the H2O cluster that you want to open.



This opens H2O Flow in a new tab.



**Note:** Refer to the H2O Flow documentation for information on how to use Flow.

# Starting the Steam CLI

The CLI is an optional utility that can be used to maintain a Steam environment and to create new roles, workgroups, and users. The CLI will primarily be used by admins and/or Steam superusers. The steps below describe how to start the Steam CLI.

Perform the following steps to start the Steam CLI.

1. Open a terminal window and ssh to the machine running Steam. Be sure to provide the correct password for the node when prompted.

```
ssh <user>@<yarn_edge_node>
```

2. Change directories to the Steam folder. From within this folder, log in to the machine running Steam. Use the password that you provided when you created superuser. The example below logs in a user named **Bob**.

```
cd steam-0
```

```
./steam login 192.168.2.182:8080 --username=bob --password=bobSpassword
```

3. Run the following to verify that the CLI is working correctly.

```
./steam help
```

Refer to the [CLI Command Reference Appendix](#) for information on the commands available in the CLI.

## Stopping Steam

When you are finished using Steam, press Ctrl+C in each of the Steam, Compilation Service, and postgres terminal windows to stop the services end your session.

# CLI Command Reference Appendix

- `create identity`
- `create role`
- `create workgroup`
- `deactivate identity`
- `delete cluster`
- `delete engine`
- `delete model`
- `delete role`
- `delete service`
- `delete workgroup`
- `deploy engine`
- `get cluster`
- `get clusters`
- `get engine`
- `get engines`
- `get entities`
- `get history`
- `get identities`
- `get identity`
- `get model`
- `get models`
- `get permissions`
- `get role`
- `get roles`
- `get service`
- `get services`
- `get workgroup`
- `get workgroups`
- `import model`
- `link identity`
- `link role`
- `login`
- `register cluster`
- `reset`

- `start cluster`
  - `stop cluster`
  - `stop service`
  - `unlink identity`
  - `unregister cluster`
  - `update role`
  - `update workgroup`
- 

## **create identity**

### **Description**

Creates a new user.

### **Usage**

```
./steam create identity [username] [password]
```

### **Parameters**

- `[username]` : Enter a unique string for the new user name
- `[password]` : Enter a string for the new user's password

### **Example**

The following example creates a new user with a username of "minsky" and a password of "m1n5kypassword".

```
./steam create identity minsky m1n5kypassword  
Created user minsky ID: 2
```

---

## **create role**

### **Description**

Creates a new role.

### **Usage**

---



```
./steam create role [rolename] --desc="[description]"
```

### Parameters

- `[rolename]` : Enter a unique string for the new role
- `--desc="[description]"` : Optionally enter a string that describes the new role

### Example

The following example creates an engineer role.

```
./steam create role engineer --desc="a default engineer role"  
Created role engineer ID: 2
```

---

## create workgroup

### Description

Creates a new workgroup.

### Usage

```
./steam create workgroup [workgroupname] --desc="[description]"
```

### Parameters

- `[workgroupname]` : Enter a unique string for the new workgroup
- `--desc="[description]"` : Optionally enter a string that describes the new workgroup

### Example

The following example creates a data preparation workgroup.

```
./steam create workgroup preparation --desc="data prep group"  
Created workgroup preparation ID: 1
```

---

## deactivate identity

### Description

---

Deactivates an identity based on the specified username.

### Usage

```
./steam deactivate identity [username]
```

### Parameters

- `[username]` : Specify the username of the identity that you want to deactivate.

### Example

The following example deactivates user "minsky".

```
./steam deactivate minsky
```

---

## delete cluster

### Description

Deletes the specified YARN cluster from the database. Note that this command can only be used with YARN clusters (i.e., those started using `start cluster`.) This command will not work with local clusters. In addition, this command will only work on cluster that have been stopped using `stop cluster`.

### Usage

```
./steam delete cluster [id]
```

### Parameters

- `[id]` : Specify the ID of the cluster that you want to delete.

### Example

The following example deletes cluster 1.

```
./steam get clusters
NAME      ID    ADDRESS          STATE  TYPE      AGE
user      1     localhost:54321   started external  2016-07-01 11:45:58 -0
700 PDT   Cluster deleted: 1
```

## delete engine

### Description

Deletes the specified engine from the database.

### Usage

```
./steam delete engine [id]
```

### Parameters

- `[id]` : Specify the ID of the engine that you want to delete.

### Example

The following example retrieves a list of engines currently added to the database. It then specifies to delete that automodel-hdp2.2.jar engine.

```
./steam get engines
NAME                ID    AGE
automl-hdp2.2.jar    1     2016-07-14 11:48:42 -0700 PDT
h2o-genmodel.jar     2     2016-07-14 11:49:47 -0700 PDT
./steam delete engine 1
Engine deleted: 1
```

---

## delete model

### Description

Deletes a model from the database based on the model's ID.

### Usage

```
./steam delete model [modelId]
```

### Parameters

- `[modelId]` : Specify the ID of the model that you want to delete.

### Example

---

The following example deletes model 3 from the database. Note that you can use `get models` to retrieve a list of models.

```
./steam delete model 3
```

---

## delete role

### Description

Deletes a role from the database based on its ID.

### Usage

```
./steam delete role [roleId]
```

### Parameters

- `[roleId]` : Specify the ID of the role that you want to delete.

### Example

The following example deletes role 3 from the database. Note that you can use `[ get roles ](#get roles)` to retrieve a list of roles. In the case below, this role corresponds to the default data science role.

```
./steam delete role 3
```

---

## delete service

### Description

A service represents a successfully deployed model on the Steam Scoring Service. This command deletes a service from the database based on its ID. Note that you must first stop a service before it can be deleted. (See `stop service .`)

### Usage

```
./steam delete service [serviceId]
```

---

## Parameters

- `[serviceId]` : Specify the ID of the service that you want to delete. Note that you can use `get services` to retrieve a list of services.

## Example

The following example stops and then deletes service 2. This service will no longer be available on the database.

```
./steam stop service 2
./steam delete service 2
```

---

## delete workgroup

### Description

Deletes a workgroup from the database based on its ID.

### Usage

```
./steam delete workgroup [workgroupId]
```

## Parameters

- `[workgroupId]` : Specify the ID of the role that you want to delete.

## Example

The following example deletes workgroup 3 from the database. Note that you can use `get workgroups` to retrieve a list of workgroups.

```
./steam delete workgroup 3
```

---

## deploy engine

### Description

Deploys an H2O engine. After an engine is successfully deployed, it can be specified when starting a cluster. (See `start cluster`.)

## Usage

```
./steam deploy engine [path/to/engine]
```

## Parameters

- `[path/to/engine]` : Specify the location of the engine that you want to deploy.

## Example

The following specifies to deploy the H2O AutoML engine.

```
./steam deploy engine ../engines/automl-hdp2.2.jar
```

---

# get cluster

## Description

Retrieves detailed information for a specific cluster based on its ID.

## Usage

```
./steam get cluster [clusterId]
```

## Parameters

- `[clusterId]` : Specify the ID of the cluster that you want to retrieve

## Example

The following example retrieves information for cluster ID 1.

```
./steam get cluster 1
      user
TYPE:      external
STATE:      healthy
H2O VERSION: 3.8.2.8
MEMORY:     3.56 GB
TOTAL CPUS: 8
ID:         1
AGE:        2016-07-15 09:23:16 -0700 PDT
```

---

## get clusters

### Description

Retrieves a list of clusters.

### Usage

```
./steam get clusters
```

### Parameters

None

### Example

The following example retrieves a list of clusters that are running H2O and are registered in Steam. (See [register cluster](#).)

```
./steam get clusters
NAME          ID    ADDRESS          STATE    TYPE    AGE
user          1    localhost:54321  started  external 2016-07-01 11:45:58 -0700 PDT
```

---

## get engine

### Description

Retrieves information for a specific engine based on its ID.

### Usage

```
./steam get engine [engineId]
```

### Parameters

- `[engineId]` : Specify the ID of the engine that you want to retrieve

### Example

The following example retrieves information about engine 1.

```
./steam get engine 1
      h2o-genmodel.jar
ID:      1
AGE:     2016-07-15 09:44:10 -0700 PDT
```

---

## get engines

### Description

Retrieves a list of deployed engines.

### Usage

```
./steam get engines
```

### Parameters

None

### Example

The following example retrieves a list of engines that have been deployed. (Refer to [deploy engine .](#))

```
./steam get engines
NAME                ID    AGE
h2o-genmodel.jar    1     2016-07-01 13:30:50 -0700 PDT
h2o.jar              2     2016-07-01 13:32:10 -0700 PDT
```

---

## get entities

### Description

Retrieves a list of supported Steam entity types.

### Usage

```
./steam get entities
```

### Parameters

---



None

### Example

The following example retrieves a list of the supported Steam entity types.

```
./steam get entities
NAME      ID
role      1
workgroup 2
identity  3
engine    4
cluster   5
project    6
model      7
service    8
```

---

## get history

### Description

Retrieves recent activity information related to a specific user or for a specific cluster.

### Usage

```
./steam get history [identity [identityName] | cluster [clusterId]]
```

### Parameters

- `identity [identityName]` : Specifies to retrieve activity information related to a specific user
- `cluster [clusterId]` : Specifies to retrieve a activity information related to a specific cluster

### Example

The following example retrieves information for user "bob".

```
./steam get history identity bob
```

USER	ACTION	DESCRIPTION	TIME
1	link	{"id":"2","name":"preparation","type":"workgroup"}	2016-07-15 09:32:55 -0700 PDT
1	link	{"id":"2","name":"engineer","type":"role"}	2016-07-15 09:32:44 -0700 PDT
1	create	{"name":"bob"}	2016-07-15 09:32:32 -0700 PDT

---

## get identities

### Description

Retrieves a list of users.

### Usage

```
./steam get identities
```

### Parameters

None

### Example

The following example retrieves a list of users that are available on the database.

```
./steam get identities
```

NAME	ID	LAST LOGIN	AGE
bob	2	0000-12-31 16:00:00 -0800 PST	2016-07-15 09:32:32 -0700 PDT
jim	3	0000-12-31 16:00:00 -0800 PST	2016-07-15 09:32:38 -0700 PDT
superuser	1	0000-12-31 16:00:00 -0800 PST	2016-07-15 09:21:58 -0700 PDT

---

## get identity

### Description

Retrieve information about a specific user.

### Usage

```
./steam get identity [identityId]
```

### Parameters

- `[identityId]` : Specify the ID of the user you want to retrieve

### Example

The following example retrieves information about user Jim.

```
./steam get identity jim
      jim
STATUS:      Active
LAST LOGIN:  0000-12-31 16:00:00 -0800 PST
ID:          3
AGE:         2016-07-15 09:32:38 -0700 PDT

WORKGROUP    DESCRIPTION
production   production group

ROLE         DESCRIPTION
datascience a default data scientist role

PERMISSIONS
Manage models
View clusters
Manage projects
```

---

## get model

### Description

Retrieves detailed information for a specific model.

### Usage

```
./steam get model [modelId]
```

### Parameters

- `[modelId]` : Specify the ID of the model that you want to retrieve

### Example

The following example retrieves information for model 2.

```
./steam get model 2
```

---

## get models

### Description

Retrieves a list of models.

### Usage

```
./steam get models
```

### Parameters

None

### Example

The following example retrieves a list of models that are available on the database.

```
./steam get models
```

---

## get permissions

### Description

Retrieves a list of permissions available in Steam along with the corresponding code. These permissions are currently hard coded into Steam.

### Usage

```
./steam get permissions
```

### Parameters

None

## Example

The following example retrieves a list of Steam permissions.

```
./steam get permissions
ID      DESCRIPTION      CODE
9       Manage clusters      ManageCluster
7       Manage engines        ManageEngine
5       Manage identities     ManageIdentity
13      Manage models         ManageModel
11      Manage projects       ManageProject
1       Manage roles          ManageRole
15      Manage services       ManageService
3       Manage workgroups     ManageWorkgroup
10      View clusters         ViewCluster
8       View engines          ViewEngine
6       View identities       ViewIdentity
14      View models          ViewModel
12      View projects        ViewProject
2       View roles           ViewRole
16      View services        ViewService
4       View workgroups      ViewWorkgroup
```

---

## get role

### Description

Retrieves detailed information for a specific role based on its name.

### Usage

```
./steam get role [roleName]
```

### Parameters

- `[roleName]` : Specify the name of the role that you want to retrieve

## Example

The following example retrieves information about the datascience role.

```
./steam get role datascience
datascience
DESCRIPTION:    a default data scientist role
ID:             3
AGE:            2016-07-15 09:32:10 -0700 PDT

IDENTITIES: 1
NAME    STATUS    LAST LOGIN
jim      Active    0000-12-31 16:00:00 -0800 PST

PERMISSIONS
Manage models
Manage projects
View clusters
```

---

## get roles

### Description

Retrieves a list of roles.

### Usage

```
./steam get roles
```

### Parameters

None

### Example

The following example retrieves a list of roles that are available on the database.

```
./steam get roles
NAME      ID    DESCRIPTION                AGE
Superuser  1     Superuser                  2016-07-14 09:25:30 -0700 PDT
datascience 3     a default data scientist role 2016-07-14 15:39:03 -0700 PDT
engineer   2     a default engineer role      2016-07-14 15:38:10 -0700 PDT
```

---

## get service

### Description

A service represents a successfully deployed model on the Steam Scoring Service. This command retrieves detailed information about a specific service based on its ID.

### Usage

```
./steam get service [serviceId]
```

### Parameters

- `[serviceId]` : Specify the ID of the service that you want to retrieve

### Example

The following example retrieve information about service 2.

```
./steam get service 2
```

---

## get services

### Description

A service represents a successfully deployed model on the Steam Scoring Service. This command retrieves a list of services available on the database.

### Usage

```
./steam get services
```

### Parameters

None

### Example

The following example retrieves a list of services that are available on the database.

```
./steam get services
```

---

## get workgroup

## Description

Retrieves information for a specific workgroup based on its name.

## Usage

```
./steam get workgroup [workgroupName]
```

## Parameters

- `[workgroupName]` : Specify the name of the workgroup that you want to retrieve

## Example

The following example retrieves information about the production workgroup

```
./steam get workgroup production
                        production
DESCRIPTION:    production group
ID:             3
AGE:            2016-07-15 09:32:27 -0700 PDT

IDENTITIES: 1
NAME    STATUS    LAST LOGIN
jim      Active    0000-12-31 16:00:00 -0800 PST
```

---

## get workgroups

## Description

Retrieves a list of workgroups currently available on the database.

## Usage

```
./steam get workgroups --identity=[identityName]
```

## Parameters

- `--identity=[identityName]` : Optionally specify to view all workgroups associated with a specific user name

## Example



The following example retrieves a list of workgroups that are available on the database.

```
./steam get workgroups
NAME          ID    DESCRIPTION          AGE
preparation    2    data prep group      2016-07-15 09:32:21 -0700 PDT
production     3    production group      2016-07-15 09:32:27 -0700 PDT
```

---

## import model

### Description

Imports a model from H2O based on its ID.

### Usage

```
./steam import model [clusterId] [modelName]
```

### Parameters

- `[clusterId]`: Specify the H2O cluster that contains the model you want to import
- `[modelName]`: Specify the name of the that you want to import into steam.

### Example

The following example specifies to import the GBM\_model\_python\_1468599779202\_1 model from Cluster 1.

```
./steam import model 1 GBM_model_python_1468599779202_1
```

---

## link identity

### Description

Links a user to a specific role or workgroup.

### Usage

```
./steam link identity [identityName] [role [roleId] | workgroup [workgroupId]]
```

## Parameters

- `[identityName]` : Specify the user that will be linked to a role or workgroup.
- `role [roleId]` : Specify the role that the user will be linked to.
- `workgroup [workgroupId]` : Specify the workgroup that the the user will be linked to.

## Example

The following example links user Jim to datascience role and then to the production workgroup.

```
./steam link identity jim role datascience
./steam link identity jim workgroup production
```

---

# link role

## Description

Links a role to a certain set of permissions.

## Usage

```
./steam link role [roleId] [permissionId1 permissionId2 ...]
```

## Parameters

- `[roleId]` : Specify the role that the user will be linked to.
- `[permissionId]` : Specify a single permission or a list of permissions to assign to this role.

## Example

The following example links the datascience role to the ManageProject, ManageModel, and ViewCluster permissions.

```
./steam link role datascience ManageProject ManageModel ViewCluster
```

---

# login

---

## Description

Logs a user in to Steam

## Usage

```
./steam login [address:port] --username=[userName] --password=[password]
```

## Parameters

- `[address:port]` : Specify the address and port of the Steam server.
- `--username=[userName]` : Specify the username.
- `--password=[password]` : Specify the user's password.

## Example

The following example logs user Bob into a Steam instance running on localhost:9000.

```
./steam login localhost:9000 --username=bob --password=bobSpassword  
Login credentials saved for server localhost:9000
```

---

# register cluster

## Description

Registers a cluster that is currently running H2O (typically a local cluster). Once registered, the cluster can be used to perform machine learning tasks through Python, R, and Flow. The cluster will also be visible in the Steam web UI.

Note that clusters that are started using this command can be stopped from within the web UI or using `unregister cluster`. You will receive an error if you attempt to stop registered clusters using the `stop cluster` command.

## Usage

```
./steam register cluster [address]
```

## Parameters

- `[address]` : Specify the IP address and port of the cluster that you want to register.

## Example

The following example registers Steam on localhost:54323. Note that this will only be successful if H2O is already running on this cluster.

```
./steam register cluster localhost:54323  
Successfully connected to cluster 2 at address localhost:54323
```

---

## reset

### Description

Resets the current Steam cluster instance. This removes the current authentication from Steam. You will have to re-authenticate in order to continue to use Steam.

### Usage

```
./steam reset
```

### Parameters

None

### Examples

The following example resets the current Steam instance.

```
./steam reset  
Configuration reset successfully. Use 'steam login <server-address>' to re-authent  
icate to Steam
```

---

## start cluster

### Description

After you have deployed engine, you can use this command to start a new cluster through YARN using a specified engine. Note that this command is only valid when starting Steam on a YARN cluster. To start Steam on a local cluster, use `register cluster` instead.

### Usage

---

```
./steam start cluster [id] [engineId] --size=[numNodes] --memory=[string]
```

## Parameters

- `[id]` : Enter an ID for this new cluster.
- `[engineId]` : Specify the ID of the engine that this cluster will use. If necessary, use `get engines` to retrieve a list of all available engines.
- `--size=[numNodes]` : Specify an integer for the number of nodes in this cluster.
- `--memory=[string]` : Enter a string specifying the amount of memory available to Steam in each node (for example, "1024m", "2g", etc.)

## Example

The following example retrieves a list of engines, then starts a cluster through YARN using one from the list. The cluster is configured with 2 nodes that are 2 gigabytes each.

```
./steam get engines
NAME                ID    AGE
h2o-genmodel.jar    1     2016-07-01 13:30:50 -0700 PDT
h2o.jar             2     2016-07-01 13:32:10 -0700 PDT
./steam start cluster 9 1 --size=2 --memory=2g
```

---

# stop cluster

## Description

Stops a YARN cluster that was started through the CLI or web UI. (See `start cluster` .) Note that you will receive an error if you attempt to stop a cluster that was started using `register cluster` .

## Usage

```
./steam stop cluster [id]
```

## Parameters

- `[id]` : Specify the ID of the cluster that you want to stop. If necessary, use `get clusters` to retrieve a list of clusters.

## Example

The following example stops a cluster that has an ID of 9.

```
./steam stop cluster 9
```

---

## stop service

### Description

A service represents a successfully deployed model on the Steam Scoring Service. Use this command to stop a service.

### Usage

```
./steam stop service [serviceId]
```

### Parameters

- `[serviceId]` : Specify the ID of the scoring service that you want to stop. If necessary, use `get services` to retrieve a list of running services.

### Example

The following example stops a service that has an ID of 2.

```
./steam stop service 2
```

---

## unlink identity

### Description

Removes a user's permissions from a specific role or workgroup.

### Usage

```
./steam unlink identity [identityName] [role [roleId] | workgroup [workgroupId]]
```

### Parameters

- `[identityName]` : Specify the user that will be unlinked from a role or workgroup

- `role [roleId]` : Specify the role that the user will be unlinked from
- `workgroup [workgroupId]` : Specify the workgroup that the the user will be unlinked from

### Example

The following example removes user Jim from the datascience role and then from the production workgroup.

```
./steam unlink identity jim role datascience
./steam unlink identity jim workgroup production
```

---

## unregister cluster

### Description

Stops a cluster that was registered through the CLI or the web UI. (See `register cluster` .) Note that this does not delete the cluster. Also note that you will receive an error if you attempt to unregister a cluster that was started using `start cluster` .

### Usage

```
./steam unregister cluster [id]
```

### Parameters

- `[id]` : Specify the ID of the cluster that you want to stop. If necessary, use `get clusters` to retrieve a list of clusters.

### Example

The following example stops a cluster that has an ID of 9.

```
./steam unregister cluster 2
Successfully unregistered cluster %d 2
```

---

## update role

### Description

---

Edits the description and/or name of an existing role. When a role is edited, the edit will automatically propagate to all identities that are associated with this role.

### Usage

```
./steam update role [rolename] --desc="[description]" --name="[newRoleName]"
```

### Parameters

- `[rolename]` : Enter the role name that you want to edit
- `desc="[description]"` : Optionally enter a string that describes the new role
- `name="[newRoleName]"` : Enter a unique string for the new role name

### Example

The following example changes the name of the engineer role to be "science engineer".

```
./steam update role engineer --desc="A better engineer" --name="science engineer"  
Successfully updated role: engineer
```

---

## create workgroup

### Description

Edits the description and/or name of an existing workgroup. When a workgroup is edited, the edit will automatically propagate to all identities that are associated with this workgroup.

### Usage

```
./steam update workgroup [workgroupname] --desc="[description]" --name="[newWorkgroup  
Name]"
```

### Parameters

- `[workgroup]` : Enter the workgroup name that you want to edit
- `desc="[description]"` : Optionally enter a string that describes the new workgroup
- `name="[newWorkgroupName]"` : Enter a unique string for the new workgroup name

### Example



The following example changes the name of the production workgroup to be "deploy".

```
./steam update workgroup production --desc="A deploy workgroup" --name="deploy"  
Successfully updated workgroup: production
```