Creating a Linux virtual machine cluster and running simulation analysis with IPython Cluster

Overview

In this hands-on lab you will learn to deploy an IPython notebook cluster on Windows Azure. You will also execute the Monte Carlo simulation example with IPython Notebook on the cluster you created on Windows Azure.

Objectives

In this hands-on lab, you will learn how to:

- Provision a Linux virtual machine cluster with Python.
- Deploy IPython notebook on your virtual machines.
- Run Monte Carlo Simulation on Ipython Notebook on Windows Azure.

Prerequisites

The following is required to complete this hands-on lab:

- A Windows Azure subscription - sign up for a free trial
- Lab: Using Windows Azure Virtual Machine

Exercises

This hands-on lab includes the following exercises:

1. Build an Environment to manage Windows Azure with Python
2. Deploy IPython notebook on Windows Azure
3. Run Monte Carlo Simulation on IPython

Estimated time to complete this lab: **60** minutes.

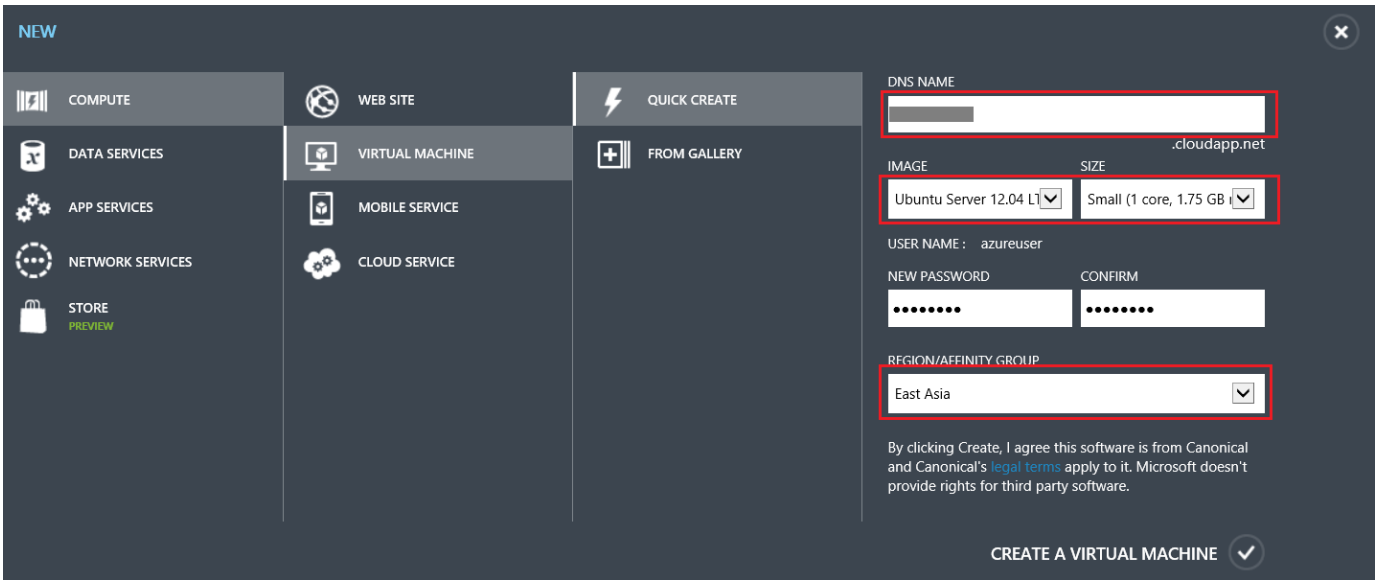
Excercise 1: Build an Environment to manage Windows Azure with Python

First, you will need to deploy required software on your linux machine. You are going to install git, Python 2.7, workerpool and paramiko, then you will connect Windows Azure by Python with some configuration. For the lab, we will either provide you with a shared virtual machine to do the exercise or let you provision a Linux virtual machine on Windows Azure to avoid issues with your laptop, etc. Please remember this can be any Linux /Unix machine including your laptop. It is simply an environment you will use to deploy a cluster on Windows Azure.

**In the case we provide you with a virtual machine to login, you will skip steps 5, 6, 7, 8 below and start with Azure subscription or Task 2 - Setup Windows Azure Subscription **

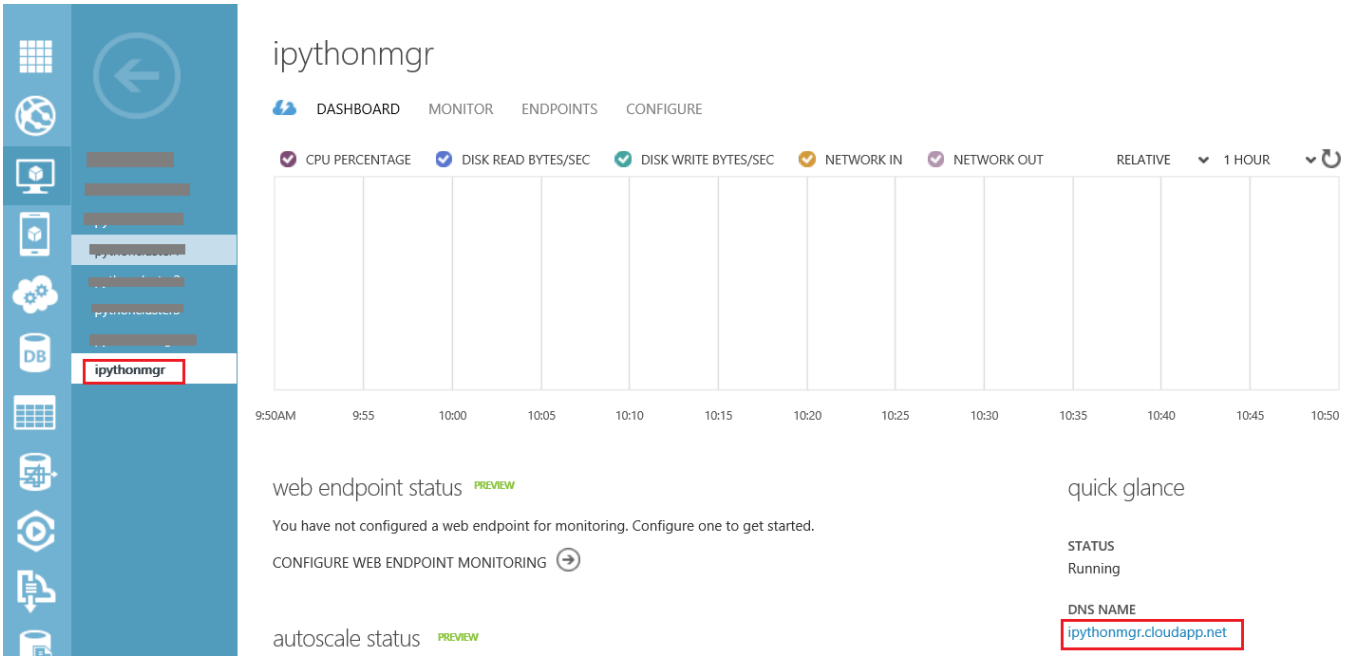
Task 1 - Deploy software on an Ubuntu Linux Server

1. On Windows Azure Management Porcal, click **New** -> **Compute** -> **Virtual Machine** -> **Quick Create**, input all required fields including *DNS Name*, *Image* = Ubuntu Server 12.04 LTS, *Size* = Small, *Password* and *Region*:



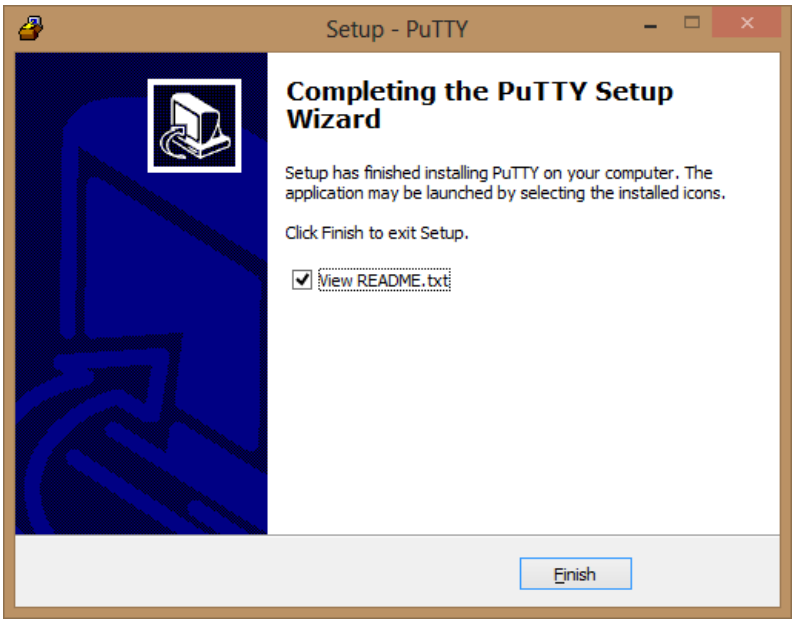
Create an Ubuntu VM

2. After the machine is created, we can use putty to connect to that machine from the DNS name. You can find the DNS name from the dashboard of the virtual machine.



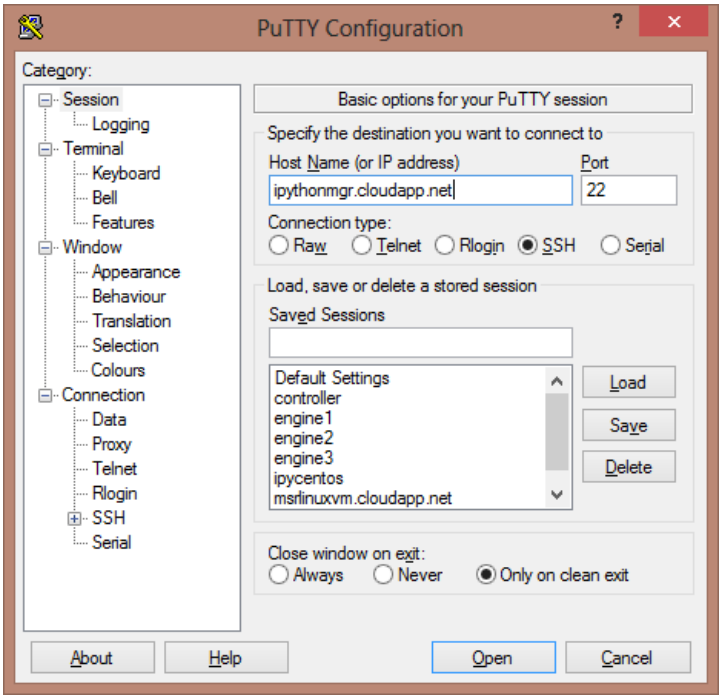
IPython Ubuntu DNS Name

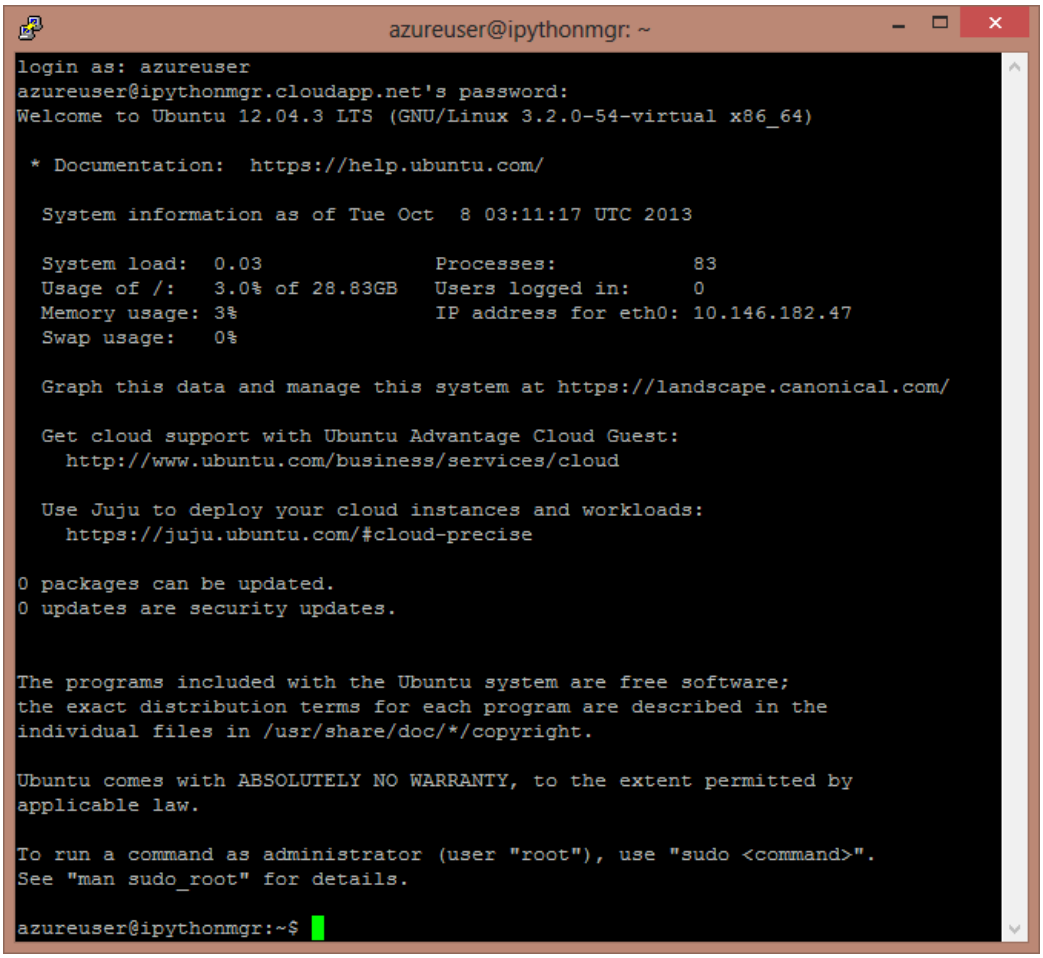
3. For Putty, you can download the Windows installer for everything except PuTTYtel and install it.



Install Putty

4. Launch Putty and connect to the remote machine with the DNS name.





Connect Remote Machine

5. Then we can execute following command to install git

```
sudo apt-get install git
```

```
azureuser@ipythonmgr:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-arch git-cvs git-svn
  git-email git-gui gitk gitweb
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,741 kB of archives.
After this operation, 15.2 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://azure.archive.ubuntu.com/ubuntu/ precise/main liberror-perl all 0.1
7-1 [23.8 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu/ precise/main git-man all 1:1.7.9.5
-1 [630 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu/ precise/main git amd64 1:1.7.9.5-1
 [6,087 kB]
Fetched 6,741 kB in 0s (19.9 MB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 50831 files and directories currently installed.)
Unpacking liberror-perl (from .../liberror-perl_0.17-1_all.deb) ...
Selecting previously unselected package git-man.
Unpacking git-man (from .../git-man_1%3a1.7.9.5-1_all.deb) ...
Selecting previously unselected package git.
Unpacking git (from .../git_1%3a1.7.9.5-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up liberror-perl (0.17-1) ...
Setting up git-man (1:1.7.9.5-1) ...
Setting up git (1:1.7.9.5-1) ...
```

Install Git

6. Execute following command to install Python Setup Tools.

```
sudo apt-get install python-setuptools
```

```
azureuser@ipythonmgr:~$ sudo apt-get install python-setuptools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  python-setuptools
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 441 kB of archives.
After this operation, 1,068 kB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu/ precise/main python-setuptools all
 0.6.24-1ubuntu1 [441 kB]
Fetched 441 kB in 0s (2,468 kB/s)
Selecting previously unselected package python-setuptools.
(Reading database ... 51480 files and directories currently installed.)
Unpacking python-setuptools (from .../python-setuptools_0.6.24-1ubuntu1_all.deb)
 ...
Setting up python-setuptools (0.6.24-1ubuntu1) ...
```

Install Python Setup Tools

7. Execute following commands to install Windows Azure SDK for Python

```
git clone https://github.com/WindowsAzure/azure-sdk-for-python.git
cd ~/azure-sdk-for-python
sudo python setup.py install
```

```
azureuser@ipythonmgr:~$ git clone https://github.com/WindowsAzure/azure-sdk-for-
python.git
Cloning into 'azure-sdk-for-python'...
remote: Counting objects: 617, done.
remote: Compressing objects: 100% (338/338), done.
remote: Total 617 (delta 314), reused 557 (delta 268)
Receiving objects: 100% (617/617), 3.20 MiB | 590 KiB/s, done.
Resolving deltas: 100% (314/314), done.
```

```
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/http/batchclient.py
to batchclient.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/http/httpclient.py t
o httpclient.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/http/winhttp.py to w
inhttp.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/http/__init__.py to
__init__.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/servicebus/servicebu
sservice.py to servicebusservice.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/azure/servicebus/__init__.
py to __init__.pyc
running install_egg_info
Writing /usr/local/lib/python2.7/dist-packages/azure-0.7.0.egg-info
```

Install Windows Azure SDK for Python

8. Install WorkerPool and Paramiko

```
sudo easy_install workerpool
sudo easy_install paramiko
```

```
Installed /usr/local/lib/python2.7/dist-packages/workerpool-0.9.2-py2.7.egg
Processing dependencies for workerpool
Finished processing dependencies for workerpool
azureuser@ipythonmgr:~/azure-sdk-for-python/src$ sudo easy_install paramiko
Searching for paramiko
Best match: paramiko 1.7.7.1
Adding paramiko 1.7.7.1 to easy-install.pth file

Using /usr/lib/python2.7/dist-packages
Processing dependencies for paramiko
Finished processing dependencies for paramiko
```

Install WorkerPool and Paramiko

All software has been installed on your machine, next we setup to connect to Windows Azure Portal by Python.

Task 2 - Setup Windows Azure Subscription

1. To connect to the Service Management endpoint, you need your Windows Azure subscription ID and the path to a valid management certificate. You can obtain your subscription ID through the management portal, and you can create management certificates in a number of ways. Here we use OpenSSL to create it.

   You actually need to create two certificates, one for the server (a .cer file) and one for the client (a .pem file). To create the .pem file, execute this:

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
   ```

   ```
   azureuser@ipythonmgr:~$ openssl req -x509 -nodes -days 365 -newkey rsa:1024 -key
   out mycert.pem -out mycert.pem
   Generating a 1024 bit RSA private key
   ..........................................++++++
   ..................++++++
   writing new private key to 'mycert.pem'
   -----
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:CN
   State or Province Name (full name) [Some-State]:Shanghai
   Locality Name (eg, city) []:Shanghai
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:Training
   Organizational Unit Name (eg, section) []:Azure
   Common Name (e.g. server FQDN or YOUR name) []:training.com
   Email Address []:training@azure.com
   ```

   Create Client Certificate File(a .pem file)

2. To create the .cer certificate, execute this:

```
openssl x509 -inform pem -in mycert.pem -outform der -out mycert.cer
```

You can use ls to view the file you created.

```
azureuser@ipythonmgr:~$ openssl x509 -inform pem -in mycert.pem -outform der -ou
t mycert.cer
azureuser@ipythonmgr:~$ ls
azure-sdk-for-python  mycert.cer  mycert.pem
```

Create Server Certificate File(a .cer file)

> For more information about Windows Azure certificates, see Managing Certificates in Windows Azure. For a complete description of OpenSSL parameters, see the documentation at http://www.openssl.org/docs/apps/openssl.html.

3. You need to use pscp to download the **mycert.cer** file to local for uploading to windows azure. Execute the following command in the command line:

```
cd [Your Putty Folder]
[Your Putty Folder]\pscp.exe -p [username]@[DNSName]:/home/[username]/mycert.cer mycert.cer
[Your Putty Folder]\pscp.exe -p [username]@[DNSName]:/home/[username]/mycert.pem mycert.pem
```

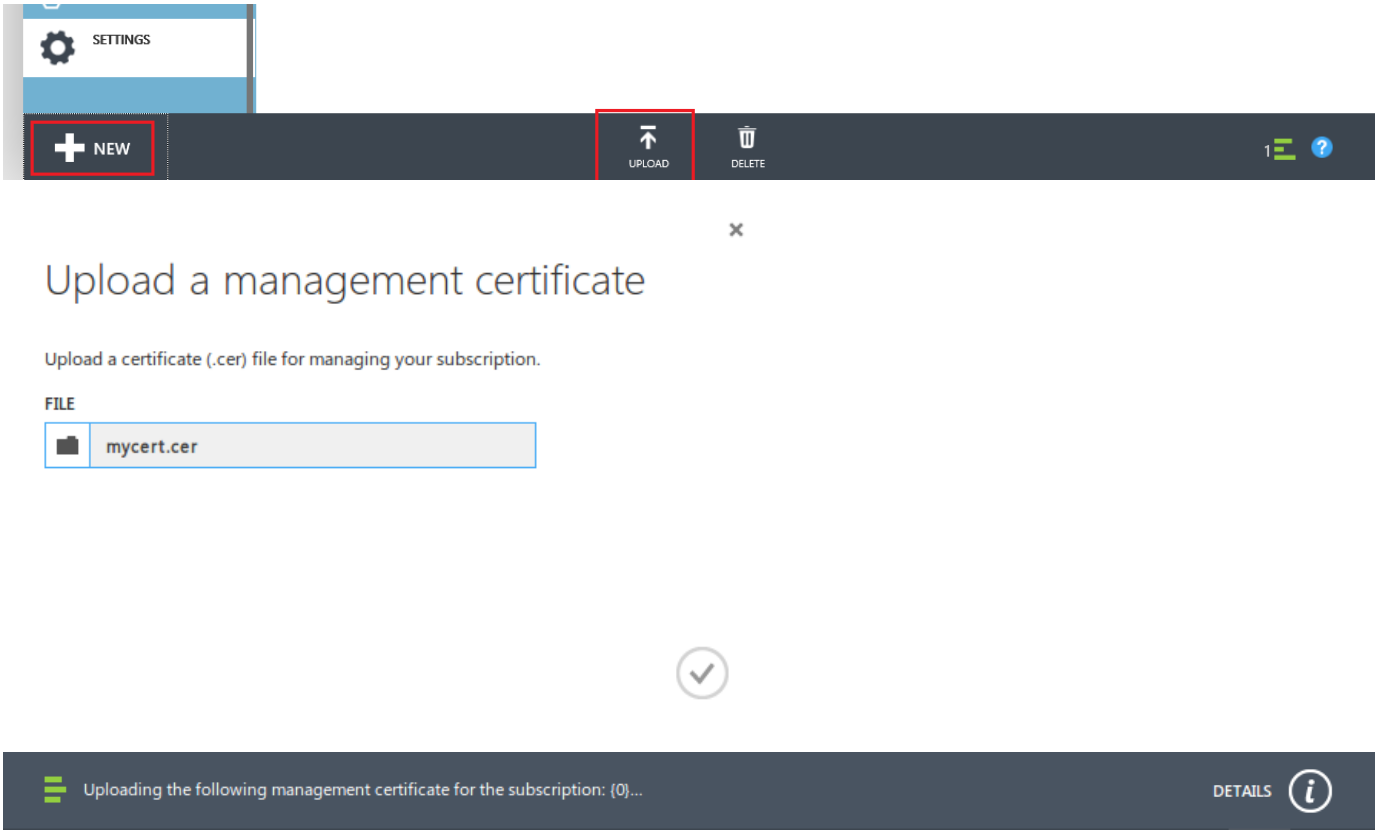
Just replace [Your Putty Folder],[username] and [DNSName].

```
c:\Program Files (x86)\PuTTY>pscp.exe -p azureuser@ipythonmgr.cloudapp.net:/home
/azureuser/mycert.cer mycert.cer
azureuser@ipythonmgr.cloudapp.net's password:
mycert.cer                 | 0 kB |   0.7 kB/s | ETA: 00:00:00 | 100%
```

Donwload cer file

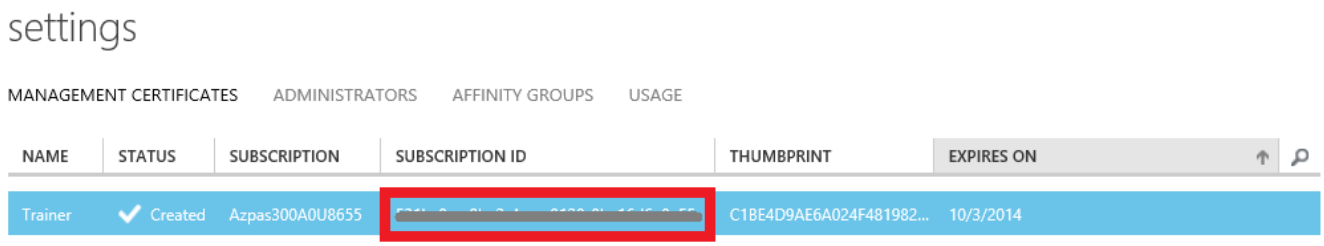
> If you get an error about "Cannot create file xxx", please run your command line in administrators mode and run again.

4. After you have downloaded the file *mycert.cer*, you will need to upload the .cer file to Windows Azure via the "Upload" action of the "Settings" tab of the management portal, and you will need to make note of where you saved the .pem file.



Upload .cer file to Windows Azure

5. Please also note the subscription id for future use.

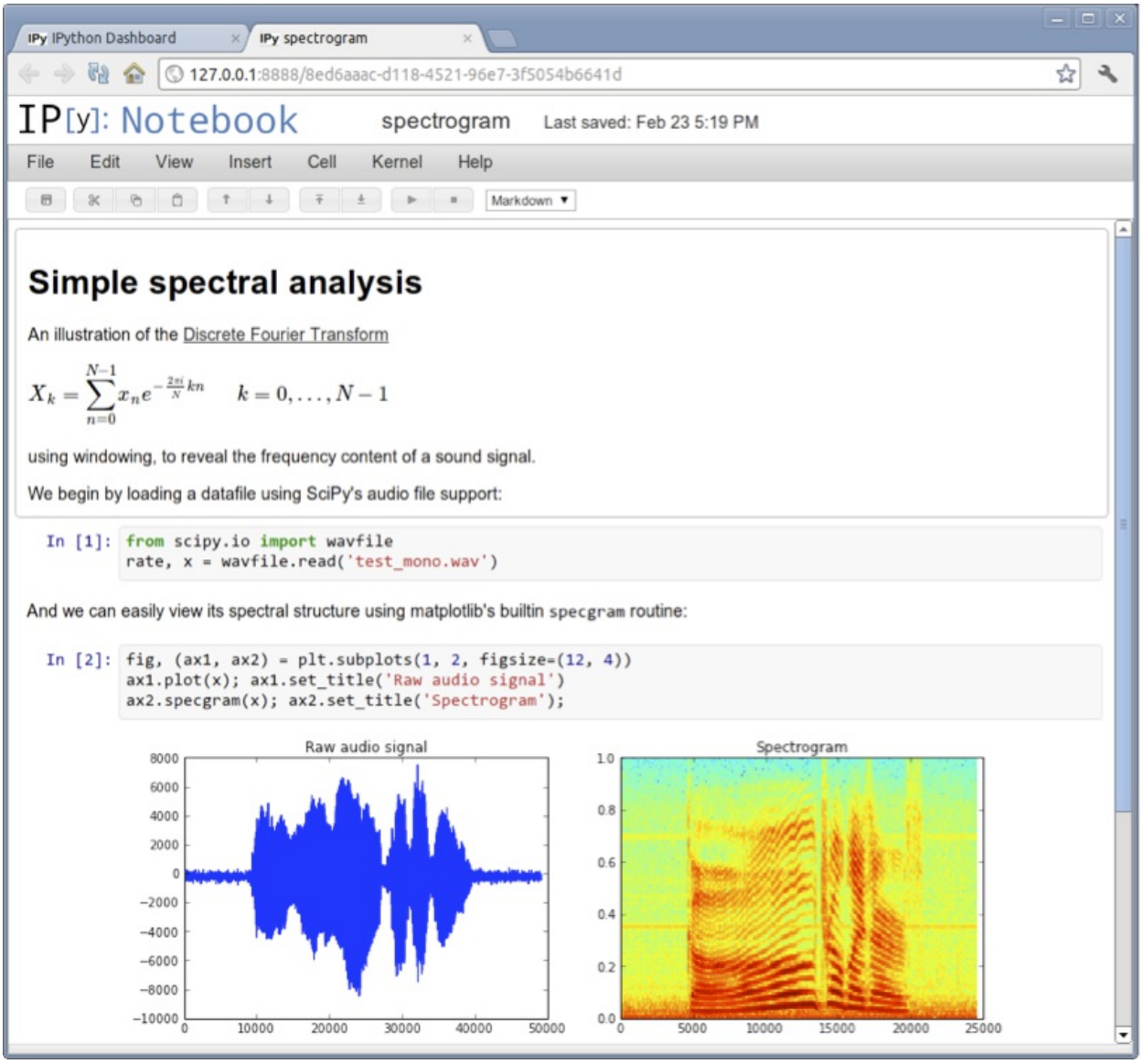


Get Subscription Id

Excerise 2 - Deploy IPython Notebook on Windows Azure

The IPython project provides a collection of tools for scientific computing that include powerful interactive shells, high-performance and easy to use parallel libraries and a web-based environment called the IPython Notebook. The Notebook provides a working environment for interactive computing that combines code execution with the creation of a live computational document. These notebook files can contain arbitrary text, mathematical formulas, input code, results, graphics, videos and any other kind of media that a modern web browser is capable of displaying.

Whether you're absolutely new to Python and want to learn it in a fun, interactive environment or do some serious parallel/technical computing, the IPython Notebook is a great choice. As an illustration of its capabilities, the following screenshot shows the IPython Notebook being used, in combination with the SciPy and matplotlib packages, to analyze the structure of a sound recording:



IPython Notebook Spectral

1. First we need to copy the toolkit under **Source\Ex02-DeployIPython** to local Ubuntu machine, then copy the **mycert.pem** to the same folder of the source.

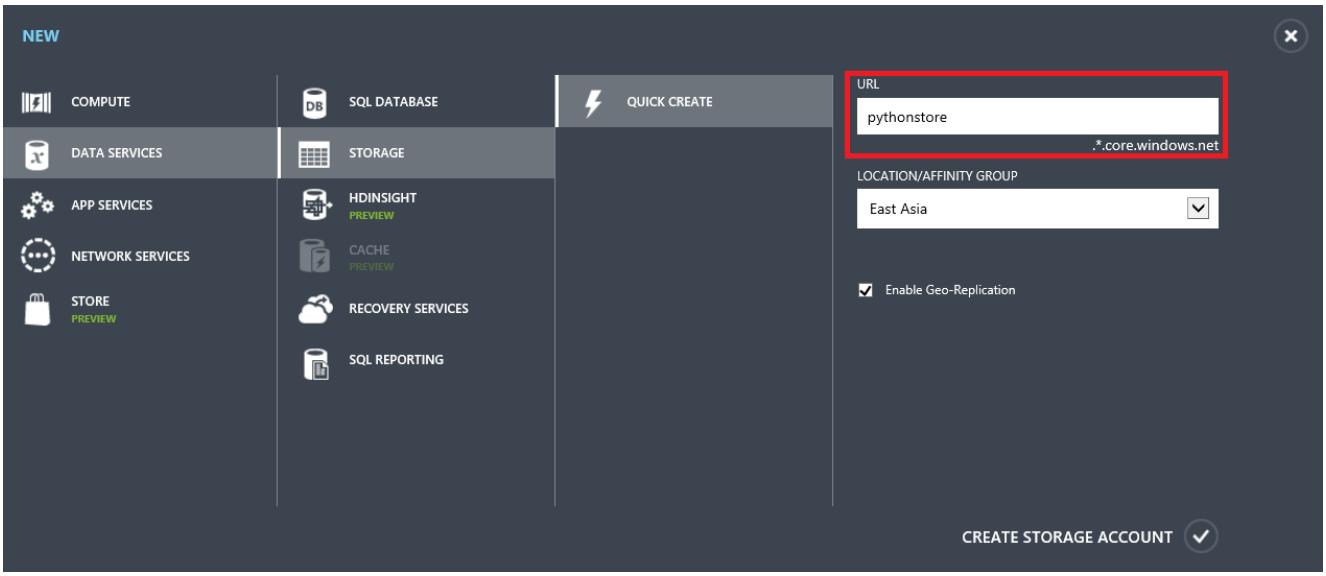
2. Open the file **configSample.py** in gedit. You need to replace the subscription id with yours and the path to your private key file (**mycert.pem**).

```
###################################
#         Azure Settings          #
###################################
# the subscription id, replace <subscription_id> to your real id
subscription_id = "<subscription id>"
# the pem file path, replace <pem_file_name> to your real pem file name
pem_path = "<pem_file_name>"

# the number of vms
# one controller and n - 1 engines
# default 2
num_vm = 2

# service_name should be globally unique
# service_name should only contain letters, numbers and hyphen, the length should be less th
# suggest: service_name, deployment_name and role_name set to the same
service_name = "<service_name>"
deployment_name = "<deployment name>"
role_name = "<role name>"

# Please replace the <storage_account_name> to real storage account name in your subscriptio

media_link_base = "http://<storage_account_name>.blob.core.windows.net/vhds/ipythonvm"
# location of the vm
location = "<your region>"
```

Edit configSample.py File

You may also need to change the following sections in that files including, **The number of nodes**, **The name of VM nodes**, **Service Name**, **Deployment Name**, **Role Name**, **Media Link Base**, **The password of the Notebook**.

> For the "Media Link Base", you must set in **http://.blob.core.windows.net/vhds/ipythonvm** to your correct storage account. If you don't have any storage account under your subscription, you can just create one. Replace to your real region like "East Asia" or "East US". Now we set the number of nodes to 3, which means 1 controller and 2 engines.



Create Storage Account

3. Next we need to upload all files under **Source\Ex02-DeployIPython**. First create a folder in Putty.

```
cd ~
mkdir Ex02-DeployIPython
```

```
azureuser@ipythonmgr:~$ cd ~
azureuser@ipythonmgr:~$ mkdir Ex02-DeployIPython
```

Create Folder

Then we run following command in Comand line:

```
"[You Putty Folder]\pscp.exe" -r "[Source\Ex02 filder]\*" [username]@[DNSName]:/home/[username]/Ex02-DeployIPython
```

```
d:\Ex02-DeployIPython>"c:\Program Files (x86)\PuTTY\pscp.exe" -r d:\Ex02-DeployI
Python * azureuser@ipythonmgr.cloudapp.net:/home/azureuser/Ex02-DeployIPython
azureuser@ipythonmgr.cloudapp.net's password:
configSample.py            | 7 kB |   7.8 kB/s | ETA: 00:00:00 | 100%
configSample.pyc           | 6 kB |   6.3 kB/s | ETA: 00:00:00 | 100%
createMultiVM.py           | 3 kB |   3.4 kB/s | ETA: 00:00:00 | 100%
creatVM.py                 | 12 kB |  12.4 kB/s | ETA: 00:00:00 | 100%
creatVM.pyc                | 11 kB |  11.1 kB/s | ETA: 00:00:00 | 100%
installMultiSoftware.py    | 1 kB |   1.3 kB/s | ETA: 00:00:00 | 100%
installSoftware.py         | 4 kB |   4.3 kB/s | ETA: 00:00:00 | 100%
main.py                    | 1 kB |   1.5 kB/s | ETA: 00:00:00 | 100%
mycert.cer                 | 0 kB |   0.7 kB/s | ETA: 00:00:00 | 100%
mycert.pem                 | 1 kB |   1.9 kB/s | ETA: 00:00:00 | 100%
ssh.py                     | 7 kB |   7.7 kB/s | ETA: 00:00:00 | 100%
ssh.pyc                    | 7 kB |   7.9 kB/s | ETA: 00:00:00 | 100%
util.py                    | 0 kB |   0.3 kB/s | ETA: 00:00:00 | 100%
util.pyc                   | 0 kB |   0.6 kB/s | ETA: 00:00:00 | 100%
configSample.py            | 7 kB |   7.8 kB/s | ETA: 00:00:00 | 100%
configSample.pyc           | 6 kB |   6.3 kB/s | ETA: 00:00:00 | 100%
createMultiVM.py           | 3 kB |   3.4 kB/s | ETA: 00:00:00 | 100%
creatVM.py                 | 12 kB |  12.4 kB/s | ETA: 00:00:00 | 100%
creatVM.pyc                | 11 kB |  11.1 kB/s | ETA: 00:00:00 | 100%
installMultiSoftware.py    | 1 kB |   1.3 kB/s | ETA: 00:00:00 | 100%
installSoftware.py         | 4 kB |   4.3 kB/s | ETA: 00:00:00 | 100%
main.py                    | 1 kB |   1.5 kB/s | ETA: 00:00:00 | 100%
mycert.cer                 | 0 kB |   0.7 kB/s | ETA: 00:00:00 | 100%
mycert.pem                 | 1 kB |   1.9 kB/s | ETA: 00:00:00 | 100%
ssh.py                     | 7 kB |   7.7 kB/s | ETA: 00:00:00 | 100%
ssh.pyc                    | 7 kB |   7.9 kB/s | ETA: 00:00:00 | 100%
util.py                    | 0 kB |   0.3 kB/s | ETA: 00:00:00 | 100%
util.pyc                   | 0 kB |   0.6 kB/s | ETA: 00:00:00 | 100%
```

Copy Files

4. Execute **main.py** with following command:

```
python main.py [start|create|deploy|delete]
```

start creates VMs and deploy IPython; **create** just creates VMs; **deploy** deploys IPython on existing VMs depending on 'create'; **delete** removes all resources on Windows Azure.

Now we execute with **start**.

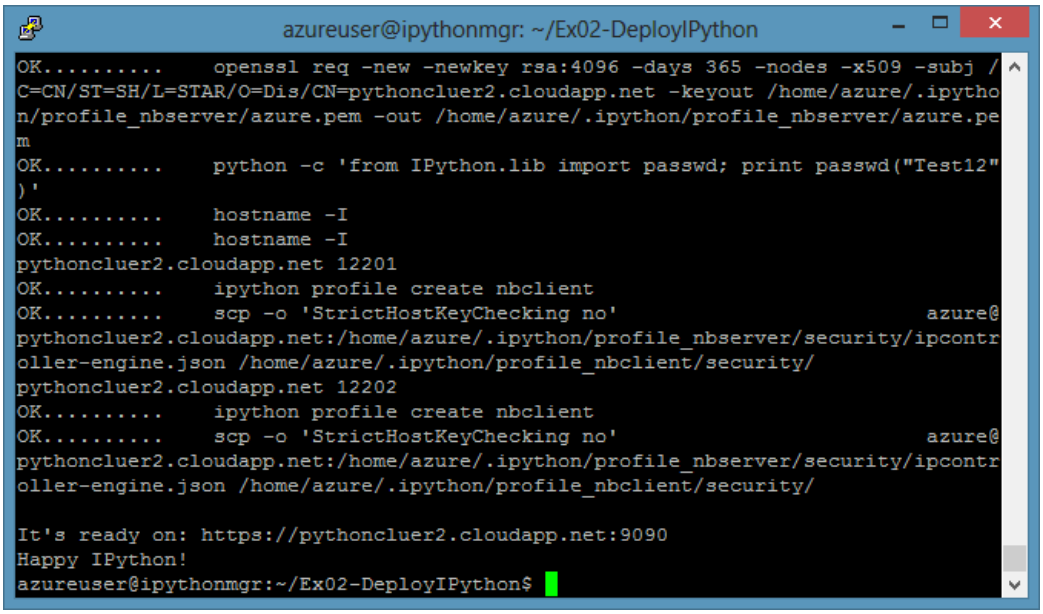
```
python main.py start
```

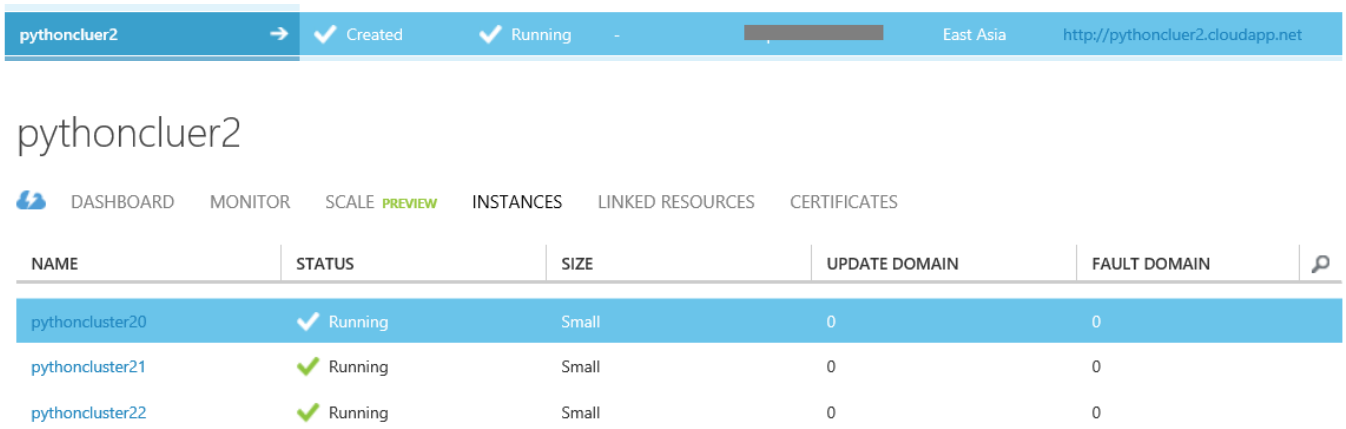
```
azureuser@ipythonmgr:~/Ex02-DeployIPython$ python main.py start
```

Execute Python Commands

At first, the **start** command creates a cloud service with the **Service Name** you defined in the configuration file. Then it creates 4 small instances in the cloud service. After those machines are launched, the code will connect to those machines and deploy required software and IPython Notebook automatically.

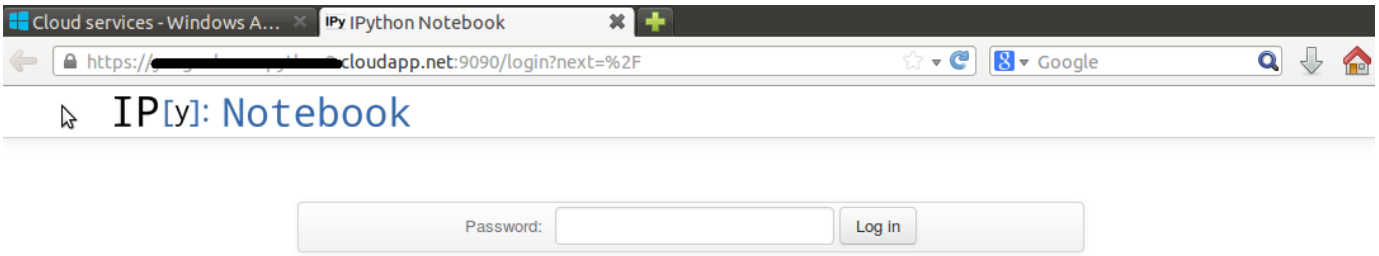
5. After about 10 minutes, the deployment is done. You will see the IPython cloud service is running in Windows Azure. There are 3 small instances running.





IPython Notebook is running on Windows Azure

Just click the link on windows azure and you will see IPython is ready. If you see warnings for certification issue, just ignore it and continue.



IPython Notebook

Excerise 3 - Run Monte Carlo Simulation on IPython

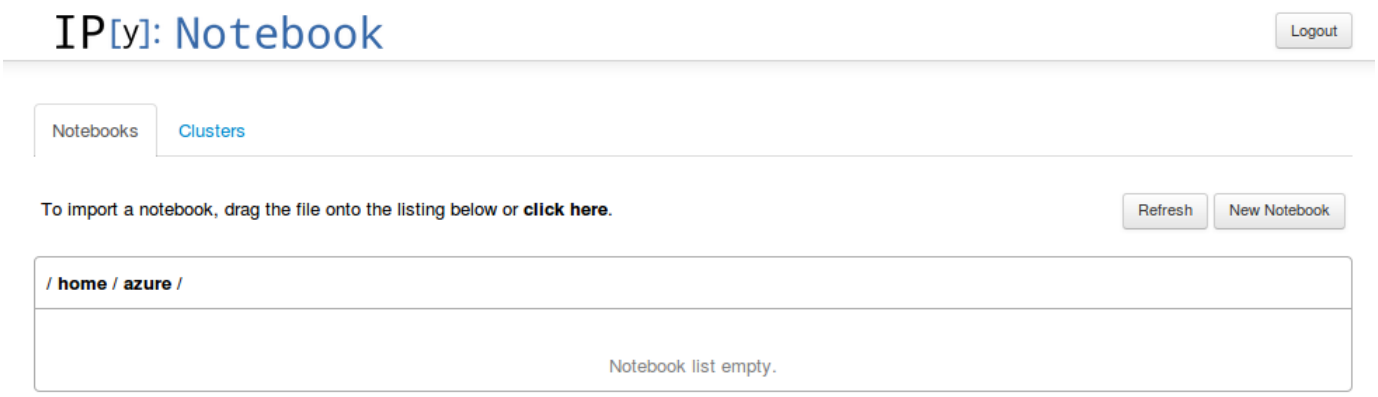
Monte Carlo simulation is a computerized mathematical technique that allows people to account for risk in quantitative analysis and decision making. The technique is used by professionals in such widely disparate fields as finance, project management, energy, manufacturing, engineering, research and development, insurance, oil & gas, transportation, and the environment.

Monte Carlo simulation furnishes the decision-maker with a range of possible outcomes and the probabilities they will occur for any choice of action.. It shows the extreme possibilities—the outcomes of going for broke and for the most conservative decision—along with all possible consequences for middle-of-the-road decisions.

The technique was first used by scientists working on the atom bomb; it was named for Monte Carlo, the Monaco resort town renowned for its casinos. Since its introduction in World War II, Monte Carlo simulation has been used to model a variety of physical and conceptual systems.

In this execise, you will run a Monte Carlo simulation code in your IPython notebook. This notebook shows how to use IPython.parallel to do Monte-Carlo options pricing in parallel. We will compute the price of a large number of options for different strike prices and volatilities, where each task will consist of computing the option price for a single strike price and volatility.

1. Login your IPython Notebook with your predefined password, it is **Test12** if you didn't change it.



Login IPython Notebook

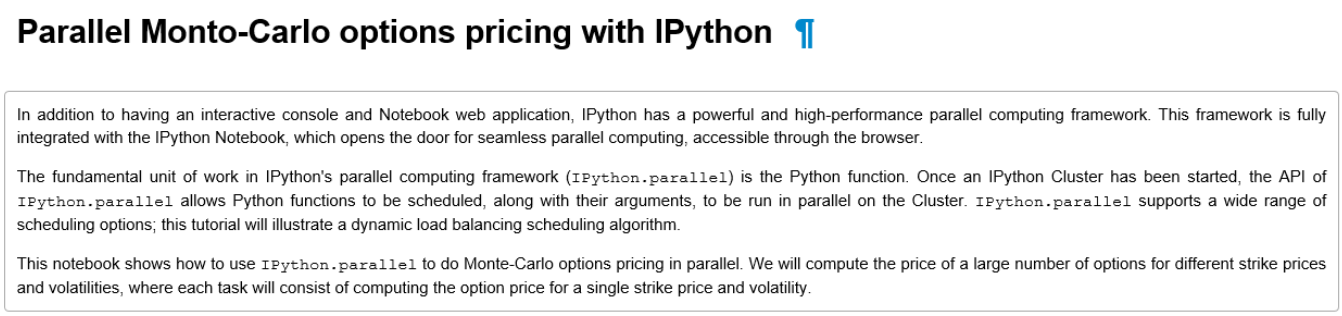
2. Create a new notebook. Execute the following command in a cell.

```
!wget https://raw.github.com/wenming/BigDataSamples/master/ipythonMLsamples/Cluster%20-%20ParallelMCOptions.ipynb
```

```
In [2]:
!wget https://raw.github.com/wenming/BigDataSamples/master/ipythonMLsamples/Cluster%20-%20ParallelMCOptions.ipynb

--2013-09-13 12:54:10--  https://raw.github.com/wenming/BigDataSamples/master/ipythonMLsamples/Cluster%20-%20ParallelMCOptions.ipynb
Resolving raw.github.com (raw.github.com)... 103.245.222.133
Connecting to raw.github.com (raw.github.com)|103.245.222.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 80446 (79K) [text/plain]
Saving to: `Cluster - ParallelMCOptions.ipynb'

100%[======================================>] 80,446      382K/s   in 0.2s

2013-09-13 12:54:11 (382 KB/s) - `Cluster - ParallelMCOptions.ipynb' saved [80446/80446]
```

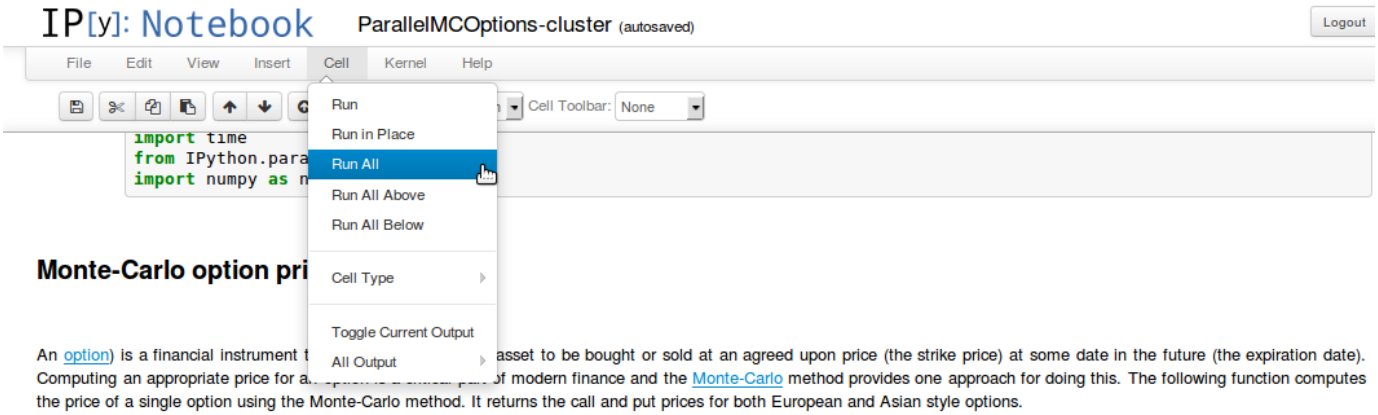
Load Monte Carlo Simulation

3. Click the Notebooks tab. There is a new notebook called **ParallelMCOptions-cluster**, click the notebook.



Cluster - ParallelMCOptions

4. Click **Cell->Run All** to execute the sample. It will run the simulation in parallel.



Execute Monte Carlo Simulation in Parallel on Azure

The IPython Notebook and IPython.parallel enable you to parallelize your code on a remote cluster using nothing more than a web browser. As this example shows, once you have a Python function that performs a unit of work, it is easy to invoke that function in parallel for different arguments. The example shown here is extremely simple; the full API is rich and powerful. Details can be found in the IPython Documentation.

After couple of minutes, you will see the result in the page.

```
In [10]:
```

```
%%timeit -n1 -r1

global async_results
async_results = []
for strike in strike_vals:
    for sigma in sigma_vals:
        # This line submits the tasks for parallel computation.
        ar = view.apply_async(price_option, price, strike, sigma, rate, days, paths)
        async_results.append(ar)

c.wait(async_results)  # Wait until all tasks are done.
```

```
1 loops, best of 1: 2min 56s per loop
```

```
In [11]:
```

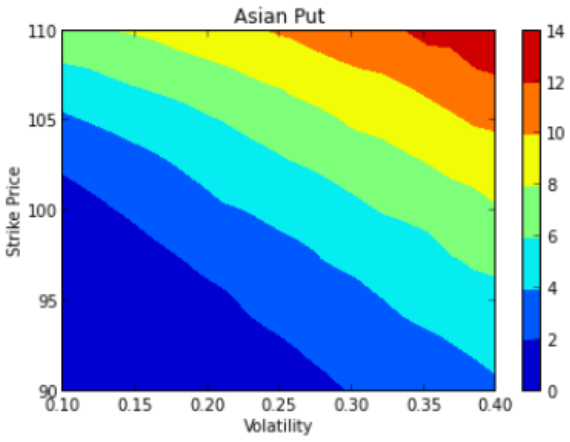
```
len(async_results)
```

```
Out[11]:
```

```
400
```

```
Out[17]:
```

```
<matplotlib.text.Text at 0x3c53410>
```



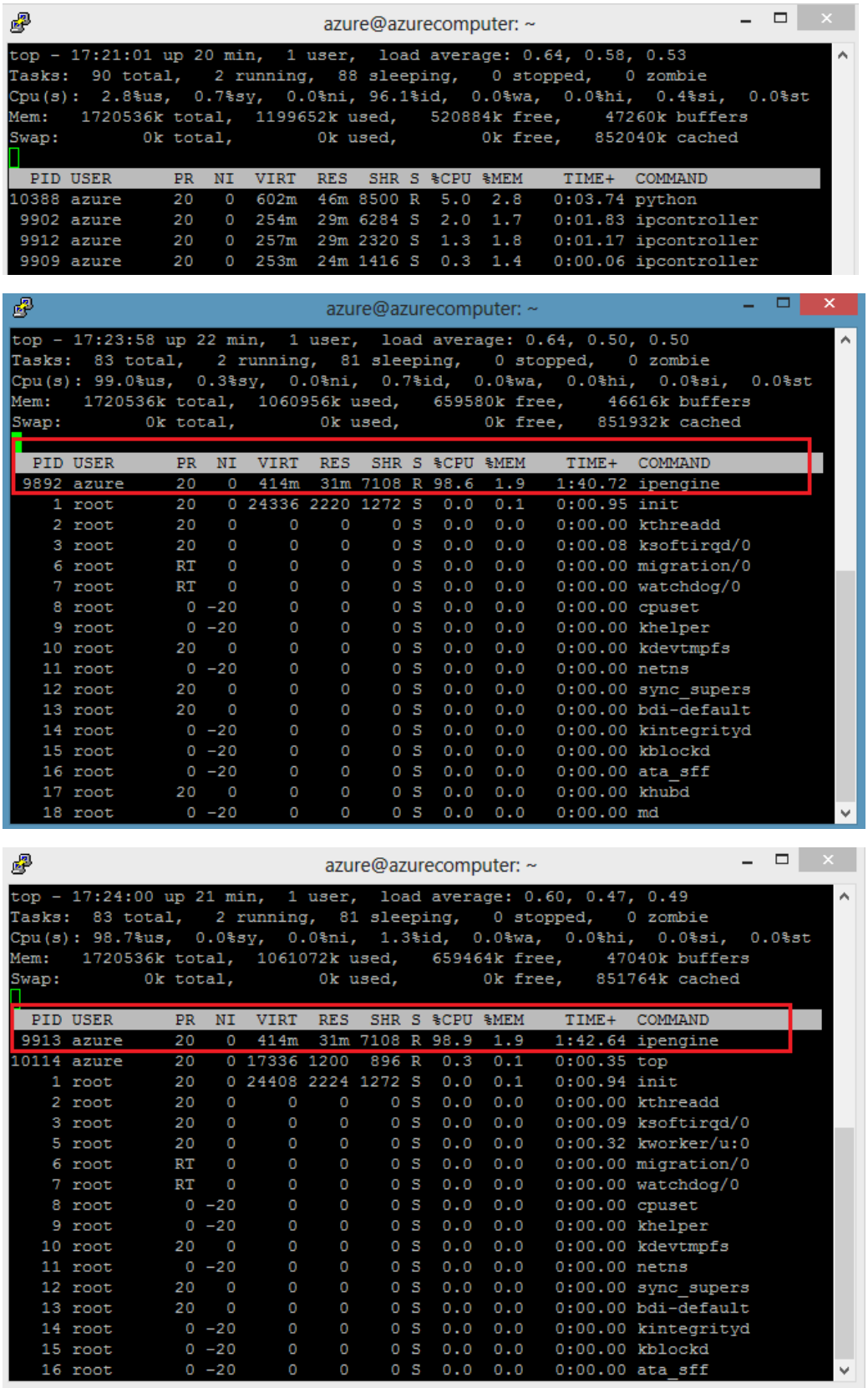
Monte Carlo Simulation Result

5. We can use SSH to controller and engines to check the CPU load for different machines. Run *top* command to check the most CPU

consumption process.

```
top
```

During the Monte Carlo is executing, the ipengine consumes almost 100% CPU on all engine machines.



IPython Cluster CPU consumption

Summary

By completing this hands-on lab you learned the following:

- Provision Virtual machines with Python.
- Deploy IPython notebook on your virtual machines.
- Run Monte Carlo Simulation on IPython in parallel.

If you would like to read more about IPython cluster: http://ipython.org/ipython-doc/dev/parallel/parallel_process.html