

Bright Cluster Manager 8.0

Machine Learning Manual

Revision: f4bb9ed

Date: Mon Dec 10 2018



©2017 Bright Computing, Inc. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Bright Computing, Inc.

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Table of Contents	i
0.1 About This Manual	iii
0.2 About The Manuals In General	iii
0.3 Getting Administrator-Level Support	iv
0.4 Getting Professional Services	iv
1 Introduction And Installing the Machine Learning RPMs	1
1.1 Introduction	1
1.2 Packages Available	1
1.3 Requirements	2
1.3.1 Software Installation	2
1.3.2 Module Installation	3
1.3.3 Further Reading	4
2 Running Caffe	5
2.1 Downloading The MNIST Data	5
2.2 Setting The Default Matplotlib Backend	5
2.3 Creating A Working Directory	6
2.4 Using The Web App	6
2.5 Logging In	7
2.6 Creating Training And Validation Datasets	8
2.7 Training A Model	14
2.8 Testing A Model	17
3 Running TensorFlow	21
3.1 Hello World	21
3.2 Training A Convolutional Neural Network	21
3.3 Image Recognition	22
4 Jupyter And JupyterHub Usage	25
4.1 Installation Options	25
4.1.1 Requirements	25
4.1.2 Manual Installation	26
4.1.3 Integrated Installation	26
4.1.4 Integrated Installation Alongside Spark Installation	27
4.1.5 Integrated Installation Using <code>cm-spark-maint</code>	28
4.1.6 Verifying Jupyter And JupyterHub Installation	28
4.2 Creating And Running A Notebook	29
4.3 An Example Of A Notebook Connecting To Spark: Word2Vec	33
4.4 Removal Of JupyterHub	34

Preface

Welcome to the *Machine Learning Manual* for Bright Cluster Manager 8.0.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage basic machine learning capabilities easily using Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 8.0 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at <http://support.brightcomputing.com/manuals>.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Big Data Deployment Manual* describes how to deploy Big Data with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.
- The *Machine Learning Manual*—this manual—describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at manuals@brightcomputing.com.

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website <https://support.brightcomputing.com>. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 13.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Introduction And Installing the Machine Learning RPMs

1.1 Introduction

A number of Machine/Deep Learning library and framework RPM packages have been included in Bright Cluster Manager since version 7.3. Bright makes it faster and easier for organizations install the latest state-of-the-art libraries, and to gain insights from rich, complex data.

1.2 Packages Available

Currently the following RPMs are available:

Table 1.1: Machine Learning Packages Included

Package name	Description
cm-ml-distdeps	Meta-package containing library dependencies for Caffe, NVIDIA DIGITS, Theano, Torch, and TensorFlow. This should be installed on the head nodes and the software images
cm-ml-pythondeps	Pre-packaged Python dependencies for Bright's RPM packages for Caffe, NVIDIA DIGITS, Theano, Torch, and TensorFlow.
caffe	A deep learning framework made with expression, speed, and modularity in mind.
torch7	A Lua(JIT)/C++ library for developing Open Source speech and machine learning applications.
theano	A Python library that allows mathematical expressions involving multi-dimensional arrays to be defined, optimized, and evaluated efficiently.
tensorflow	TensorFlow is an Open Source Software Library for Machine Intelligence

...continues

Table 1.1: Machine Learning Packages Included...continued

Package name	Description
cuda	A GPU-accelerated library of primitives for deep neural networks.
cub	Reusable software components for CUDA
nccl	(pronounced "Nickel") NVIDIA Collective Communication Library — A standalone CUDA library similar in concept to MPI. It has communication routines that have been optimized to achieve high bandwidth over PCIe and other interconnects
mlpython	MLPython is a library for organizing machine learning research.
digits	DIGITS is a web frontend to Caffe and Torch, developed by NVIDIA
cm-jupyter	Jupyter Notebook is a BSD-licensed web notebook
cm-jupyterhub	JupyterHub is a multi-user server for notebooks
cntk	The Cognitive Toolkit by Microsoft Research, is a unified deep-learning toolkit

The following packages are planned for the future:

Table 1.2: Machine Learning Packages Planned

Package name	Description
caffeonspark	CaffeOnSpark brings deep learning to Hadoop and Spark clusters. By combining salient features from deep learning framework Caffe and big-data frameworks Apache Spark and Apache Hadoop, CaffeOnSpark enables distributed deep learning on a cluster of GPU and CPU servers.
bidmach	BIDMach is a very fast machine learning library.

1.3 Requirements

The following requirements must be met before installing the preceding machine learning packages

- The base distribution used must be RHEL, Centos or Scientific Linux 7.x
- There must be access to the Linux distribution's online YUM repositories as well the EPEL repository
- There must be 2 GB of free space for the RPMs that are installed on the head node, and an additional 400 MB for each software image that will be used for running machine learning pipelines

It is recommended, though not required, that the NVIDIA GPUs be Maxwell or more recent, with compute capability 3.5 or later.

1.3.1 Software Installation

Compute Nodes Installation

The `cm-ml-distdeps` RPM package must be installed onto all compute nodes that are to run machine learning applications. The `cm-ml-distdeps` meta-package instructs YUM to install the necessary system libraries as well as the development packages, e.g. `blas-devel`.

For example, if the name of the software image is `gpu-image`, then the administrator can install the RPM as follows:

Example

```
[root@bright80 ~]# yum install --installroot=/cm/images/gpu-image cm-ml-distdeps
```

The preceding command must be applied to all software images that are used to run machine learning applications.

Head Node Installation

The head node must also have the `cm-ml-distdeps` RPM package installed on it. If it is not already installed, then it should be installed:

Example

```
[root@bright80 ~]# yum install cm-ml-distdeps
```

The Bright Cluster Manager machine learning packages have proper RPM dependencies defined. This means that the cluster administrator does not need to spend time figuring out what needs to be installed.

For example, if the administrator wants to install NVIDIA's DIGITS, then all that needs to be done is to run the following command on the head node:

```
[root@bright80 ~]# yum install digits
```

In general, a `yum install <name of desired package>` should do installation for supported machine learning packages. That is, YUM then automatically installs `cm-ml-pythondeps`, `cm-ml-distdeps`, `cuda80-toolkit`, `caffe`, `torch` and `cuda80-toolkit` as dependencies if needed.

At the time of writing (October 2017) there is a minor bug open for CentOS 7.3, which may prevent `cm-ml-pythondeps` installing. A workaround for it is to first install `http-parser` as explained in section 4.1.1.

The RPMs get installed in the `/cm/shared` directory, which is exported over NFS. The RPMs are therefore available to all the compute nodes. The installation of the machine learning libraries is therefore done within minutes, rather than the days that it typically takes to build and install all the necessary dependencies.

Software Libraries Useful For Developers

Developers that work on extending the machine learning libraries typically do not want to use the pre-packaged RPMs. For this use case, Bright can help minimize the time spent to get started.

By installing the `cm-ml-distdeps` on the head node and compute nodes, and the `cm-ml-pythondeps` and `cuda80-toolkit` RPM packages on the head node, a developer can get ready for machine learning development within minutes and spend time on the interesting application at hand, rather than wasting time in dependency hell.

1.3.2 Module Installation

Bright provides environment module definitions for all the machine learning packages. The environment module files are also compatible with the Lmod software introduced in Bright Cluster Manager 7.3.

The machine learning environment modules automatically load additional environment modules as dependencies.

For example, loading the DIGITS module with:

```
module load shared digits
```

automatically loads additional modules such `cuda80-toolkit`, `openblas`, `hdf5_18`, and so on. Those modules are the dependencies needed to use DIGITS.

2

Running Caffe

This chapter goes through an example workflow in order to train a Caffe model to recognize hand-written digits. It closely follows the sample run by NVIDIA's Luke Yeager at <https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md>.

The example uses the MNIST handwritten digit database (<http://yann.lecun.com/exdb/mnist>) as the input to provide a training and validation dataset, and LeNet-5 (<http://yann.lecun.com/exdb/lenet/>) for the neural network model that is to be trained to classify the dataset. Both are generously made available by Yann LeCun from his website at <http://yann.lecun.com/>.

2.1 Downloading The MNIST Data

The MNIST dataset can be downloaded using the DIGITS downloader. For example, a user `tim`, could unpack the datasets into a directory `/mnist` as follows):

```
[tim@bright80 ~]$ python /cm/shared/apps/digits/current/digits/download_data/__main__.py \
mnist /home/tim/
Downloading url=http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz ...
Uncompressing file=train-images-idx3-ubyte.gz ...
Uncompressing file=train-labels-idx1-ubyte.gz ...
Uncompressing file=t10k-images-idx3-ubyte.gz ...
Uncompressing file=t10k-labels-idx1-ubyte.gz ...
Reading labels from /home/tim/train-labels.bin ...
Reading images from /home/tim/train-images.bin ...
Reading labels from /home/tim/test-labels.bin ...
Reading images from /home/tim/test-images.bin ...
Dataset directory is created successfully at '/home/tim/'
Done after 64.6851580143 seconds.
[tim@bright80 ~]$
```

2.2 Setting The Default Matplotlib Backend

The administrator can set the default `matplotlib` GUI backend supported by editing the file:

```
/cm/shared/apps/cm-ml-pythondeps/lib64/python2.7/site-packages/matplotlib/mpl-data/
matplotlibrc
```

and, for example, changing the line:

```
backend: agg
```

to:

```
backend: gtk3agg
```

The preceding change allows GTK3 GUI support, if, for example, just basic X11 GUI support is not wanted, and GTK3 is installed.

2.3 Creating A Working Directory

A working directory can be created with, for example:

```
[tim@bright80 ~]$ mkdir -p /tmp/digits/jobs
```

The actual path, which is set by the parameter `jobs_dir`, can be changed by the administrator in DIGITS version 6.0.0 in:

```
/cm/shared/apps/digits/6.0.0/digits/digits.cfg
```

2.4 Using The Web App

The DIGITS server can be started with:

```
[root@bright80 ~]# module load shared digits
[root@bright80 ~]# cd /cm/shared/apps/digits/6.0.0
[root@bright80 6.0.0]# ./digits-devserver
```

```

  ____  ____  ____  ____  ____  ____
 |  \  \  \  \  \  \  \  \  \  \  \
 |  |  |  |  |  |  |  |  |  |  |  \
 |  /  /  /  /  /  /  /  /  /  /  / 6.0.0

```

```
2017-08-11 08:33:45 [INFO ] Loaded 4 jobs.
```

The server can run with or without a GPU. Running without a GPU is generally not recommended. If run without a GPU the `error #35` response is shown in the output. If there is a GPU on the node, and the `error #35` response is still displayed, then a possible cause is that the NVIDIA driver has not been loaded, or has loaded incorrectly.

Once the DIGITS server is up and running, a web browser can be used to navigate to the home screen of DIGITS. If all is well, then the server location should be indicated by the last line. The locations are typically at:

- `http://localhost/`
- or at
- `http://localhost:5000/` if using `digits-devserver`
- or at
- `http://localhost:34448/` if using `digits-server`

Instead of using `localhost` in the preceding URLs, the external IP address of the system can be used, if the Shorewall firewall is opened for the appropriate port. Typically the administrator adds lines to the `rules` file, similar to the output of the following `grep`:

```
[root@bright80 ~]# grep 5000 /etc/shorewall/rules
# -- Allow port 5000 (digits) traffic from elsewhere to master
ACCEPT:info net fw tcp 5000 #DIGITS
```

After adding the lines, Shorewall should be restarted, with, for example, `service shorewall restart`.

The DIGITS home screen should then show up in the browser (figure 2.1).

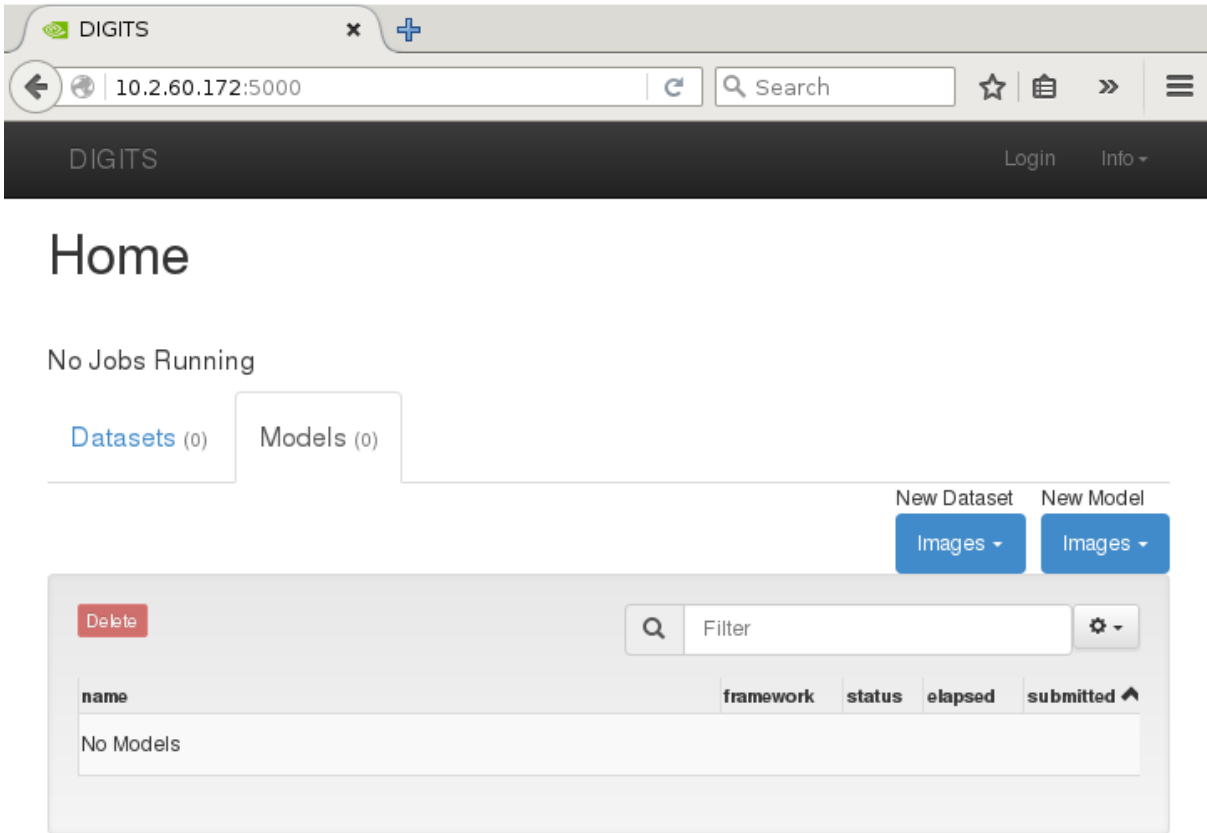
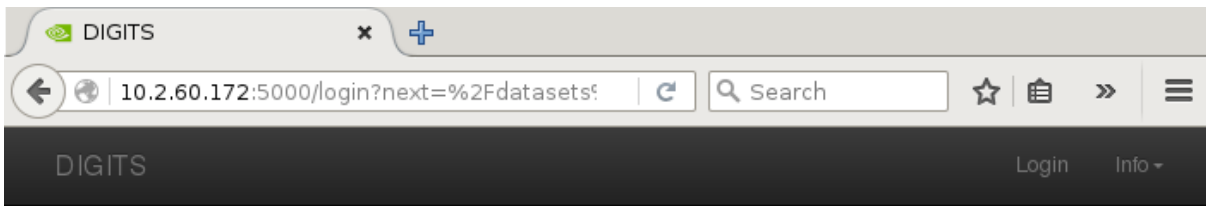


Figure 2.1: DIGITS Home Screen

2.5 Logging In

Clicking on the `Datasets` tab, then on the `Images` menu button under the `New Dataset` label, opens up a menu. Choosing the `Classification` menu item leads to the login page (figure 2.2). The login is carried out without authentication, for convenience. It is not a security feature.



Login

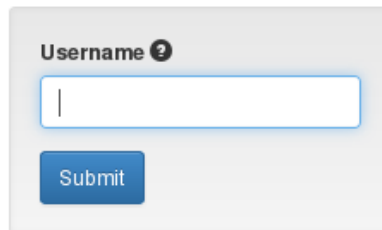
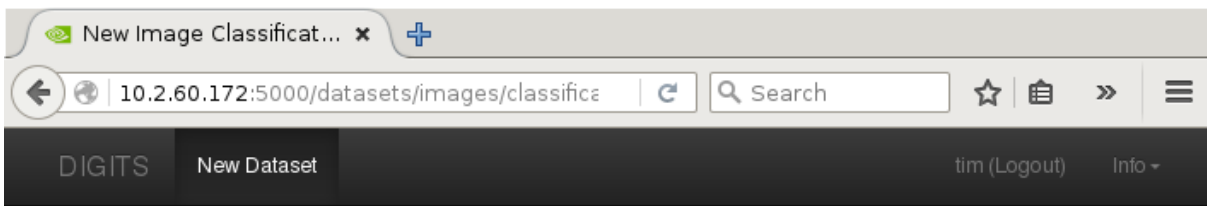
A login form with a 'Username' label, a text input field, and a 'Submit' button.

Figure 2.2: DIGITS Login

2.6 Creating Training And Validation Datasets

After login the New Image Classification Dataset page appears (figure 2.3). The page shows several panes which group the input values for the training dataset that is about to be imported.



New Image Classification Dataset

Image Type ?

Color

Image size (Width x Height) ?

256 x 256

Resize Transformation ?

Squash

[See example](#)

Use Image Folder Use Text Files

Training Images ?

folder or URL

Minimum samples per class ?

2

Maximum samples per class ?

% for validation ?

25

% for testing ?

0

Separate validation images folder

Separate test images folder

DB backend

LMDB

Image Encoding ?

PNG (lossless)

Dataset Name

[Create](#)

Figure 2.3: New Image Classification Dataset Dialog

For the pane starting with the Image Type field:

- The Image Type should be set to Grayscale
- The Image size should be changed to 28 x 28

For the pane starting with the Use Image Folder tab label:

- For `Training Images`, the path to the MNIST training images should be entered. For the download described earlier in section 2.1 the path would be `/home/tim/mnist/train`.
- For the checkbox options:
 - The folder of MNIST test images can optionally also be added as a `Separate validation images folder`.
 - The `Separate test images folder` should not be used—test images are not used for anything in DIGITS yet.

For the pane starting with the `DB backend` field:

- The dataset should be given a name, for example `mnistdataset`

The `Create` button can then be clicked to start the job run to create the classification databases.

A new screen showing the classification job status is then displayed. The screen is scrollable in the browser. The screen is shown in this manual, for printing reasons, as a top part (figure 2.4) and a bottom part (figure 2.5).

A `Job Status` pane at the top right hand side (figure 2.4) shows the expected job completion time if it is refreshed before the job completes, and shows the time the job took after it has completed.

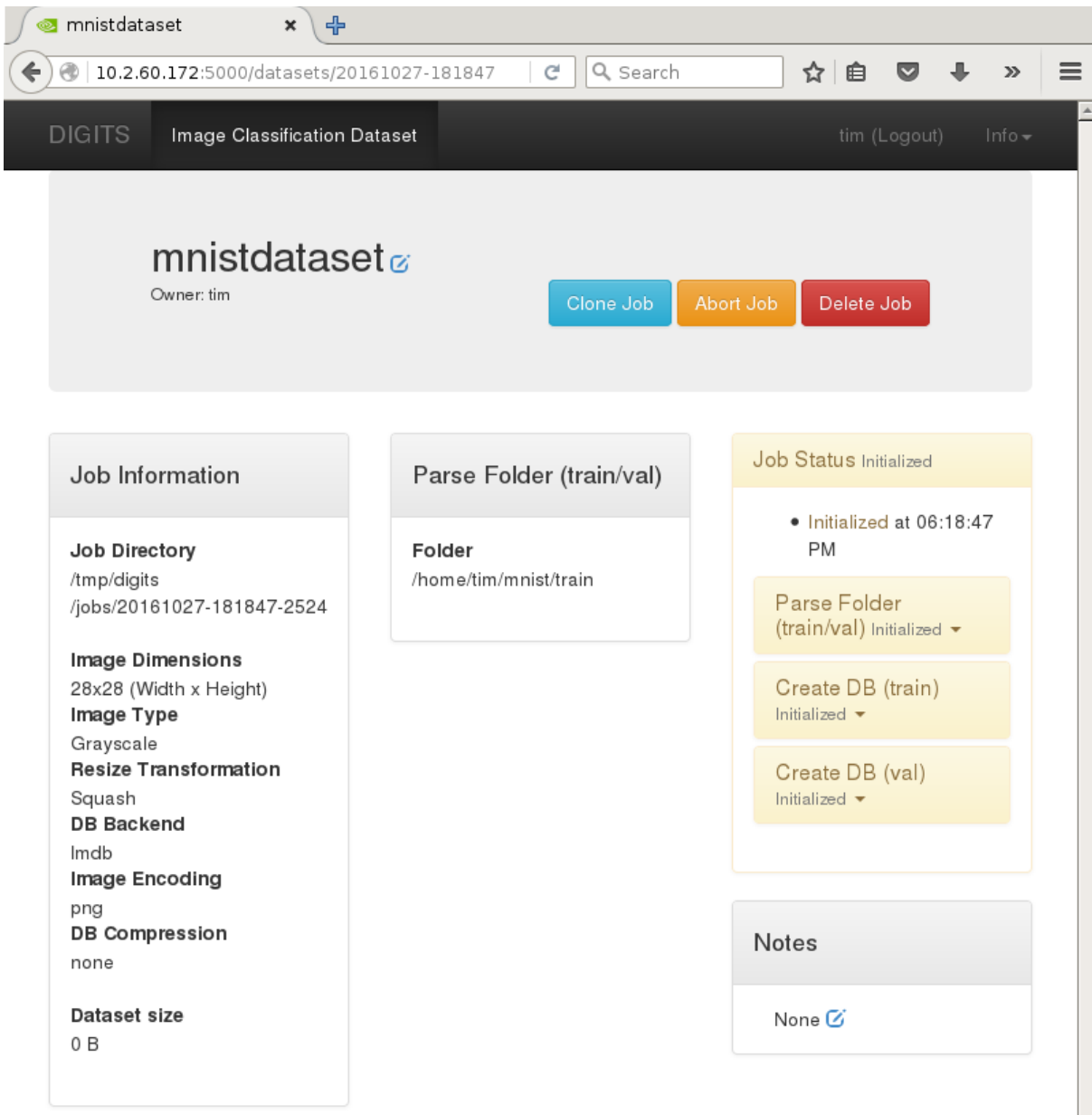


Figure 2.4: DIGITS MNIST Job Run Screen, Top Part

The bottom section (figure 2.5) allows access to the input and log files.

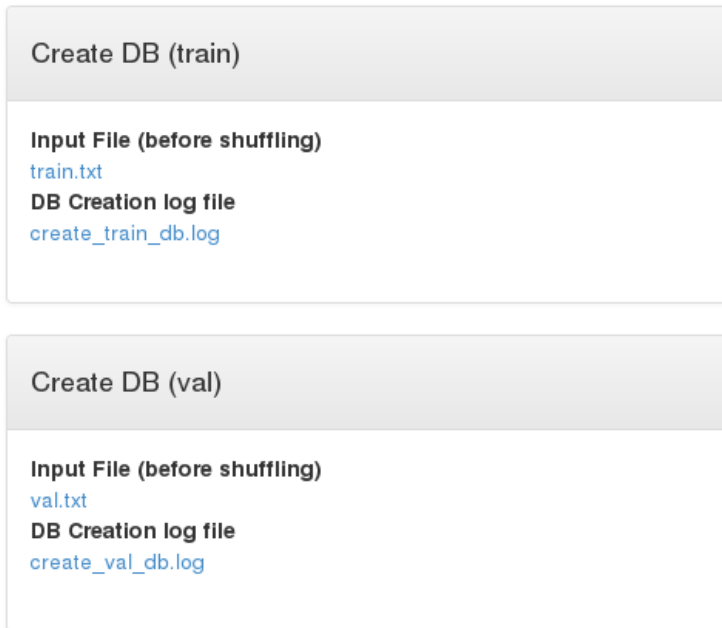


Figure 2.5: DIGITS MNIST Job Run Screen, Bottom Part

After the training and validation database import run is complete, the classification spread can be visualized in the refreshed updated bottom section (figure 2.6).



Figure 2.6: DIGITS MNIST Job Run Screen, Bottom Part, After Run

The corresponding training (`train`) and validation (`val`) database entries can be explored by clicking the `Explore the db` button.

Clicking `DIGITS` in the top left hand part of the web page brings up the home page again. Within the `Datasets` tab (figure 2.7), the name `mnistdataset` that was set earlier is now visible. Clicking on the name brings a screen where like in figure 2.6 the classification spread can be visualized, and the corresponding training (`train`) and validation (`val`) database entries can be explored by clicking the `Explore the db` button.

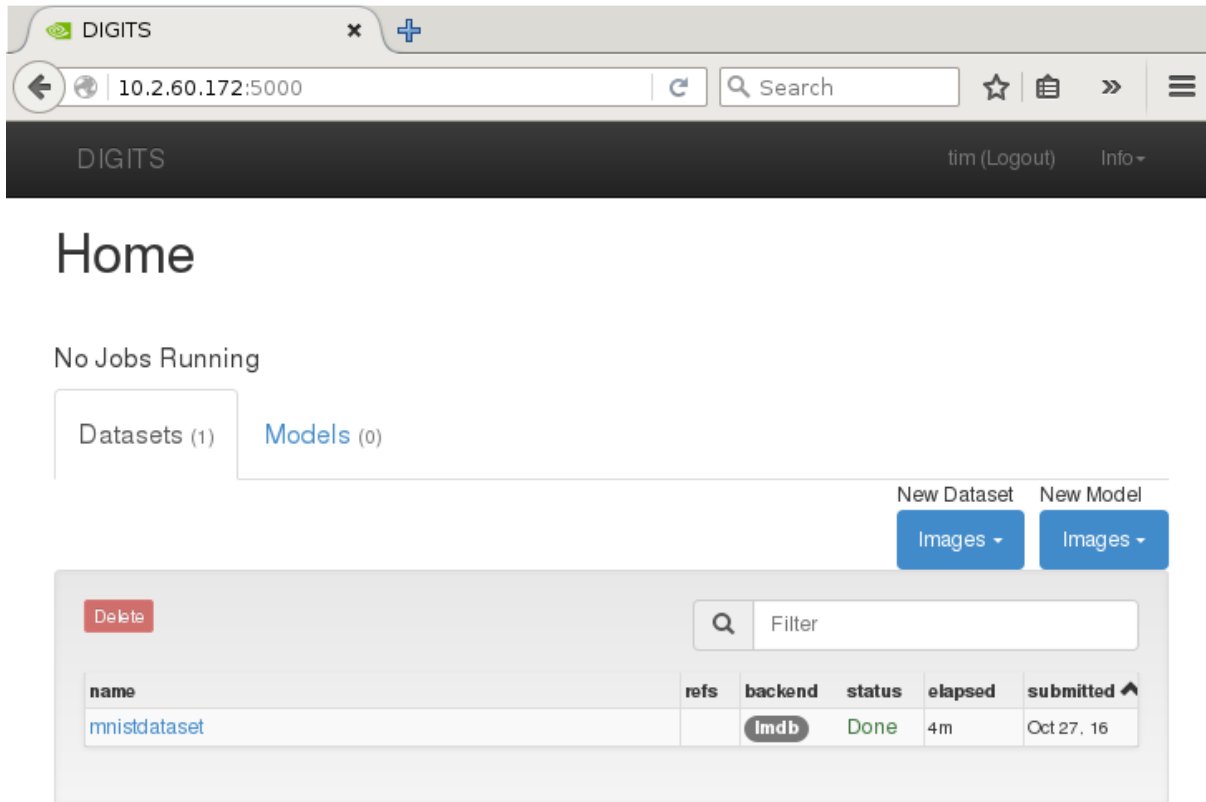


Figure 2.7: Generated Dataset Tab

2.7 Training A Model

The datasets can now be used to train the LeNet model.

Clicking on the `Images` button that is associated with the `New Model` label (figure 2.7), and then on the `Classification` option, opens up the `New Image Classification Model` page. The top part of the page is shown in figure 2.8:

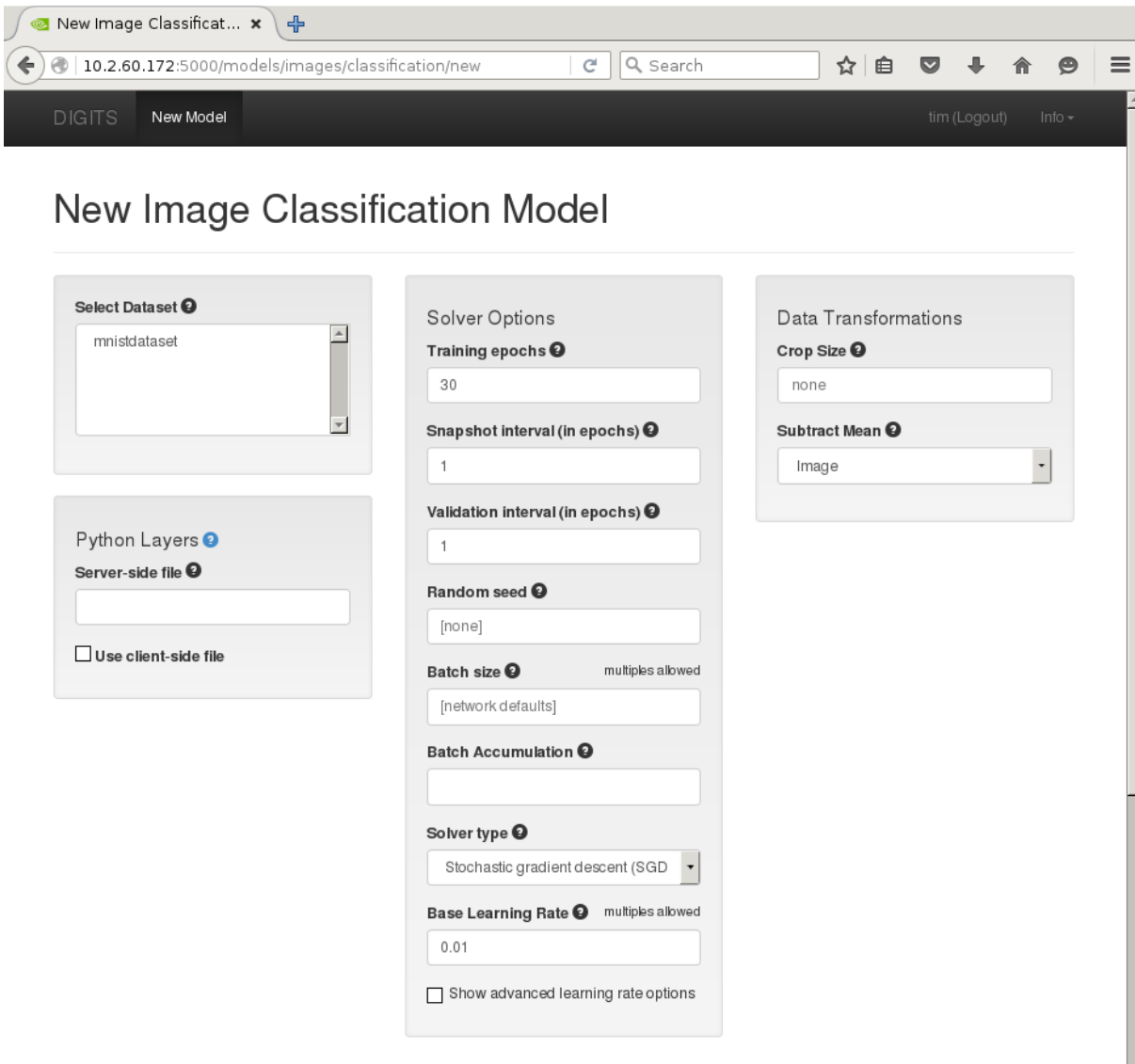


Figure 2.8: New Image Classification Model Top Part

The bottom part of the page, with network and GPU options (figure 2.9) shows network and GPU options.

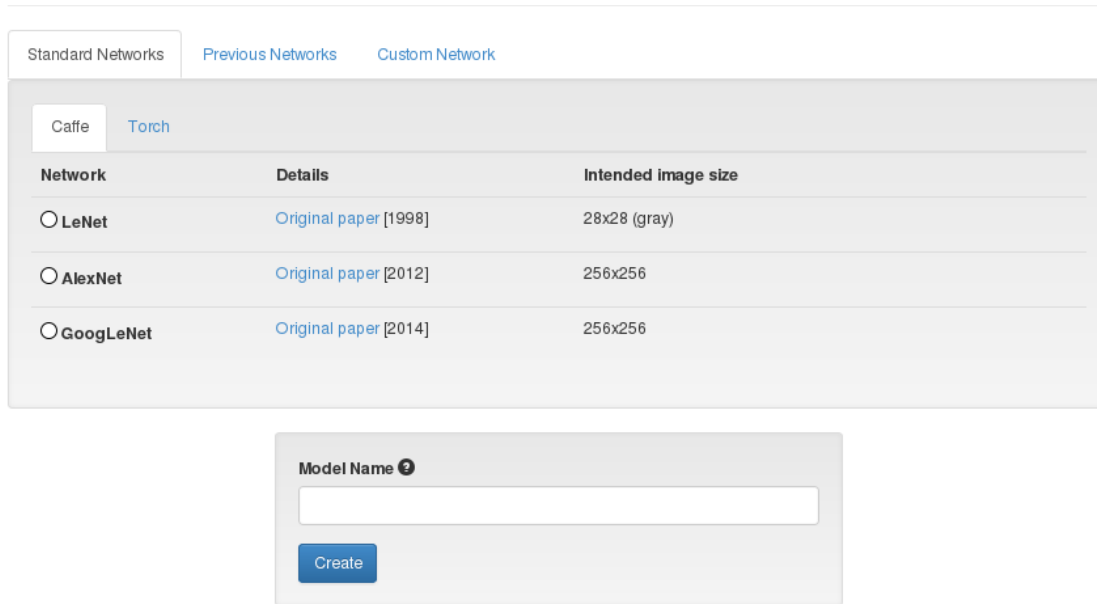


Figure 2.9: New Image Classification Model Dialog Bottom Part

In the New Image Classification Model page the following steps can be followed to classify the data:

- `mnistdataset` is selected in the `Select Dataset` field
- The `LeNet` network can be selected from the `Standard Networks` tab
- The model is given a name, for example `Len`
- The `Create` button is clicked

If there is no GPU on the system, then the `Torch (experimental)` tab should be selected instead of `Caffe`.

While training the model, the expected completion time is seen on the right side (figure 2.10):

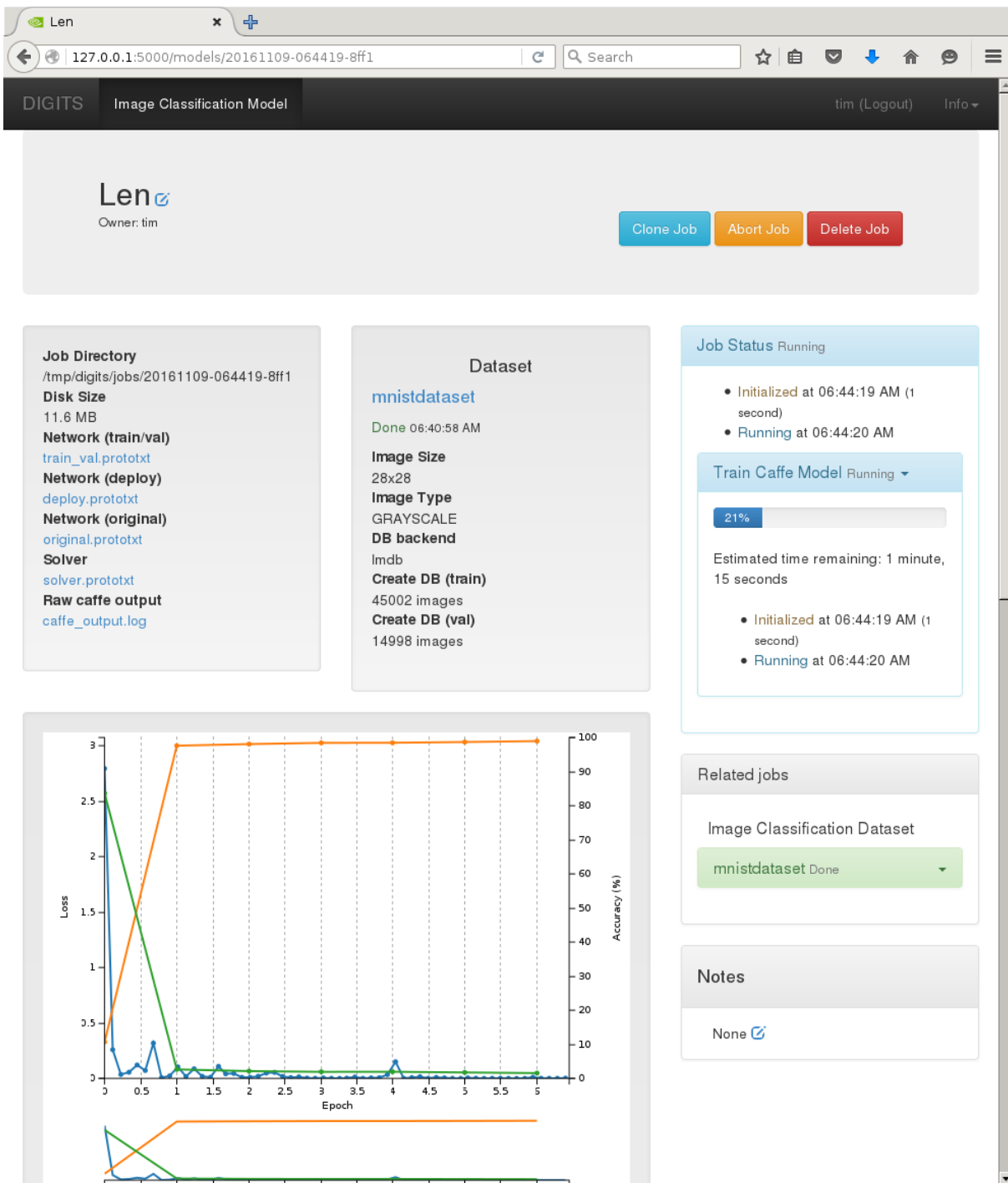


Figure 2.10: New Image Classification Model Training

2.8 Testing A Model

The bottom of the page (figure 2.11) allows the model to be tested.

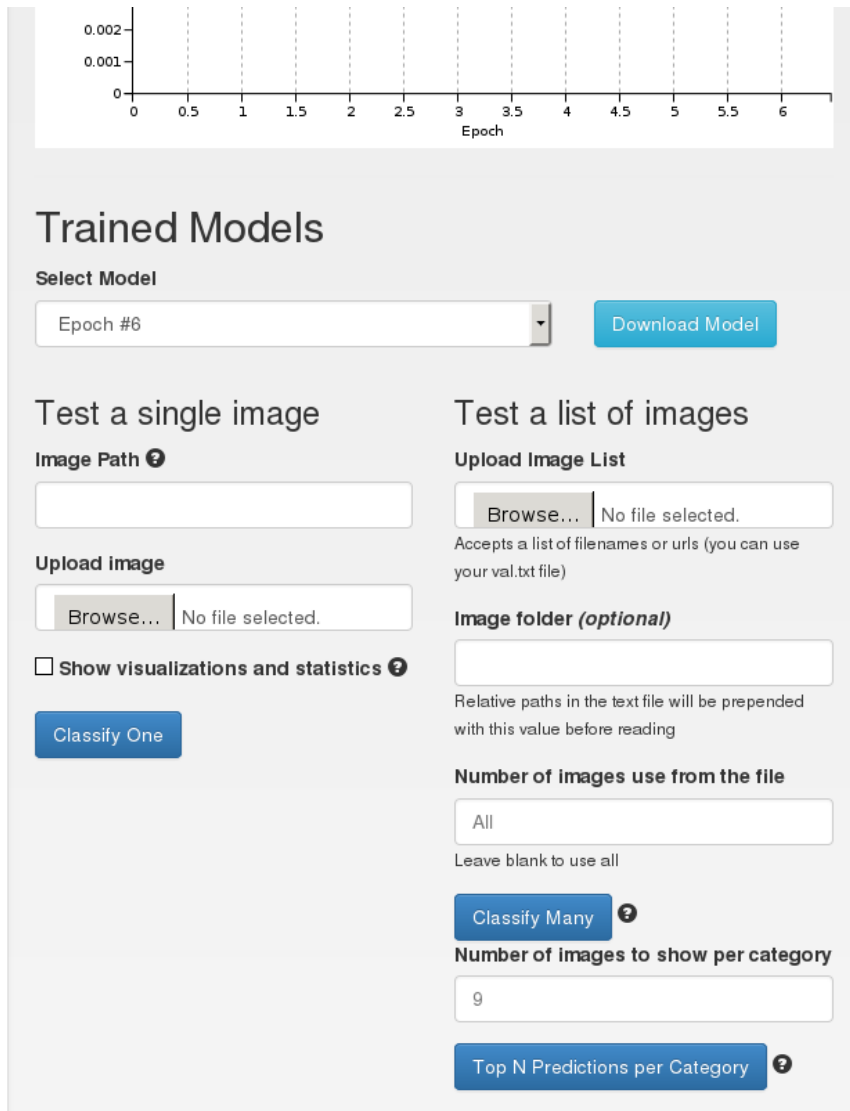


Figure 2.11: New Image Classification Of One Image

- The `Browse` button in the `Upload Image` field, in the `Test a single image` can be clicked upon and a file chosen.
 - There are plenty to choose from under the `/home/username/mnist/test/` directory.
 - Alternatively an image of a digit can be created by hand in an image manipulation software such as Gimp.
 - Alternatively an image from the web can have its URL pasted into the `Image URL` field
- The `Show visualizations and statistics` box can be checked for extra information on how the algorithm was applied.

Clicking on the `Classify One` button then displays a classified image (figure 2.12):

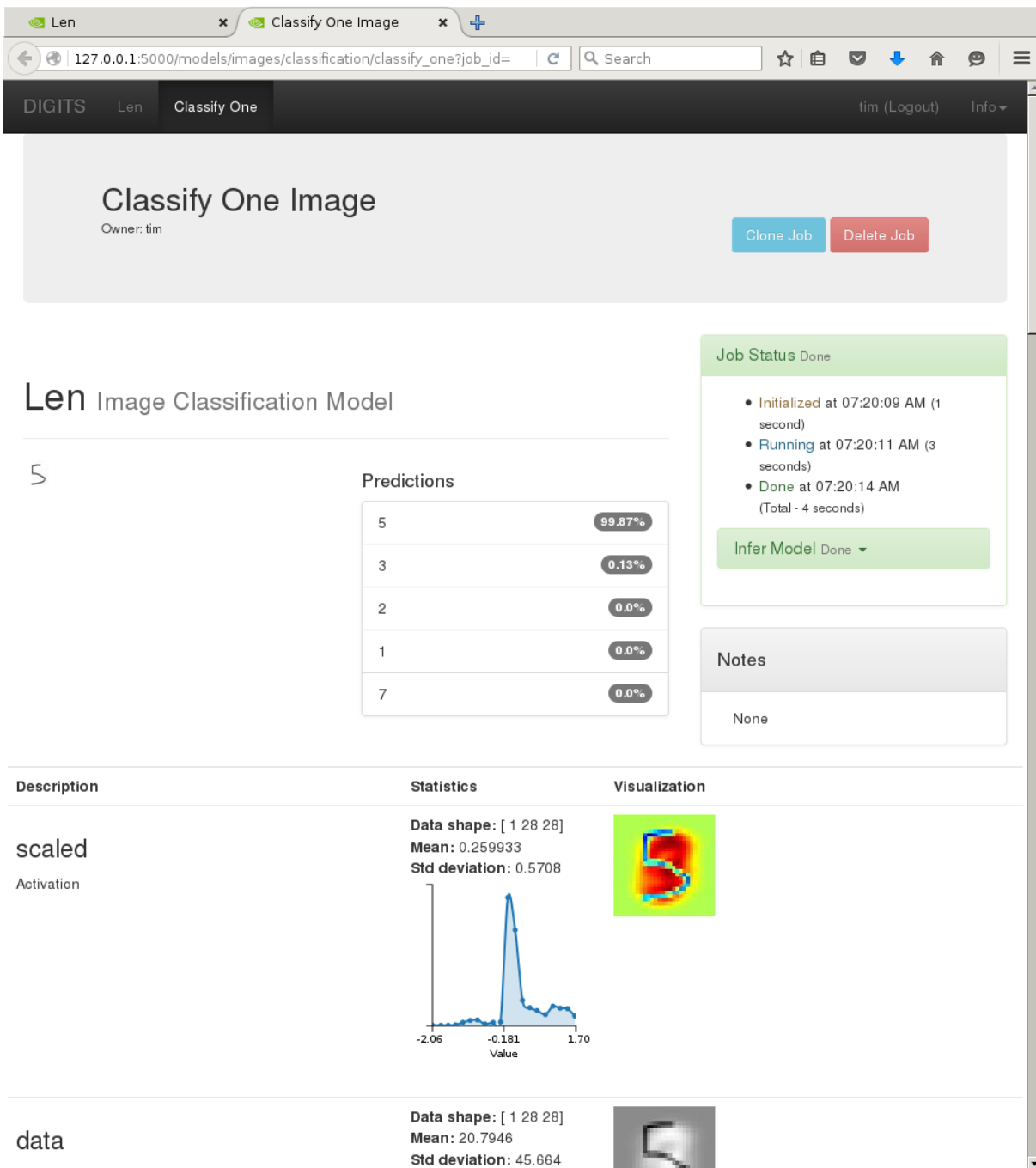


Figure 2.12: New Image Classification Result For One Image

At the top of the page, DIGITS displays the top five classifications and corresponding confidence values. DIGITS also provides further visualizations and statistics about the weights and activations of each layer of the network.

3

Running TensorFlow

This chapter goes through some example runs with TensorFlow. The INFO output messages have been removed in the runs for readability.

3.1 Hello World

A “Hello World” example that just shows that the software is in place for TensorFlow 0.10 can be run as follows:

Example

```
[root@bright80 ~]# python
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf

hello = tf.constant('Hello, TensorFlow!')>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()

name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB

> (device: 0, name: Tesla K40c, pci bus id: 0000:05:00.0)
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

3.2 Training A Convolutional Neural Network

The following trains a convolutional neural network similar to LeNet-5, as explained in <https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html>.

The code uses the TensorFlow convolutional module at `/cm/shared/apps/tensorflow/0.10/lib/python2.7/site-packages/tensorflow/models/image/mnist/convolutional.py`. It picks up training images and labels from the MNIST site, and places them in a directory `data` if it needs to. The images are then used to train and validate the model.

Example

```
[root@bright80 ~]# python -m tensorflow.models.image.mnist.convolutional
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB
Initialized!
Step 0 (epoch 0.00), 36.5 ms
Minibatch loss: 12.054, learning rate: 0.010000
Minibatch error: 90.6%
Validation error: 84.6%
Step 100 (epoch 0.12), 14.2 ms
Minibatch loss: 3.301, learning rate: 0.010000
Minibatch error: 4.7%
Validation error: 7.4%
Step 200 (epoch 0.23), 14.2 ms
Minibatch loss: 3.458, learning rate: 0.010000
Minibatch error: 14.1%
...
Step 8400 (epoch 9.77), 14.1 ms
Minibatch loss: 1.596, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Step 8500 (epoch 9.89), 14.1 ms
Minibatch loss: 1.626, learning rate: 0.006302
Minibatch error: 1.6%
Validation error: 0.9%
Test error: 0.8%
[root@bright80 ~]#
```

3.3 Image Recognition

The following session shows a pre-trained model recognizing a test image of a tiger:

Example

```
[root@bright80 ~]# module load tensorflow/0.10
[root@bright80 ~]# cd /cm/shared/apps/tensorflow/0.10/lib/python2.7/site-packages/tensorflow\
/models/image/imagenet
[root@bright80 imagenet]# python classify_image.py --image_file=Indochinese-Tiger-Zoo.jpg
name: Tesla K40c
major: 3 minor: 5 memoryClockRate (GHz) 0.745
pciBusID 0000:05:00.0
Total memory: 11.92GiB
Free memory: 11.78GiB
W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalization is depre\
```

```
cated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
tiger, Panthera tigris (score = 0.71628)
tiger cat (score = 0.11725)
lynx, catamount (score = 0.00376)
jaguar, panther, Panthera onca, Felis onca (score = 0.00371)
cougar, puma, catamount, mountain lion, painter, panther, Felis concolor (score = 0.00218)
[root@bright80 imagenet]#
```


4

Jupyter And JupyterHub Usage

This chapter covers the usage of Jupyter and JupyterHub, as well as the possibility to integrate it alongside a Spark deployment within Bright Cluster Manager.

Jupyter on its own is single user. However JupyterHub allows it to provide a multi-user service, and is therefore commonly installed with Jupyter. In any case, in Bright Cluster Manager, the package `cm-jupyter` depends upon `cm-jupyterhub`, and so in this chapter, Jupyter and JupyterHub will be referred by only JupyterHub

To be able to use it as multi-user, it is necessary to have JupyterHub. It is possible to install and use Jupyter alone. The package `cm-jupyter` won't trigger the installation of `cm-jupyterhub`. However, this chapter covers the most common case, which is to have them working together. Indeed, the Bright Cluster Manager package `cm-jupyter` is a dependence of `cm-jupyterhub`. Therefore, in this chapter, the combination of Jupyter and JupyterHub are conveniently referred to as JupyterHub.

4.1 Installation Options

There are two ways in which JupyterHub can be installed in Bright Cluster Manager: manual installation, or integrated installation with Spark.

1. Manual installation: This installs only the default Jupyter kernels, normally `python3` and whichever others are found in the package. It can be installed on any node, independently of any Spark deployment.
2. Integrated installation: This is associated with a Spark deployment. The Spark deployment can be standalone or YARN. Besides providing the default kernels, custom kernels are provided for Python 2 and 3, Scala, R (if R is installed on the system), and SQL interpreters. These are for the associated Spark deployment. The advantage of an integrated installation, in practice, is being able to access the Spark context that is automatically set by Toree, by using the variable `sc`.

4.1.1 Requirements

Python 3.3 and higher are recommended, although older versions of Python can be used. Using an older version however results in restricted functionality. Incidentally, Python 2 can be installed alongside Python 3, and users can run kernels normally with any version.

Most administrators should simply install python with:

Example

```
[root@bright80 ~]# (yum install -y python3) || (yum install -y python34)
```

At the time of writing of this section of text (October 2017), there is a minor bug open for CentOS 7.3, at https://bugzilla.redhat.com/show_bug.cgi?id=1481008. The bug affects the installation of the `nodejs` package. JupyterHub requires `nodejs` indirectly. If needed, the installation can be carried out manually with the following workaround:

Example

```
[root@bright80 ~]# rpm -ivh https://kojipkgs.fedoraproject.org//packages/http-parser/2.7.1/3.e17/x86_64/http-parser-2.7.1-3.e17.x86_64.rpm && yum -y install nodejs
```

For an integrated deployment, the SparkR kernel is only available if the \widehat{R} package is installed on the cluster:

Example

```
[root@bright80 ~]# yum install R
```

4.1.2 Manual Installation

A manual installation of JupyterHub consists of the following steps:

- Installation of the `cm-jupyterhub` package to each software image that is to be used
- Re-provisioning of the active nodes that are using these software images, and waiting for the provisioning to finish
- Starting the `cm-jupyterhub` service on each of the re-provisioned nodes, and waiting for the services to come up on those nodes

The `cm-jupyter` and `cm-jupyterhub` service should then be ready for use as a systemd static unit file, and can be added as a service. In Bright Cluster Manager the service can be added within the device mode of `cmsh`, within the `services` submode.

For example, if the administrator would like to run JupyterHub on the head node and on `node001`, and if `node001` has `/cm/images/default-image/` as its path for its software image, then it can be configured as follows:

Example

```
[root@bright80 ~]# yum install -y cm-jupyterhub
...
Complete!
[root@bright80 ~]# yum install -y cm-jupyterhub --installroot=/cm/images/default-image
...
Complete!
[root@bright80 ~]# cmsh -c "device; imageupdate -w -c default"
[root@bright80 ~]# cmsh -c "device use master; services; add cm-jupyterhub; commit"
[root@bright80 ~]# cmsh -c "device use node001; services; add cm-jupyterhub; commit"
```

4.1.3 Integrated Installation

Integrated installation is only supported for Spark versions starting from 2.0.0.

There are two ways to set up an integrated installation of JupyterHub with Spark in Bright Cluster Manager:

1. alongside the Spark installation itself (section 4.1.4)
or
2. carrying it out as a Spark maintenance option (section 4.1.5)

4.1.4 Integrated Installation Alongside Spark Installation

If setting up an integrated installation alongside a Spark installation, then the extra option `--connect-jupyter` is used with `cm-spark-setup` (section 5.1.2 of the *Big Data Deployment Manual*). The option takes a list of nodes as argument. The installation works with deployments of Spark YARN, as well as with deployments of Spark standalone.

So, for example, if there is a Hadoop instance `hdfs1` with YARN already deployed, and the administrator wishes to install Spark and JupyterHub on the head node and have `node001` connected to it, then the command to install Jupyter with Spark can be run as:

Example

```
[root@bright80 ~]# cm-spark-setup -i hdfs1 -t spark-2.2.0-bin-hadoop2.7.tgz --connect-jupyter \
er $(hostname),node001
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk/
Spark release '2.2.0-bin-hadoop2.7'
Found Hadoop instance 'hdfs1', release: 2.7.4
Spark will be installed in YARN (client/cluster) mode.
Spark is already installed in /cm/shared/apps/hadoop/Apache/spark-2.2.0-bin-hadoop2.7/
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Updating images... done.
Waiting for NameNode to be ready... done.
Initializing Spark YARN role... done.
Updating configuration in CMDaemon... done.
Doing Jupyter, JupyterHub and Toree packages installation and integration... done.
Waiting for NameNode to be ready... done.
Validating Spark setup...
-- testing '--master yarn --deploy-mode client' mode...
-- testing '--master yarn --deploy-mode cluster' mode...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

The Spark standalone installation (section 5.3 of the *Big Data Deployment Manual*) has an XML template at `/cm/local/apps/cluster-tools/hadoop/conf/sparkconf.xml`. Instead of using an existing instance as in the preceding example, a Spark standalone can be set up instead using an XML file based on that template. If the instance is called `spark1`, as defined in an XML file `spark1.xml`, and JupyterHub is to be installed on the head node with `node001` connected to it, then the command to install Jupyter with Spark can be run as:

Example

```
[root@bright80 ~]# cm-spark-setup -c "spark1.xml" --connect-jupyter $(hostname),node001
Reading config from file '/tmp/spark1.xml'... done.
Spark release '2.2.0-bin-hadoop2.7'
Creating Spark instance 'spark1'... done.
Spark will be installed in Standalone mode .
Spark Master service will be run on: node001
Spark Worker service will be run on: node002,node003,node004
Spark is already installed in /cm/shared/apps/hadoop/Apache/spark-2.2.0-bin-hadoop2.7/
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Updating images... done.
Initializing Spark Master service... done.
```

```
Initializing Spark Worker services... done.
Updating configuration in CMDaemon... done.
Doing Jupyter, JupyterHub and Toree packages installation and integration... done.
Validating Spark setup...
-- testing Python application...
-- testing Java application...
-- testing Scala application...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

4.1.5 Integrated Installation Using `cm-spark-maint`

If setting up an integrated installation using `cm-spark-maint` (section 5.7 of the *Big Data Deployment Manual*), then the extra option `--connect-jupyter` is used. A list of nodes is used as the input to the option.

The method can be carried out for both Spark YARN and Spark standalone. Its API is identical to the previous method.

So, for example, if there is a Spark deployment of a Hadoop instance `hdfs1`, and the administrator would like to connect JupyterHub on the head node with `node001`, then the command that can be used is:

Example

```
[root@bright80 ~]# cm-spark-maint -i hdfs1 --connect-jupyter $(hostname),node001
Doing Jupyter, JupyterHub and Toree integration... done.
```

Alternatively, if installing to an already-installed Spark standalone called `spark1`, and the administrator would like to connect JupyterHub on the head node with `node001`, then the command that can be used is:

Example

```
[root@bright80 ~]# cm-spark-maint -i spark1 --connect-jupyter $(hostname),node001
Doing Jupyter, JupyterHub and Toree integration... done.
```

4.1.6 Verifying Jupyter And JupyterHub Installation

After the `cm-jupyterhub` service is started, it can take some time until the service is fully up and running. Even if `systemctl status cm-jupyterhub -l` shows that the service is already running, it can still take some seconds longer start functioning.

At this point, whatever the method that was used to install JupyterHub, the `cm-jupyterhub` service should be running on the nodes. Each node should be accessible via a web browser on port 8000 (figure 4.1):

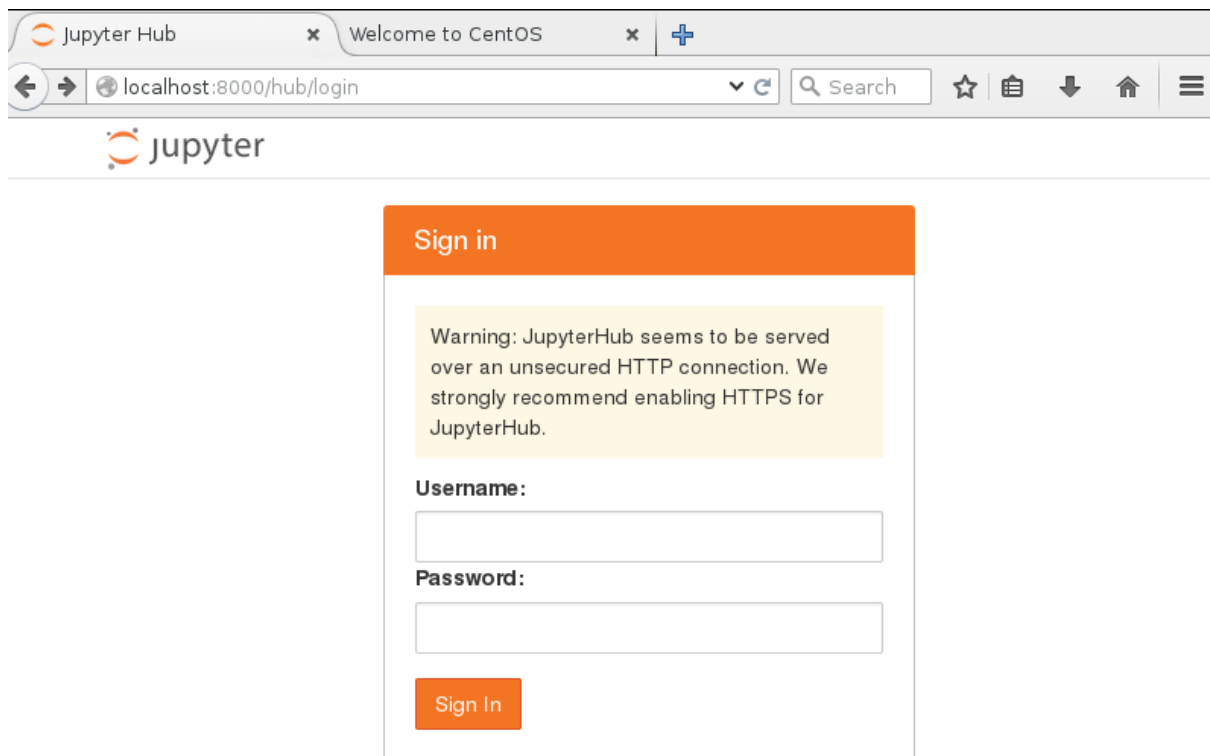


Figure 4.1: JupyterHub login screen

4.2 Creating And Running A Notebook

Any user registered in the Linux-PAM system can log in to JupyterHub. For example, a test user `jupyterhubuser` with password `pw1` can be created with:

Example

```
[root@bright80 ~]# cmssh -c "user ; add jupyteruser ; set password pw1 ; commit "
```

To be able to access the HDFS in the case of integrated deployment with Spark Yarn, this user has to be granted access. Access can be granted with the `cm-hadoop-user` utility, for a given instance `hdfs1`, with:

Example

```
[root@bright80 ~]# cd /cm/local/apps/cluster-tools/hadoop
[root@bright80 hadoop]# cm-hadoop-user --grant jupyteruser=hdfs1
```

A login can then be carried out for the test user in the JupyterHub screen. The user is forwarded to a Jupyter instance (figure 4.2):

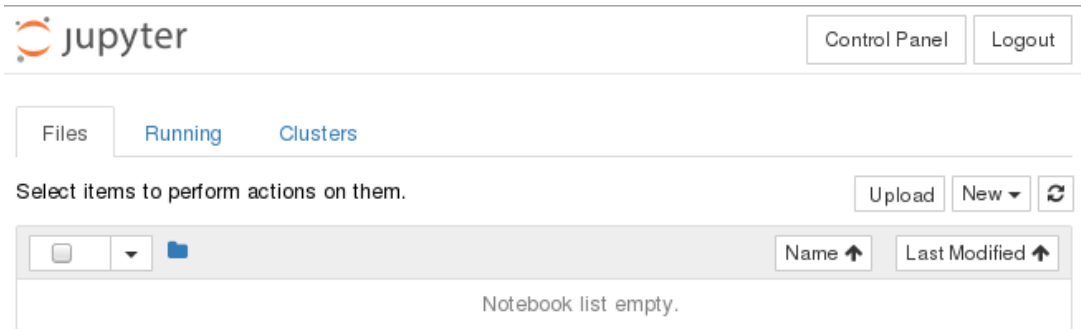


Figure 4.2: JupyterHub landing screen

Clicking on the `New` button of figure 4.2 displays a list of kernels (figure 4.3):

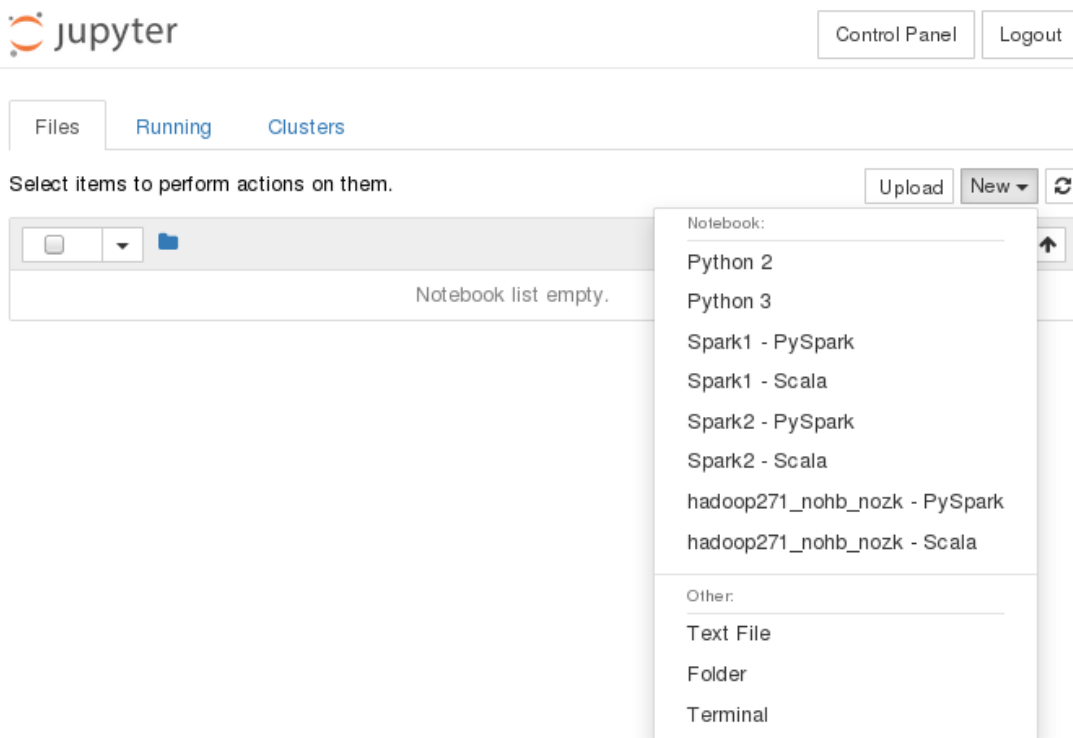


Figure 4.3: JupyterHub kernel list

The two first kernels in figure 4.3, `Python 2` and `Python 3`, are default kernels. At least one of them will be present.

The remaining ones in the example were installed using the integration methods.

If `Python 3` is chosen, then a notebook can be created for it (figure 4.4):

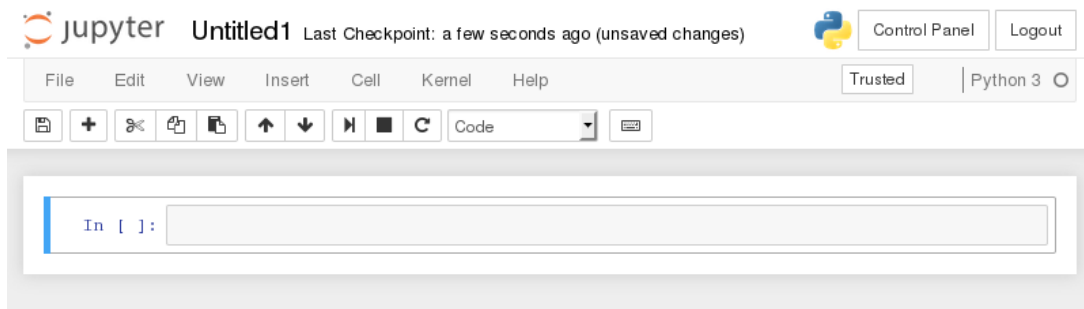


Figure 4.4: Sample Python 3 notebook

A simple Python 3 code, such as `print('Hello')` can be typed in the text entry box and run (figure 4.5):

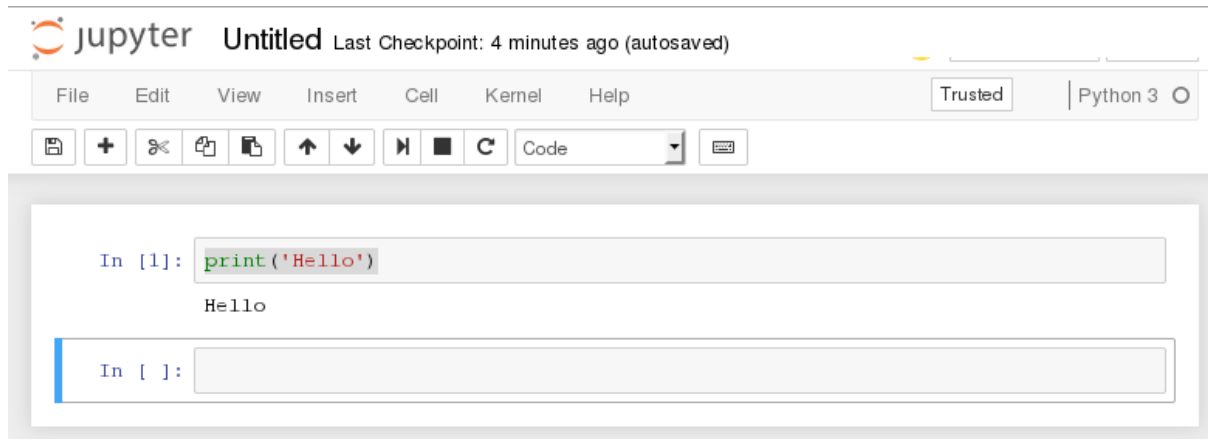


Figure 4.5: Sample Python 3 notebook: Run

It is not possible to import `pyspark` here, because the kernel does not know about the Spark deployment location:

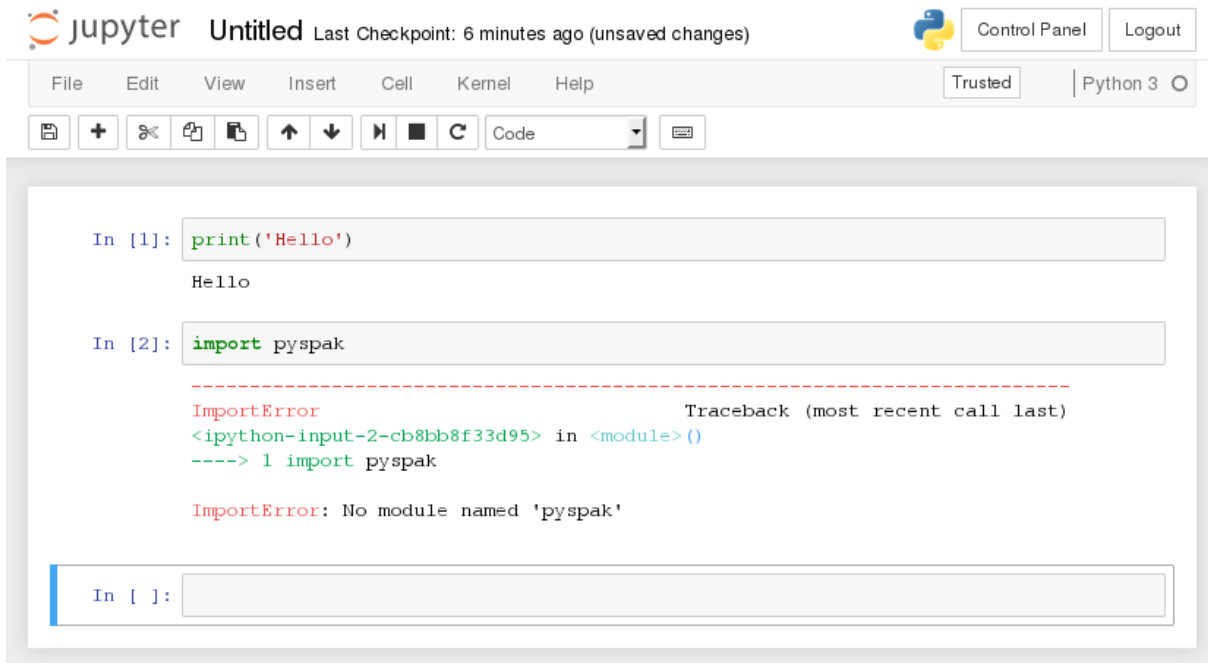


Figure 4.6: Sample Python 3 notebook: PySpark import not possible

The notebook can be closed by clicking on `File` and `Close` and `Halt`.

If the user has any kernel for a Spark deployment, then a notebook can be created for it. In this case, no error is shown after importing `pyspark`, or even accessing the Spark context automatically created by Spark directly (figure 4.7):

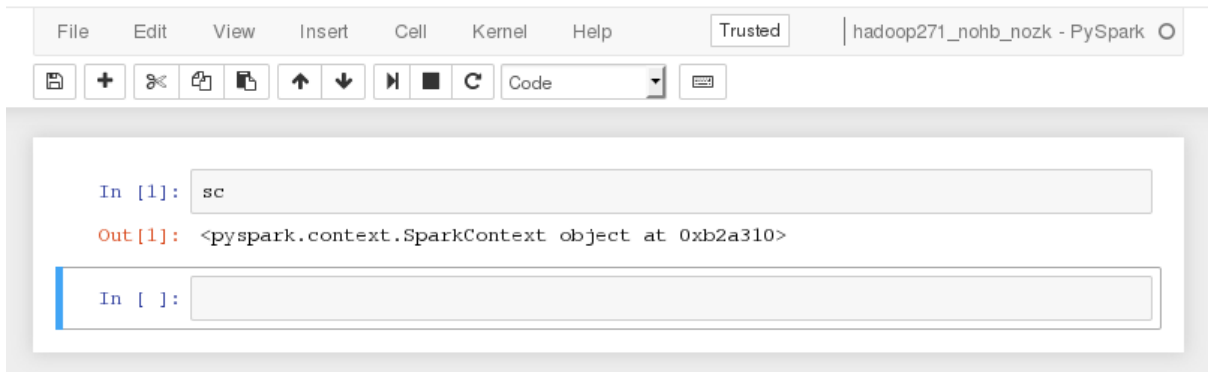


Figure 4.7: Sample integrated Spark notebook: successful import PySpark/access Spark context

That same context is available in all Toree kernels, such as the Scala one:

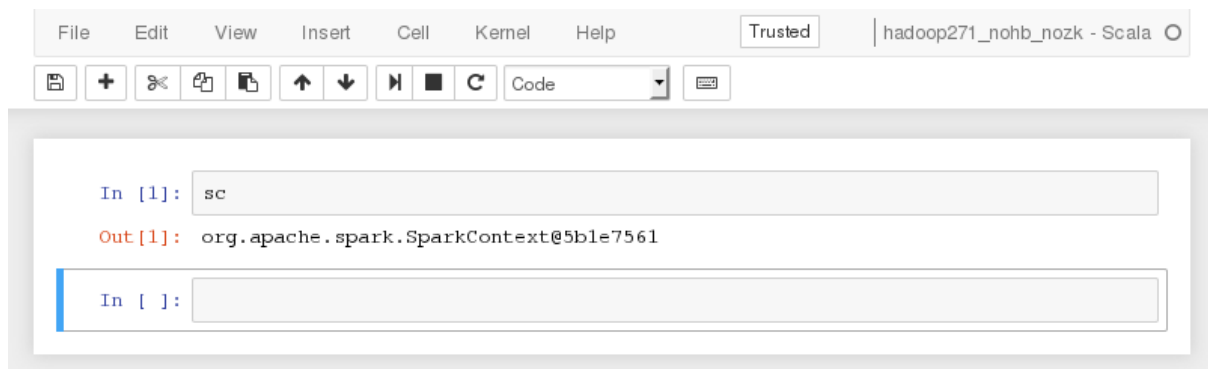


Figure 4.8: Sample integrated Spark notebook: successful access Spark context with Scala

Extra caution is required with the SQL Torea kernel, as of the writing of this chapter. The kernel hang if given invalid SQL queries. This is an open issue at <https://issues.apache.org/jira/browse/TOREE-419> at the time of writing of this section (October 2017).

Also at the time of writing, the R kernel suffers from instabilities. These may prevent the kernel from running. There is an open issue for this at <https://issues.apache.org/jira/browse/TOREE-396>, and for which Bright Computing has proposed a patch.

4.3 An Example Of A Notebook Connecting To Spark: Word2Vec

If JupyterHub deployment has been carried out as in the preceding sections, and it has been verified and its notebook creation capability checked, then it should be ready for use. An example run is now carried out in this section.

The machine learning library code Word2Vec at <https://spark.apache.org/docs/latest/ml-lib-feature-extraction.html#word2vec> takes a dataset of words, and outputs a set of words that can be used in similar contexts. All that is needed is to place the sample data file in a location that both `jupyteruser` and Spark users can access.

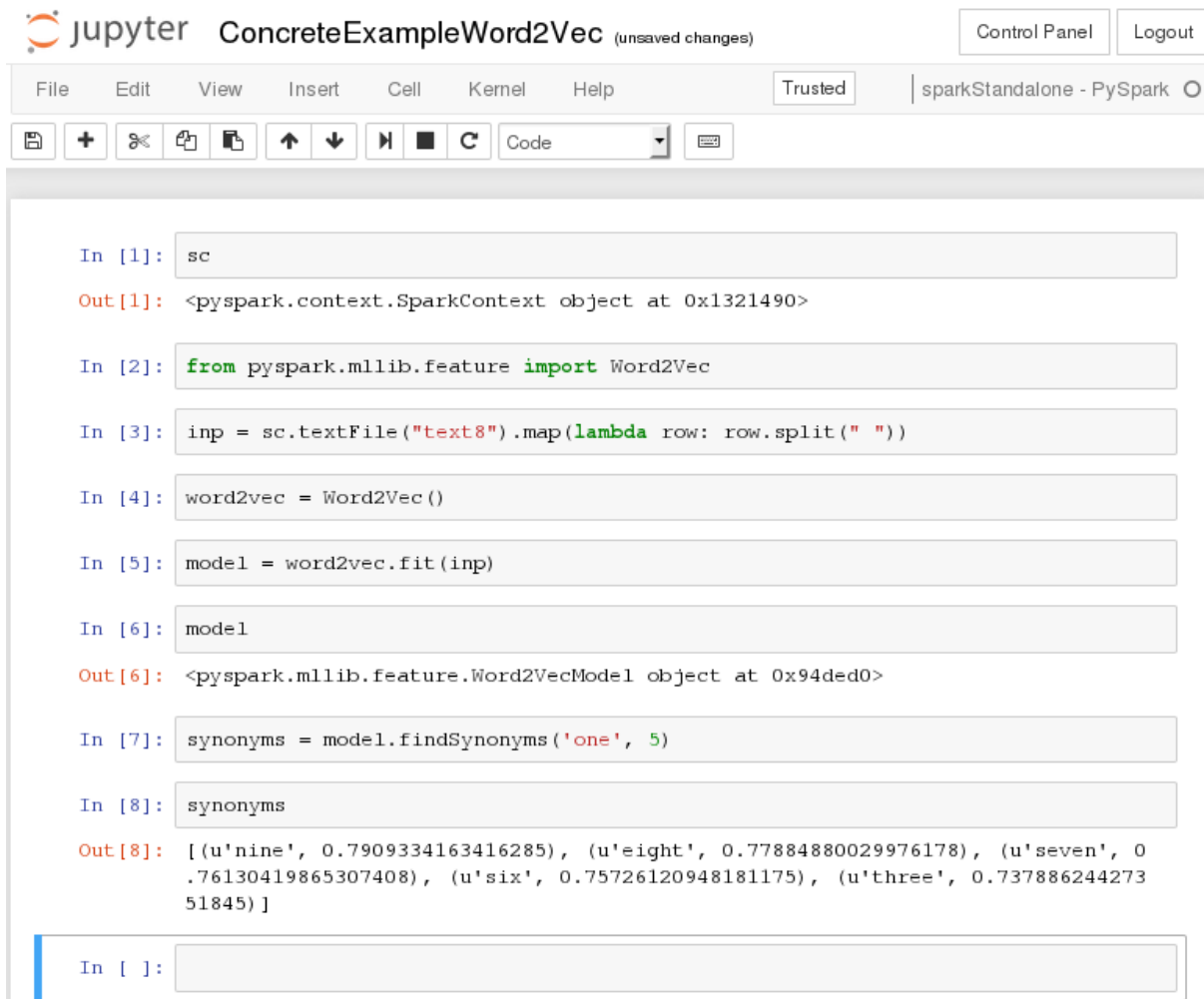
To run the example, a Python programming language, and a Spark standalone deployment, are chose. The kernel PySpark for Spark standalone is an appropriate choice. The preparation can be carried out as follows:

Example

```
[root@bright80 ~]# su - jupyteruser
[root@bright80 ~]# wget http://mattmahoney.net/dc/text8.zip
[root@bright80 ~]# unzip text8.zip
[root@bright80 ~]# truncate -s 10000000 text8 # truncate to 9.6 MB file
[root@bright80 ~]# sudo su - spark ls $PWD/text8 # ensure spark user can access it
[root@bright80 ~]#
```

The `truncate` step ensures the file is small enough to run in a cluster with few resources. Truncation can be skipped for clusters that can run large enough instances.

Instead of PySpark and Spark standalone, Scala and Spark YARN could have been chosen instead. In this case, the file should be uploaded to the HDFS. To use a bare Python 2 or Python 3 kernel, the user must also take care to obtaining the context `sc`, which is set automatically by the integrated kernels.



```

In [1]: sc
Out[1]: <pyspark.context.SparkContext object at 0x1321490>

In [2]: from pyspark.mllib.feature import Word2Vec

In [3]: inp = sc.textFile("text8").map(lambda row: row.split(" "))

In [4]: word2vec = Word2Vec()

In [5]: model = word2vec.fit(inp)

In [6]: model
Out[6]: <pyspark.mllib.feature.Word2VecModel object at 0x94ded0>

In [7]: synonyms = model.findSynonyms('one', 5)

In [8]: synonyms
Out[8]: [(u'nine', 0.7909334163416285), (u'eight', 0.77884880029976178), (u'seven', 0.76130419865307408), (u'six', 0.75726120948181175), (u'three', 0.73788624427351845)]

In [ ]:

```

Figure 4.9: Word2Vec Example

Cell 1 shows that the `sc` context is properly set. The Spark RDD is created in cell 3 using the source file as input.

Cell 2 shows the `Word2Vec` model factory being imported. It takes the Spark RDD to create `Word2VecModel` in cell 5. This is where intensive computation is carried out, and the user may expect significant latency, as indicated by an asterisk `[*]`, depending on the cluster and RDD size. If no errors as cell 2 to 5 are processed, then no output is expected.

Cell 6 shows the output of `model`, and ensures all is well.

In cell 7, the already-computed model is queried to fetch 5 synonyms for the word `one`. Synonyms in the context of `Word2Vec` can differ from the concept as understood by humans.

In cell 8 the synonym output is shown, along with correlation coefficients. The synonyms are `nine`, `eight`, `seven`, `six`, and `three`.

4.4 Removal Of JupyterHub

Before removing JupyterHub, the administrator should ensure that all kernels have been halted and that no user is still logged onto `cm-jupyterhub`. Stopping `cm-jupyterhub` services with users that are still logged in, or with running kernels, has undefined behavior.

To remove the connection between a Spark deployment and a JupyterHub installation, `cm-spark-setup` offers the option `--disconnect-jupyter`. This option takes a list of nodes as argument:

Example

```
[root@bright80 ~]# cm-spark-maint -i hdfs1 --disconnect-jupyter $(hostname),node001
Undoing Jupyter, JupyterHub and Toree packages installation and integration... done.
```

The connection of a Spark deployment with JupyterHub is removed alongside a Spark deployment, without the need of extra parameters:

Example

```
[root@bright80 ~]# cm-spark-setup -u hdfs1
Undoing Jupyter, JupyterHub and Toree packages installation and integration... done.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Cleaning ZooKeeper... done.
Removing additional Spark directories... done.
Removing Spark-related metrics... done.
Removal successfully completed.
Finished.
```

Both of the preceding methods do not remove the packages from the nodes. This is to avoid clashes with manual installations that are not managed by Bright Cluster Manager utilities.

The packages can still be manually uninstalled by reversing the steps of a manual installation by:

- Stopping the `cm-jupyterhub` service on each one of the nodes.
- Removing the package `cm-jupyterhub` for each desired software image.
- Reprovisioning the active nodes that have those software images, and waiting for the reprovisioning to finish.

After reprovisioning, `cm-jupyterhub` is no longer installed.

Thus, for example, if the administrator would like to remove JupyterHub on the headnode and `node001`, and `node001` has `/cm/images/default-image/` as the path for its software image, then the removal can be carried out with:

Example

```
[root@bright80 ~]# ssh node001 service cm-jupyterhub stop
[root@bright80 ~]# service cm-jupyterhub stop
[root@bright80 ~]# yum remove -y cm-jupyterhub --installroot=/cm/images/default-image
[root@bright80 ~]# yum remove -y cm-jupyterhub
[root@bright80 ~]# cmsh -c "use device ; imageupdate -w -c default"
```

Removing `cm-jupyterhub` with `yum` does not remove the dependent packages that were installed with it. The resolution of the dependencies may spawn a whole tree of package changes, and is outside the scope of this chapter. However, the direct dependent packages for the package `cm-jupyterhub` in Bright Cluster Manager are as follows:

- `cm-jupyterhub` depends on:
 - `cm-jupyter`
 - `cm-npm-configurable-http-proxy` depends on:
 - * `npm`

Finally, the testing user created earlier to verify the installation of JupyterHub can be removed with:

Example

```
cmsh -c "user ; remove jupyteruser ; commit "  
Successfully removed 1 Users  
Successfully committed 0 Users
```