

H₂O

Steam

Preview Release

Table of Contents

Introduction	1.1
Steam Standalone Installation and Setup	1.2
Requirements	1.2.1
Optional	1.2.1.1
Starting Steam	1.2.2
Adding Roles, Workgroups, and Users	1.2.2.1
Starting H2O and Running a Model	1.2.3
Start H2O	1.2.3.1
Build a Model	1.2.3.2
Projects	1.2.4
Creating a Steam Project	1.2.4.1
Models	1.2.4.2
Importing Models	1.2.4.2.1
Viewing Model Details	1.2.4.2.2
Comparing Models	1.2.4.2.3
Deploying Models	1.2.4.2.4
Exporting a Model	1.2.4.2.5
Deployment	1.2.5
Uploading a New Package	1.2.5.1
Making Predictions	1.2.5.2
Configurations	1.2.6
Creating a New Label	1.2.6.1
Clusters	1.2.7
Using Steam with H2O Flow	1.2.8
Stopping Steam	1.2.9
What's Next?	1.2.10
CLI Command Reference Appendix	1.3

Introduction

This document describes how to get up and running with Steam without the need for a local running instance of YARN. These instructions will walk through the following procedures:

- Installing and starting Steam, the Compilation Service, and H2O
- Adding Roles, Workgroups, and Users to the database
- Building a simple model in Python (Optional for users who don't have an existing demo.)
- Deploying the model using Steam
- Making predictions using the Steam Prediction Service

During this demo, four terminal windows will remain open for the Steam, Scoring, H2O, and postgres services. A fifth terminal window will be used to run H2O commands in the Python or R example.

Finally, these steps were created using H2O version 3.10.0.3, and that version resides in a Downloads folder. Wherever used, this version number and path should be adjusted to match your version and path.

Steam Standalone Installation and Setup

Requirements

- Web browser with an Internet connection
- JDK 1.7 or greater
- PostgreSQL 9.1 or greater
 - available from [PostgreSQL.org](https://www.postgresql.org)
 - Note that if you get a permissions error when running `brew install postgres`, then you may need to change permissions using either `rm ~/.steam/config` or `brew doctor`.
- Steam tar for Linux or OS X
 - available from www.h2o.ai/steam
- H2O jar file for version 3.10.0.3 or greater
 - available from the [H2O Download](#) page
 - If necessary, follow the instructions on the <http://www.h2o.ai/download/h2o/python> or <http://www.h2o.ai/download/h2o/r> page to upgrade H2O for Python or R.

Optional

The following are required if you use a Python or R demo.

Python

- A dataset that will be used to generate a model. This demo uses the well-known iris.csv dataset with headers (available online), and the dataset is saved onto the desktop.
- Python 2.7

R

- A dataset that will be used to generate a model.
- Comprehensive R Archive Network (R). Available from <https://cran.r-project.org/mirrors.html>.

Starting Steam

This section describes how to set up and start Steam and start the Steam CLI for user management. Five terminal windows will be open the first time you run this setup; four terminal windows will be open for subsequent logins.

1. Go to www.h2o.ai/steam and download the Steam package. Be sure to accept the EULA.
2. Open a terminal window and untar the Steam binary. Note that the command below untars the OS X binary. Replace `darwin` with `linux` in the steps that follow to build on Linux.

```
tar xvf steamY-master-darwin-amd64.tar.gz
```

3. Open a second terminal window and start PostgreSQL. This should be started from the folder where PostgreSQL was installed.

```
postgres -D /usr/local/var/postgres
```

4. Open a third terminal window to create a new user for the Steam database. The commands below only need to be performed once. The example below creates a steam **superuser**. *If prompted, do not enter a password.*

```
createuser -P steam
Enter password for new role:
Enter it again:
```

5. Change directories to the Steam `/var/master/scripts` folder and create the database.

```
cd steam-master-darwin-amd64/var/master/scripts
./create-database.sh
```

6. Change directories to your Steam directory, and start the Jetty server.

```
cd steam-master-darwin-amd64
java -jar var/master/assets/jetty-runner.jar var/master/assets/ROOT.war
```

Note: The Jetty server defaults to port 8080. You can optionally provide a `--port` value for `jetty-runner.jar`.

7. Open a fourth terminal window. From within the **steam-master-darwin-amd64** folder, start the Steam compilation and scoring service. Be sure to include the `--superuser-name=superuser` and `--superuser-password=superuser` flags. (Or provide a more secure password.) This starts Steam on `localhost:9000` and creates a Steam superuser. The Steam superuser is responsible for creating roles, workgroups, and users and maintains the H2O cluster.

```
./steam serve master --superuser-name=superuser --superuser-password=superuser
```

Note: This starts the Steam web service on `localhost:9000`, the compilation service on `localhost:8080` (same as the Jetty server), and the scoring service on `localhost`. You can change these using `--compilation-service-address=<ip_address:port>` and `--scoring-service-address=<ip_address>`. Use `./steam help serve master` or `./steam serve master -h` to view additional options.

8. Open a fifth terminal window to run CLI commands. From within the Steam folder, log in to the machine running Steam (`localhost:9000`). Use the superuser login and password that you created in the previous step.

```
./steam login localhost:9000 --username=superuser --password=superuser
```

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

```
./steam help
```

At this point, you can open a browser and navigate to `localhost:9000`. Note that you may be prompted to once more provide the login credentials supplied in Step 8.

The next section describes how to add additional users to the Steam database.

Adding Roles, Workgroups, and Users

The following example creates sample roles, workgroups, and users using the CLI. Refer to the [CLI Command Reference Appendix](#) for information about all of the commands available in the CLI. These commands are run from the terminal window

used to log in to Steam ([Step 7](#) above).

1. Create an engineer role and link that role to permissions.

```
./steam create role engineer --desc="a default engineer role"  
./steam link role engineer ViewModel ViewProject ViewWorkgroup
```

2. Create a data scientist role and link that role to permissions.

```
./steam create role datascience --desc="a default data scientist role"  
./steam link role datascience ManageProject ManageModel ViewCluster
```

3. Create preparation and production workgroups.

```
./steam create workgroup preparation --desc="data prep group"  
./steam create workgroup production --desc="production group"
```

4. Create two users - Bob and Jim.

```
./steam create identity bob bobSpassword  
./steam create identity jim jimSpassword
```

5. Link Bob to the engineer role; link Jim to the datascience role.

```
./steam link identity bob role engineer  
./steam link identity jim role datascience
```

6. Link Bob to the preparation workgroup; link Jim to the production workgroup.

```
./steam link identity bob workgroup preparation  
./steam link identity jim workgroup production
```

Starting H2O and Building a Model

In order to create a project in Steam, your cluster must already have at least a single dataset. This section describes how to start H2O and runs a small Python demo for adding a dataset and building a model.

Start H2O

1. Open another terminal window. Navigate to the folder with your H2O jar file and start H2O. This will create a one-node cluster on your local machine on port 54321.

```
cd ~/Downloads/h2o-3.10.0.3
java -jar h2o.jar
```

Build a Model

The following steps show how to build model using the Iris dataset and the GBM algorithm. The steps will be run using H2O in Python. Once created, the model can be selected in Steam when creating a new project.

Note: The rest of section can be skipped if you already have demo steps that you use in R, Python, or Flow. If you use another demo, be sure that you initialize H2O on your local cluster so that the data will be available in Steam.

Additional demos for Python are available [here](#).

Demos for R are available [here](#).

A demo of Flow can be viewed [here](#).

1. Open a terminal window. Change directories to the H2O folder, and start Python. Import the modules that will be used for this demo.

```
$ cd ~/Downloads/h2o-3.10.0.3
$ python
>>> import h2o
>>> from h2o.estimators.gbm import H2OGradientBoostingEstimator
```

2. Initialize H2O using localhost and port 54321. (Note that if started Steam on a different machine, then replace `localhost` with the IP address of that machine.)

```
>>> h2o.init(ip="localhost", port=54321)
-----
H2O cluster uptime:      2 minutes 37 seconds 168 milliseconds
H2O cluster version:    3.10.0.3
H2O cluster name:       user
H2O cluster total nodes: 1
H2O cluster total free memory: 3.35 GB
H2O cluster total cores: 8
H2O cluster allowed cores: 8
H2O cluster healthy:    True
H2O Connection ip:      127.0.0.1
H2O Connection port:    54321
H2O Connection proxy:
Python Version:         2.7.9
-----
```

Note: After initializing, verify that the H2O cluster version is 3.10.0.3 or greater. Steam will not run on earlier versions. If necessary, follow the instructions on the <http://www.h2o.ai/download/h2o/python> or <http://www.h2o.ai/download/h2o/r> page to upgrade H2O for Python or R.

3. Upload the Iris dataset. Note that in this example, Python is running from the Downloads folder, and the Iris dataset is on the Desktop:

```
>>> df=h2o.upload_file("../Desktop/iris.csv")
```

4. Specify the configuration options to use when building a GBM model.

```
>>> gbm_regressor = H2OGradientBoostingEstimator(distribution="gaussian", ntree=10, max_depth=3, min_rows=2, learn_rate="0.2")
```

5. Train the model using the Iris dataset (`df` object) and the GBM configuration options.

```
>>> gbm_regressor.train(x=range(1, df.ncol), y=0, training_frame=df)
```

6. Optionally view the model details.

```
>>> gbm_regressor
```

Once created, the model can be added to the Steam UI.

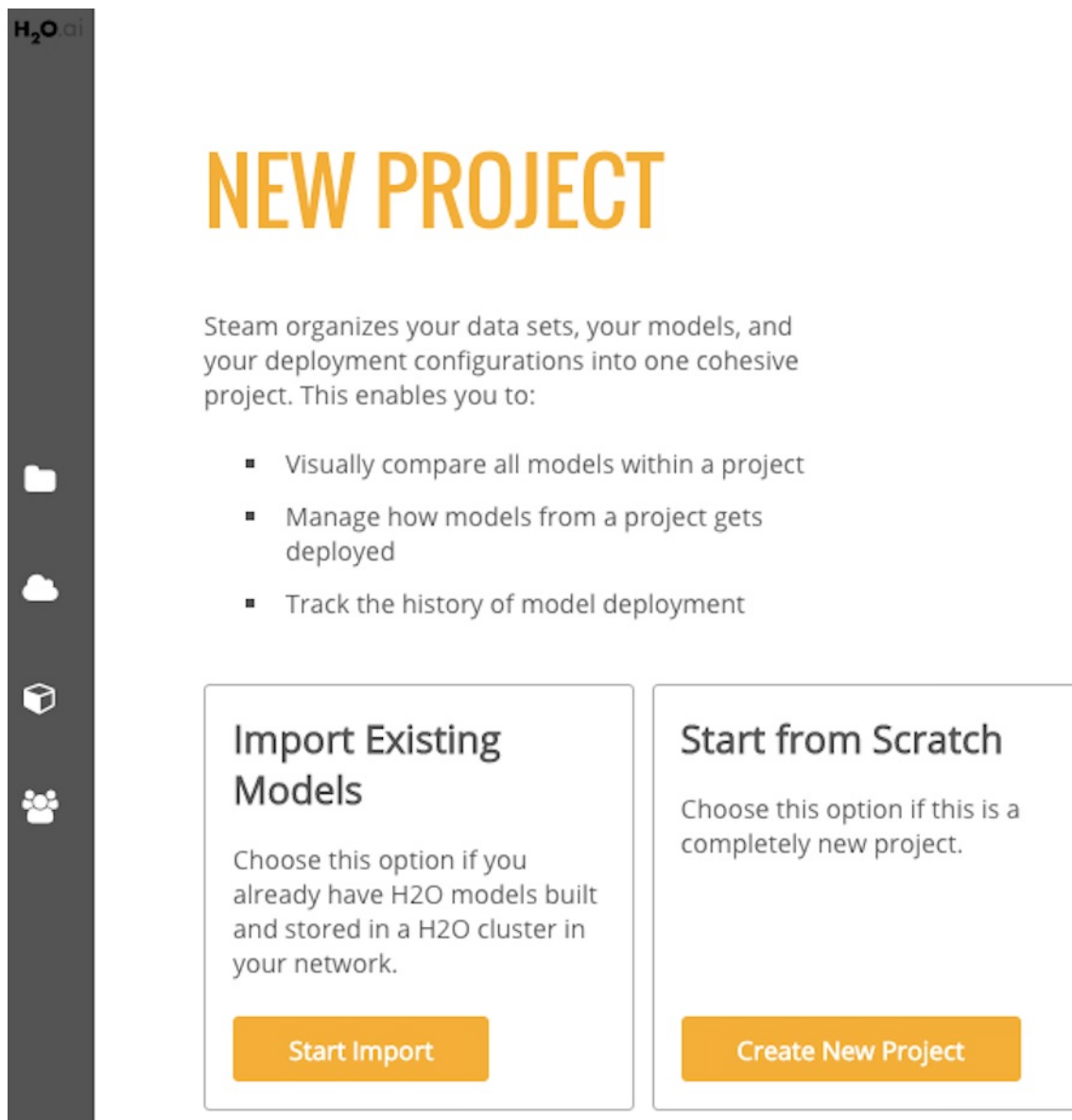
Projects

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. All Steam users within your environment can access these projects and the files within them.

Creating a Steam Project

Now that you have added files to your H2O cluster, you can create your first Steam project. Point your browser to the Steam URL, for example, <http://localhost:9000/>.

1. To start a new project from scratch, click **Create New Project**.



- When you first log in to Steam, the list of clusters will be empty. Enter the IP address of the cluster that is running H2O, then click **Connect**. Once connected, the current list of clusters will immediately populate with this cluster. Connect to this cluster to continue.
- Select an available H2O frame from the Datasets dropdown, then select the Category. Note that these dropdowns are automatically populated with information from datasets that are available on the selected cluster. If no datasets are available, the the dataset list will be empty. For clusters that contain datasets, after a dataset is selected, a list of corresponding models will display.
- 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.
- Specify a name for the project.

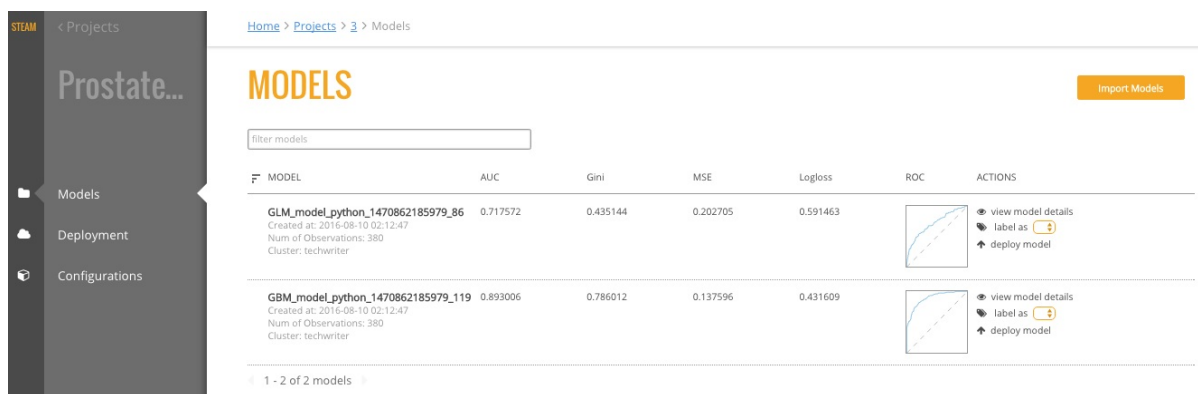
The screenshot shows the Steam Standalone installation setup interface. It is divided into five main steps:

- 1. SELECT H2O CLUSTER**: A table lists available clusters. The first cluster is 'techwriter' with 'N/A' datasets and 'N/A' models. A 'Connect' button is next to it. To the right, there is an option to '... OR CONNECT TO A NEW H2O CLUSTER' with input fields for 'localhost' and '54321', and a 'Connect' button.
- 2. SELECT DATAFRAME**: A dropdown menu shows 'prostate.hex'.
- 3. SELECT MODEL CATEGORY**: A dropdown menu shows 'Binomial'.
- 4. PICK MODELS TO IMPORT**: A table lists models available for import. The table has columns for 'MODEL', 'RESPONSE COLUMN', 'CATEGORICAL', and a checkbox for 'Select for Import'. Two models are listed: 'GBM_model_python_1470854397800_34' and 'GLM_model_python_1470854397800_1', both with 'CAPSULE' as the response column and 'Binomial' as the categorical type. Both have their 'Select for Import' checkboxes checked.
- 5. NAME PROJECT**: A text input field contains 'ProstateCompare'. Below it is a 'Create Project' button.

- 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, and the new project will be available on the **Projects** page.



7. On the **Projects** page, click on the newly created project. This opens a submenu allowing you to view the imported models, deployed models, and configurations specific to that project. Information about these topics is available in the sections that follow.



Models

The **Models** page shows a list of all models included in a selected Project. This list also includes summary information for each model. This information varies based on whether the model is binomial or regression.

For binomial models, the following values will display on the Models page.

- AUC
- Gini
- MSE
- Logloss
- ROC

For regression models, the following values will display on the Models page.

- MRD
- MSE

- R²

Home > Projects > 3 > Models

MODELS Import Models

filter models

MODEL	AUC	Gini	MSE	Logloss	ROC	ACTIONS
GLM_model_python_1470862185979_86 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.717572	0.435144	0.202705	0.591463		<ul style="list-style-type: none"> view model details label as deploy model
GBM_model_python_1470862185979_119 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.893006	0.786012	0.137596	0.431609		<ul style="list-style-type: none"> view model details label as deploy model

1 - 2 of 2 models

You can perform the following actions directly from this page:

- Import a new model
- View model details and export the model as a java, jar, or war file
- Label a model (Refer to [Configurations](#) for information on how to create labels.)
- Deploy the model

Note: The Models page lists models in alphabetical order and shows five models per page. If your project includes more than five models, use the forward and back arrows at the bottom of the page to view more models.

Importing Models

After models are added to an H2O cluster, they can be imported into an existing Steam project. In the upper-right corner of the Models page, click the **Import Models** button. This opens an Import Models popup form.

The Cluster dropdown automatically populates with a list H2O clusters. Specify the H2O cluster that has the models you want to import, then select the additional model or models that you want to add to the project.

IMPORT MODELS

CLUSTER Select the cluster to import models from
techwriter @ localhost:54321

SELECT MODELS

MODEL	DATAFRAME	RESPONSE COLUMN
GBM_model_python_1470862185979_119py_4_sid_9108		CAPSULE
Grid_GLM_py_4_sid_9108_model_python_1470862185979_2275_model_1py_4_sid_9108		CAPSULE
GLM_model_python_1470862185979_256py_4_sid_9108		CAPSULE

☐ ☒ ☐

Click **Import** when you are done. The newly added models will then appear on the Models page.

Viewing Model Details

On the **Models** page, click the **view model details** link under the Action column for the model that you want to view.

STEAM < Projects

Prostate...

Models

Deployment

Configurations

Home > Projects > 3 > Models > 6

GLM_MODEL_PYTHON_1470862185979_86

compared to: [SELECT MODEL FOR COMPARISON](#)

Export Model Deploy Model

Model Overview

Basics

Date

2016-08-10 14:12

Model Type

Generalized Linear Modeling

Model Parameters

Dataset Name

py_4_sid_9108

Response Column Name

CAPSULE

Goodness of Fit

Metrics

Mean Squared Error

0.202705

LogLoss

0.591463

R²

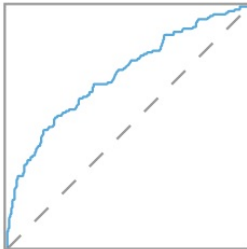
0.157219

AUC

0.717572

Gini

0.435144



This page provides information about when the model was created, the algorithm and dataset used to create the model, and the response column specified when the model was built. The Goodness of Fit section provides value information for the model, including the Mean Squared Error, LogLoss, R^2 , AUC, and Gini score. An ROC curve is available for binomial models.

From this page, you can perform the following actions:

- [Compare two models](#)
- [Deploy the model](#)
- [Export the model](#)

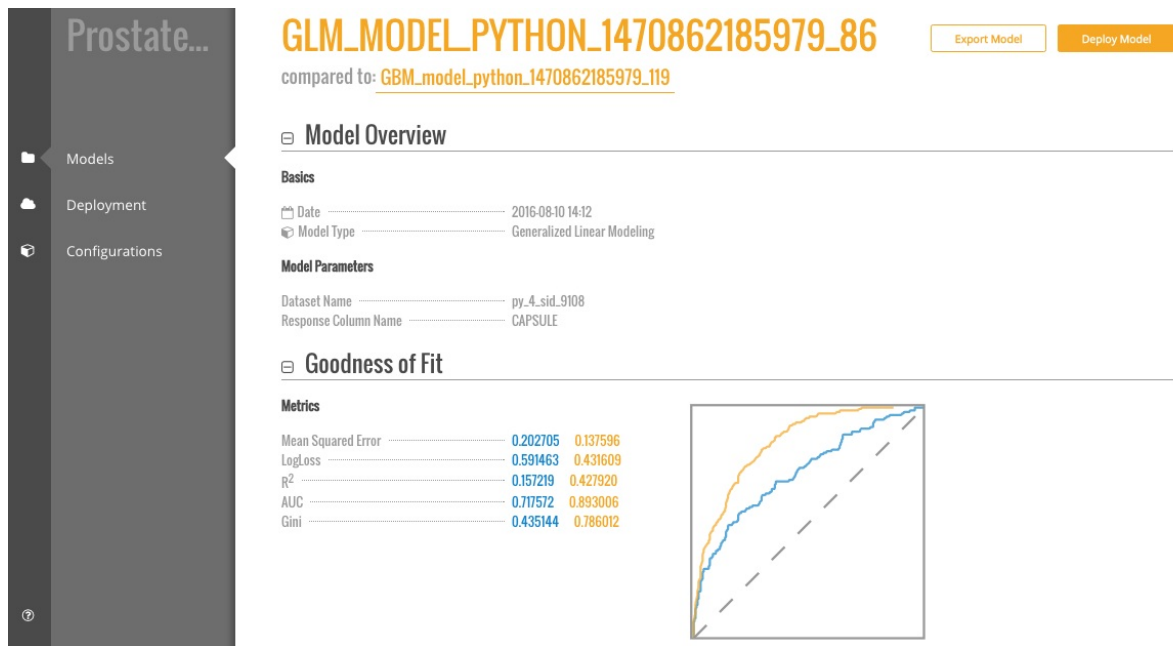
Comparing Models

1. While viewing model details, click the **Compared To** field. This opens a popup showing all models available in the current project.

CHOOSE MODEL TO COMPARE

MODEL	DATE	MSE	AUC
GBM_model_python_1470862185979_1192016-08-10 14:12		0.137596	0.893006
GLM_model_python_1470862185979_862016-08-10 14:12		0.202705	0.717572

2. Select to compare the current model with any available model. This example compares a GLM model with a GBM model. Once a model is selected, the Model Details page immediately populates with the comparison information. The current model values are displayed in blue, and the selected comparison model displays in orange.



Deploying a Model

After comparing models, you might decide to deploy one or more of the best models. Perform the steps below to deploy a model.

1. While viewing the model details, click the **Deploy Model** button. (Note that this can also be done directly from the **Models** page by selecting the **deploy model** link in the Action column.)
2. Specify a service name for the deployment.
3. To perform pre-processing on the model, specify a Preprocessing Script. Note that this dropdown is populated with scripts that are added to the project. Information about adding preprocessing scripts is available in the [Deployment](#) section.
4. Click **Deploy** when you are done.

DEPLOY GLM_MODEL_PYTHON_1470862185979_1

CONFIGURE SERVICE

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

Service name

GLM_Airline_Model

Preprocessing Script

None (Default)

Deploy

Cancel

5. 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. Refer to the [Deployment](#) section for more information.

AirlineCo...

DEPLOYMENT

Upload New Package

DEPLOYED SERVICES PACKAGING

GLM_Airline_Model @ 172.16.2.89:65044		✖ Stop Service
started		
Model	4	
Status	OK	

Exporting a Model

Steam allows you to export models to your local machine.

1. While viewing the model details, click the **Export Model** button.
2. Specify whether to export the model as a .java, .jar, or .war file.
3. To perform pre-processing on the model during the export, specify a Preprocessing Script. Note that this dropdown is populated with scripts that are added to the project. Information about adding preprocessing scripts is available in the [Deployment](#) section.
4. Click **Download** when you are done.

EXPORT GLM_MODEL_PYTHON_1470862185979_1

FILE FORMAT

☒ **.java** a POJO generated by H2O

☐ **.jar** a library file, can be used by java apps

☐ **.war** a java-based web app, can be used by Jetty / Tomcat

ADVANCED OPTIONS

Choose a preprocessing script package

None (Default)

Steam defaults to your browser default Downloads Folders

Download

Cancel

Deployment

The **Deployment** page lists all available deployed services. For each deployed service, this page shows the model name, model ID, and the status. You can stop a running service by clicking the **Stop Service** button

DEPLOYMENT Upload New Package

DEPLOYED SERVICES | PACKAGING

GBM_Airline_Model @ 172.16.2.89:46972 started		✕ Stop Service
Model	3	
Status	OK	

GLM_Airline_Model @ 172.16.2.89:65044 started		✕ Stop Service
Model	4	
Status	OK	

In addition to showing deployed services, a Packaging tab is available showing the preprocessing packages used in the deployment.



Uploading a New Package

Preprocessing packages can be used to perform additional data munging on an existing model.

1. To upload a new preprocessing package, click the **Upload New Package** button in the upper-right corner of the Deployment page.
2. Specify the main Python file that will be used for preprocessing. Click on the folder link to browse for this file.
3. Specify additional files that may be dependencies of the main Python preprocessing file.
4. Enter a name for this new package.
5. Click **Upload** when you are finished.

Upon successful completion, the new preprocessing package will display on the Packages tab of the Deployment page. This file can then be specified when deploying or exporting models. (Refer to [Deploying a Model](#) or [Exporting a Model](#).)

UPLOAD PRE-PROCESSING PACKAGE (PYTHON)

SELECT PYTHON MAIN

Select a main Python file for pre-processing.

The output from this Python file should be one row of an H2O data from that your model is expecting.



train.py



SELECT PYTHON LIBRARIES

Select a one or more Python files for your library.

Any non-standard libraries called here should be installed into your deployment environment prior to launching services.



__init__.py

modelling.py

vectorizer.pickle



NAME THE PACKAGE

Pick a name for this pre-processing package. You will use it as a reference when deploying models.

Package name

MakeWarPython

Upload

Cancel

Making Predictions

1. To reach the Steam Prediction Service, click the IP address link listed under the Deployed Services for the deployed model that you want to score. 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.

Prediction Service

Steam

Select input parameters, OR enter your own custom query string to predict

MODEL INPUT PARAMETERS

Parameters

1. Origin

2. Year

Query String

The parameters above gets automatically built into a REST API query string. You can also input your own string if that's easier for you.

http://172.16.2.89:65044/predict?

PREDICT

CLEAR

PREDICTION RESULTS

Model Runtime Stats

Service started

2016-08-11 17:05:47 UTC

Uptime

1 h 21 m 5 s

MORE STATS

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.

Configurations

Steam allows you to set labels for models (such as Production, Test, etc.) and apply permissions for using the labels. The Steam admin/superuser is responsible for creating new Steam users and setting roles and workgroups for those users. When setting Steam project configurations, labels can be created that allow, for example, only users in a Production workgroup to label a model as a production model.

When a label is applied to a model, the Project Configurations page will show all models associated with a label.

Creating a New Label

1. On the Configurations page, click the **Create New Label** button.
2. Enter a unique name for the label, then provide a description.
3. Click **Save** when you are done.

CREATE / EDIT LABEL

LABEL INFO

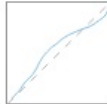
Enter a name and description of your label.
You can use this label in the project for exactly 1 model.

Label name

Label description

Upon successful completion, the new label will display on the Project Configurations page and can be edited or deleted. This label will also be available on the Models page in the **label as** dropdown. The following image shows two labels in the **label as** dropdown: deploy and test.

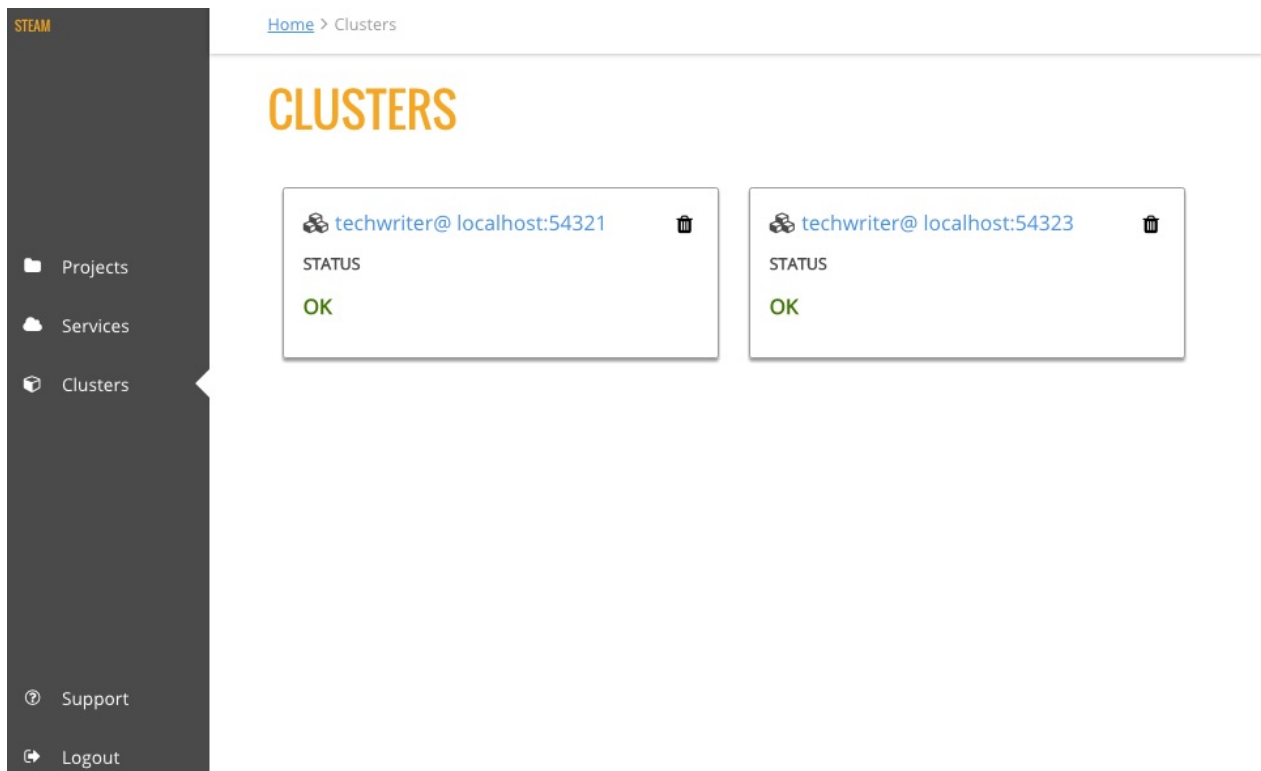
MODELS

MODEL	AUC	Gini	MSE	Logloss	ROC	ACTIONS
GLM_model_python_1470862185979_10.537138 Created at: 2016-08-10 02:12:18 Num of Observations: 30780 Cluster: techwriter	10.537138	0.074275	0.248429	0.689993		view model details label as <input checked="" type="checkbox"/> deploy deploy test
GBM_model_python_1470862185979_30.732534 Created at: 2016-08-10 02:12:18 Num of Observations: 30780 Cluster: techwriter	30.732534	0.465068	0.208744	0.603905		view model details label as <input type="text" value=""/> deploy model

1 - 2 of 2 models

Clusters

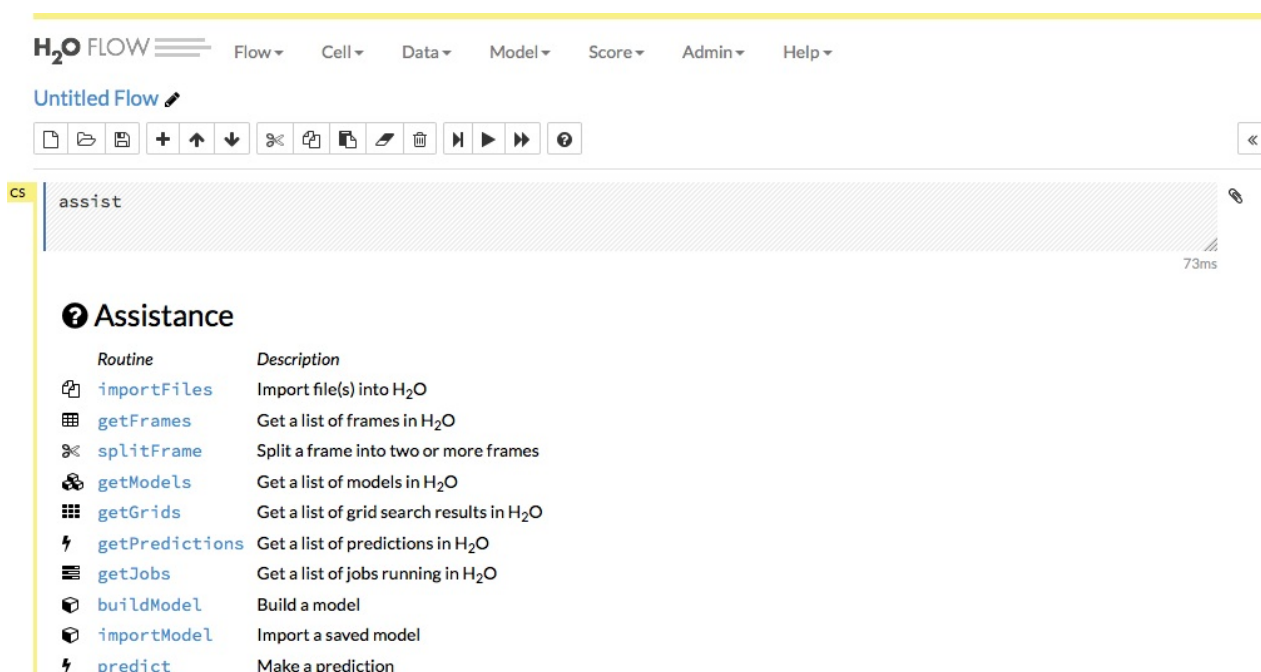
The **Clusters** page shows all H2O clusters that Steam is connected to along with the status of the cluster. From this page, you can click the link to access H2O Flow (see next section), or delete a cluster using the trashcan icon.



Using Steam with H2O Flow

As with other H2O products, Flow can be used alongside Steam when performing machine learning tasks. On the **Clusters** page, 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.

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 and end your session.

What's Next?

Now that you have completed your first demo, you are ready to begin creating models using your own data.

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
```