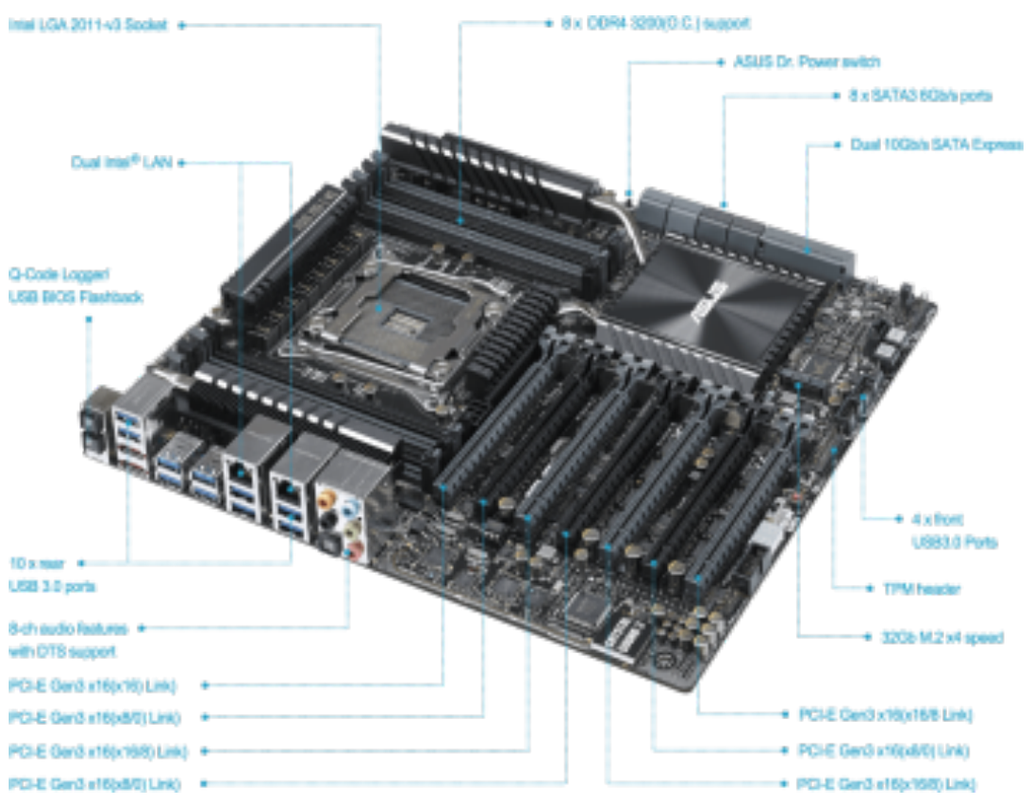
Tensorflow 실행 환경 설정

2016. 11. 25

Alluser.net Corp.

DNN 시스템 구축을 위한 하드웨어 고려사항

- 다양한 DNN 실험을 위하여 환경 설정이 중요
  - 병렬 GPU시스템 구축을 위한 하드웨어 선정
  - PCI-E 버스의 병목, CPU-GPU간 병목
  - CPU-GPU간의 메모리 사용문제
- 메인보드 고려사항
  - Gen3 16배속 GPU를
  - 많이 꽂을 수 있을 수록 유리
  - 대개의 보드의 경우 2개 이상 꽂으면 속도 저하



실험환경 구축

- Nvidia 계열의 GPU 준비
  - AMD 계열은 CUDA를 사용할 수 없음
  - GTX 1080 : 32비트 부동소수점 연산 능력이 가격 대비 우수
  - TITAN X : 32비트 부동소수점 연산에서 현존하는 가장 빠른 GPU(국내에 판매되고 있지 않음)

실험환경 구축

- Ubuntu OS (16.04)

  - 최근 대다수의 DNN 연구자들이 선호, 쉬운 설치 및 정보가 많음
  - APT(Advanced Packaging Tool)를 통한 패키지 관리의 용이성

- Python 3 (혹은 Python 2.7)

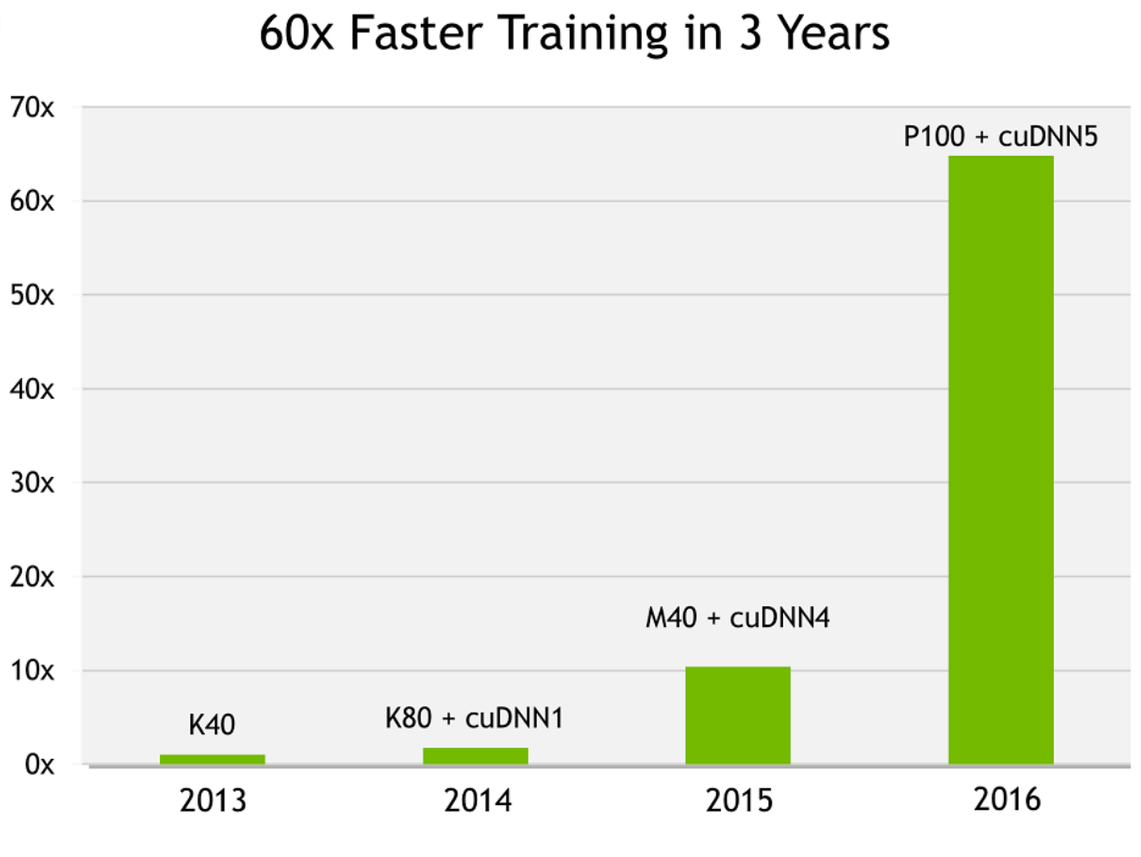
  - Ubuntu와 더불어 DNN 연구분야에서는 Python이 대세를 이루고 있으며, Ubuntu에는 python 2.7과 python 3.5 가 기본으로 설치되어 있음
  - 스크립트 언어로 진입장벽이 낮고 라이브러리가 많음

실험환경 구축

- GPU 드라이버 설치
  - 그래픽카드 모델에 맞는 최신 드라이버 설치
  - nvidia-smi 명령으로 설치 확인
- CUDA 설치 (CUDA 8.0)
  - CUDA는 빠른 발전을 하고 있으며 버전별로 연산속도 차이가 큼
  - 현재 연구에 활용할 프레임워크의 지원 여부를 함께 검토해야 함.
- cuDNN 설치 (cuDNN 5.1)
  - 설치 시 잊기 쉬움. 없을 경우 DNN 처리 속도가 느려짐.
  - DNN에서 사용한 표준적인 forward and backward convolution, pooling, normalization, and activation layers 등을 높은 성능으로 제공

cuDNN 에 따른 성능변화

cuDNN 4 + K40 vs. cuDNN 5.1 + M40 on Torch and Intel Xeon Haswell single-socket 16-core E5-2698 v3@2.3Ghz 3.6GHz Turbo



AlexNet training throughput on:
CPU: 1x E5-2680v3 12 Core 2.5GHz. 128GB System Memory, Ubuntu 14.04

실험환경 구축

- 딥러닝 프레임워크 설치
  - Keras : 입문자에게 추천 (Python)
    - https://keras.io/
  - Theano, TensorFlow : 연구자에게 추천 (Python)
    - Theano : http://deeplearning.net/software/theano/
    - TensorFlow : https://www.tensorflow.org/
- 그 외, Torch (Lua), Caffe (C++), DL4J (Java) 등.

서버접속 방법

- SSH 접속
  - Windows : 한글 PuTTY 사용
    - 다운로드 링크 : https://github.com/teamnop/HPuTTY/releases
  - OS X 또는 Linux 계열 : 터미널에서 아래 명령을 이용하여 접속 가능
    - ssh sogang2016@163.239.169.54
- 계정정보 (실습용)
  - ID : sogang2016
  - PW : sogang2016

튜토리얼 다운로드

- 디렉토리 생성
  - mkdir directory-name (디렉토리 명은 휴대폰 뒷자리로 생성)
  - cd directory-name
- 다음을 입력하여 튜토리얼 다운로드
  - git clone https://github.com/AlluserNet/TensorFlowTutorials.git

```
lifefeel@auc-dnn1:~$ git clone https://github.com/AlluserNet/TensorFlowTutorials.git
Cloning into 'TensorFlowTutorials'...
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 20 (delta 4), reused 19 (delta 3), pack-reused 0
Unpacking objects: 100% (20/20), done.
Checking connectivity... done.
lifefeel@auc-dnn1:~$
```

- 디렉토리를 이동하여 파일 확인
  - cd TensorFlowTutorials
  - ls

```
lifefeel@auc-dnn1:~/TensorFlowTutorials$ ls
00_multiply.ipynb           03_linear_regression.ipynb      07_word2vec_kor.ipynb               README.md
00_multiply.py              04_logistic_regression.ipynb    08_recurrent_neural_network.ipynb   word2vec.png
01_helloworld.ipynb         05_multilayer_perceptron.ipynb  09_tensorboard.ipynb
02_basic_operations.ipynb   06_convolutional_network.ipynb  input_data.py
```

Python을 이용한 Tensorflow 코드 실행

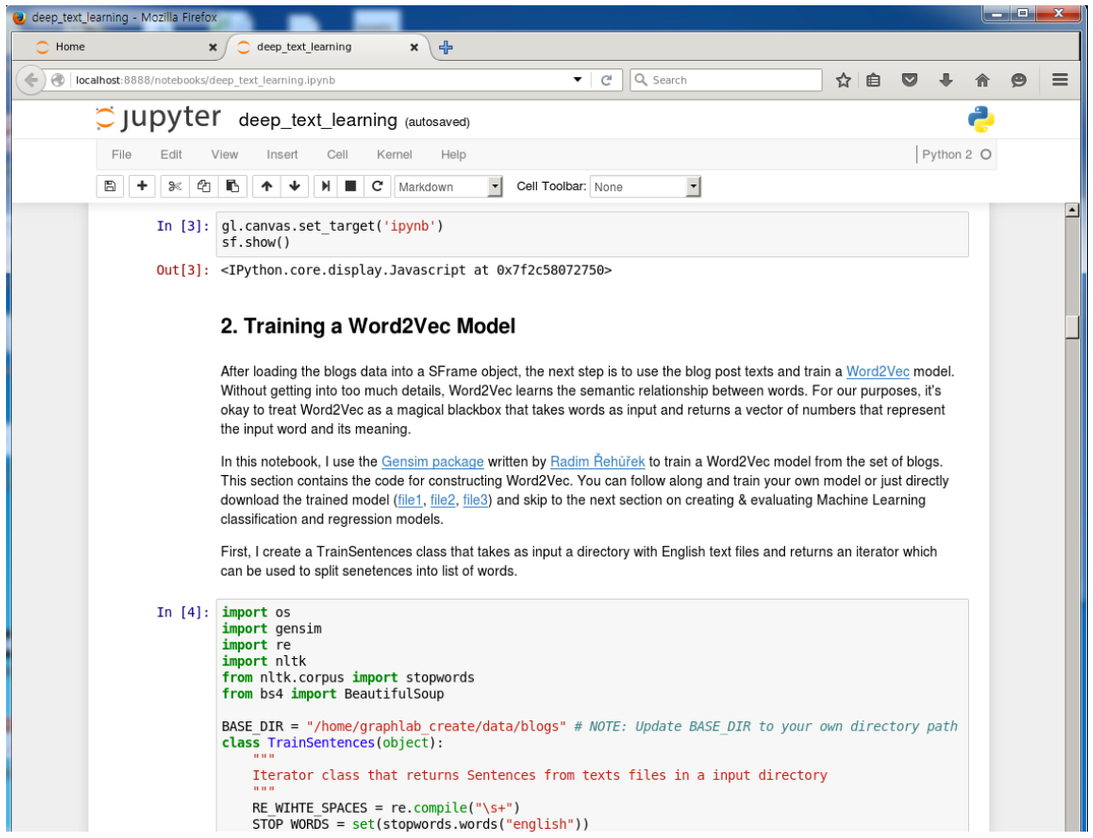
- 다음명령을 입력
  - python3 00_multiply.py
  - (python2를 사용할 경우 python 00_multiply.py 로 실행)

```
lifefeel@auc-dnn1:~/TensorFlow-Tutorials$ python3 00_multiply.py
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcurand.so locally
I tensorflow/core/common_runtime/gpu/gpu_device.cc:951] Found device 0 with properties:
name: GeForce GTX 1080
major: 6 minor: 1 memoryClockRate (GHz) 1.797
pciBusID 0000:0a:00.0
Total memory: 7.92GiB
Free memory: 7.81GiB
W tensorflow/stream_executor/cuda/cuda_driver.cc:572] creating context when one is currently active; existing: 0x3027a30
I tensorflow/core/common_runtime/gpu/gpu_device.cc:951] Found device 1 with properties:
name: GeForce GTX 1080
major: 6 minor: 1 memoryClockRate (GHz) 1.797
pciBusID 0000:09:00.0
Total memory: 7.92GiB
Free memory: 7.81GiB


2.000000 should equal 2.0
9.000000 should equal 9.0
```

iPython Notebook 소개(1/2)

- iPython은 interactive 환경에서 프로그래밍이 가능하도록 만들어진 shell입니다.
- iPython notebook은 iPython을 web-based 환경에서 파이썬의 도큐멘테이션을 효과적으로 하기 위해 만들어졌지만, 여러모로 편한점이 있어 이를 이용하여 코딩하는 사람들이 있습니다.

IP[y]: IPython
Interactive Computing



iPython Notebook 소개(2/2)

- 코드작성 뿐만 아니라 Markdown 도 지원하며, 최근에는 다른 언어도 지원합니다. (iPython notebook에서 Jupyter 로 바뀜)

- 보통은 본인의 local 환경에 설치하고 사용하지만, 서버의 경우, ssh 접속은 텍스트만 제공하기 때문에 웹 브라우저를 띄울 수가 없습니다.

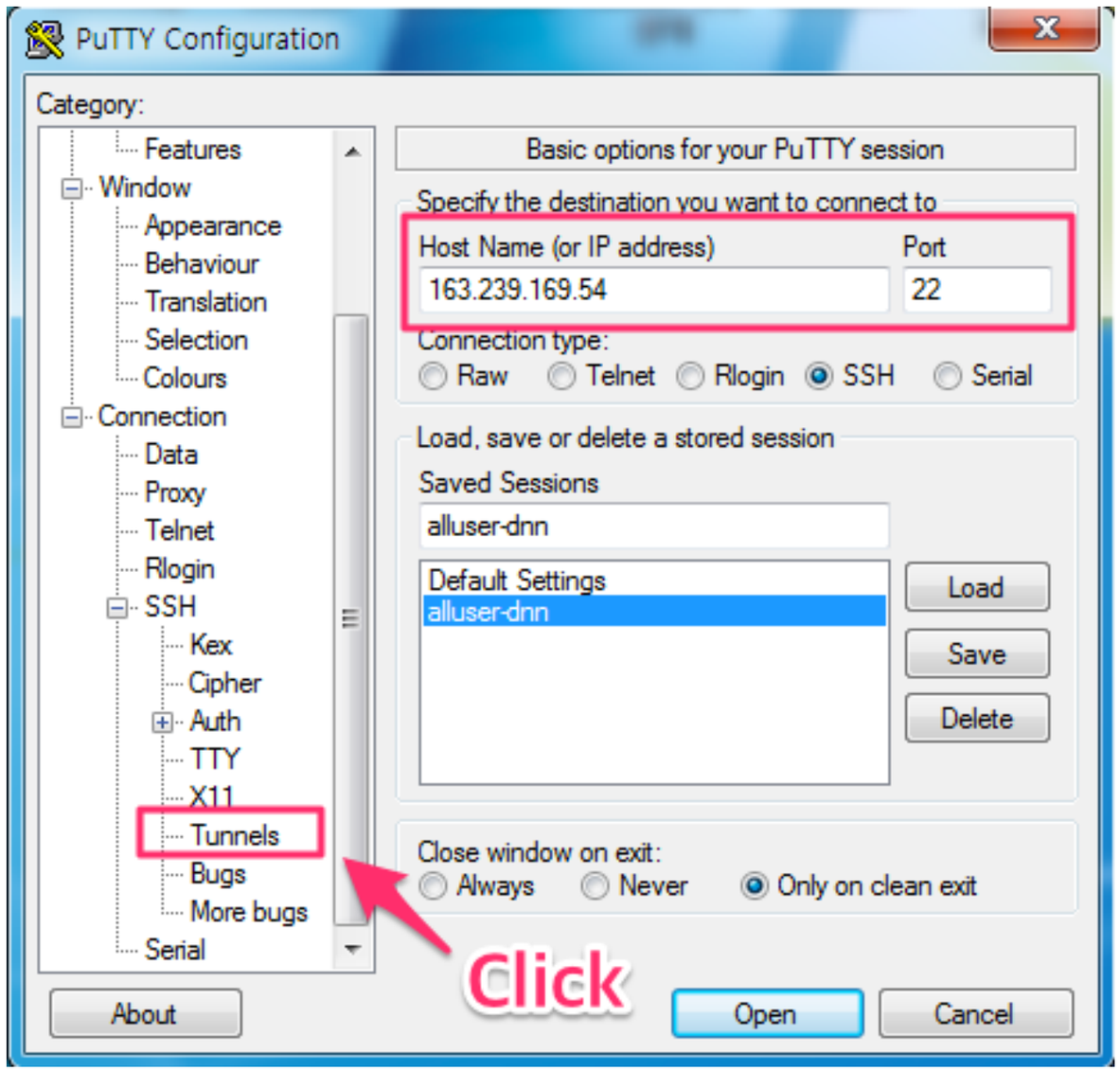
- 이러한 경우, SSH Ternel을 설정하면 22번 포트를 통해서 Remote의 포트를 Local의 포트로 포워딩하여 사용할 수 있습니다.

- 관련자료
  - 공식 홈페이지 : http://ipython.org/

SSH Terneling 설정(1/3)

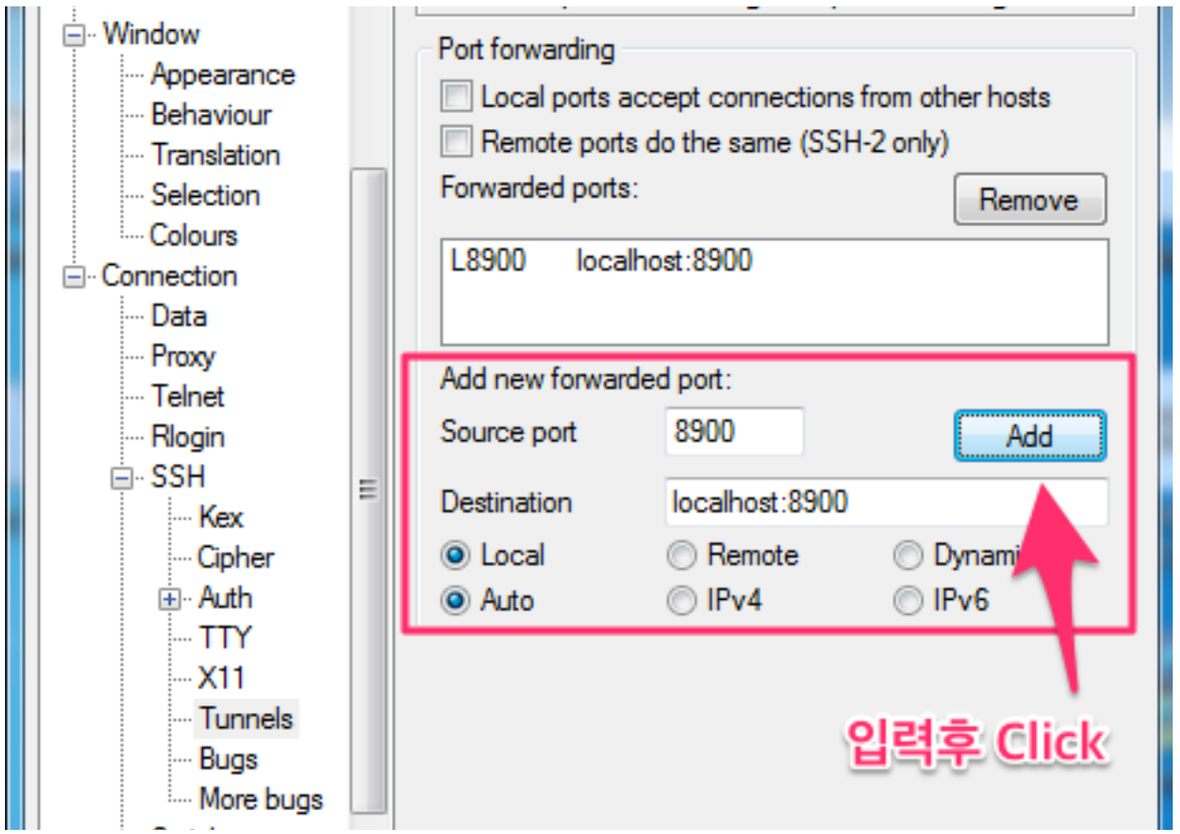
- 이전 페이지에서 iPython을 실행할 때 지정한 포트번호를 내 웹브라우저에서 실행하기 위해서는 SSH Terneling을 설정합니다.

- PuTTY 프로그램을 설치 후 아래와 같이 서버 정보를 입력한 후, SSH -> Ternels를 클릭합니다.



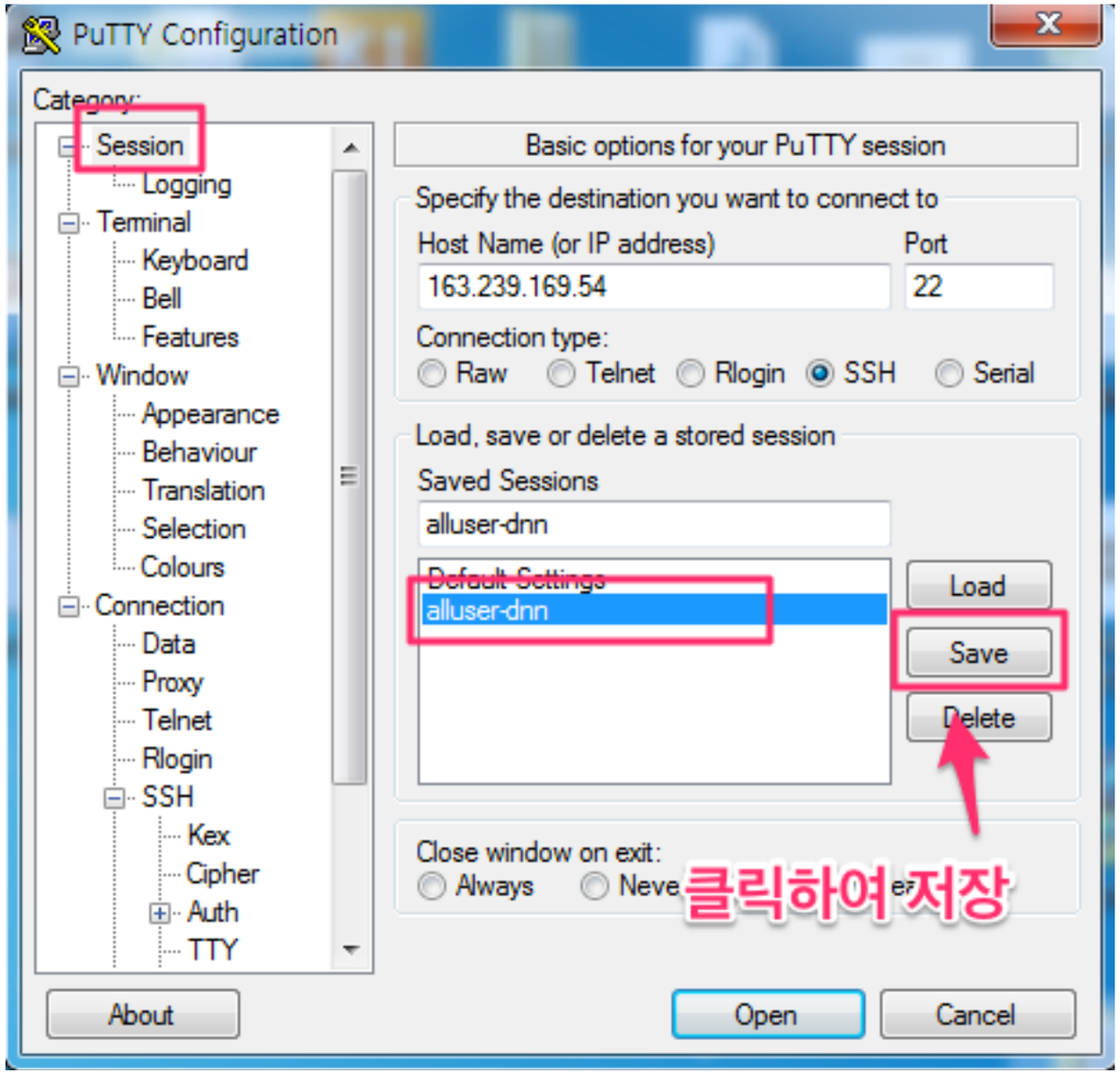
SSH Terneling 설정(2/3)

- SSH Ternel을 설정하면 22번 포트를 통해서 Remote의 포트를 Local의 포트로 포워딩하여 사용할 수 있습니다.
  - Source port : localhost에서 사용할 포트번호
  - Destination : remote 환경의 포트번호
- iPython을 사용하기 위한 포트 및 Tensorboard를 위한 포트 총 2개를 미리 지정합니다.
  - 예 : 8900 (ipython), 8901 (tensorboard)



SSH Terneling 설정(3/3)

- Session 항목으로 돌아와 Saved Sessions에 저장할 이름을 입력 후 Save 버튼을 클릭합니다.
- 저장 후, 목록에서 이름을 더블클릭 하시면 서버에 접속하는 화면이 뜹니다.



Unix/Linux 사용자의 경우 접속방법

- ssh -N -L localhost:8900:localhost:8900 -L localhost:8901:localhost:8901 username@host_address
  - 이 접속은 ssh를 이용하여 Terneling 만을 위한 접속입니다.
  - 위 명령 실행 후, PW 입력하고 나면 아무 반응이 없는 것이 정상입니다.
  - 옵션 설명
    - -N : 원격 명령을 실행하지 않는 상태로 유지(포트포워딩만 이용 시)
    - -L : forwarding 할 포트번호 지정 (bind_address:port:host:hostport)
- 참고 Url : https://coderwall.com/p/ohk6cg/remote-access-to-ipython-notebooks-via-ssh

ipython 실행하기(1/5)

- 우선 이전 이전 페이지에서 설명한 SSH Terneling을 이용하여 포트포워딩 설정 및 실행(접속)을 합니다.
- 새로운 쉘을 접속 후, 다음을 입력하여 ipython을 실행합니다.
    - cd TensorFlowTutorials
    - ipython3 notebook --port=8900
    - (포트번호는 일반적으로 사용하지 않는 포트를 지정)

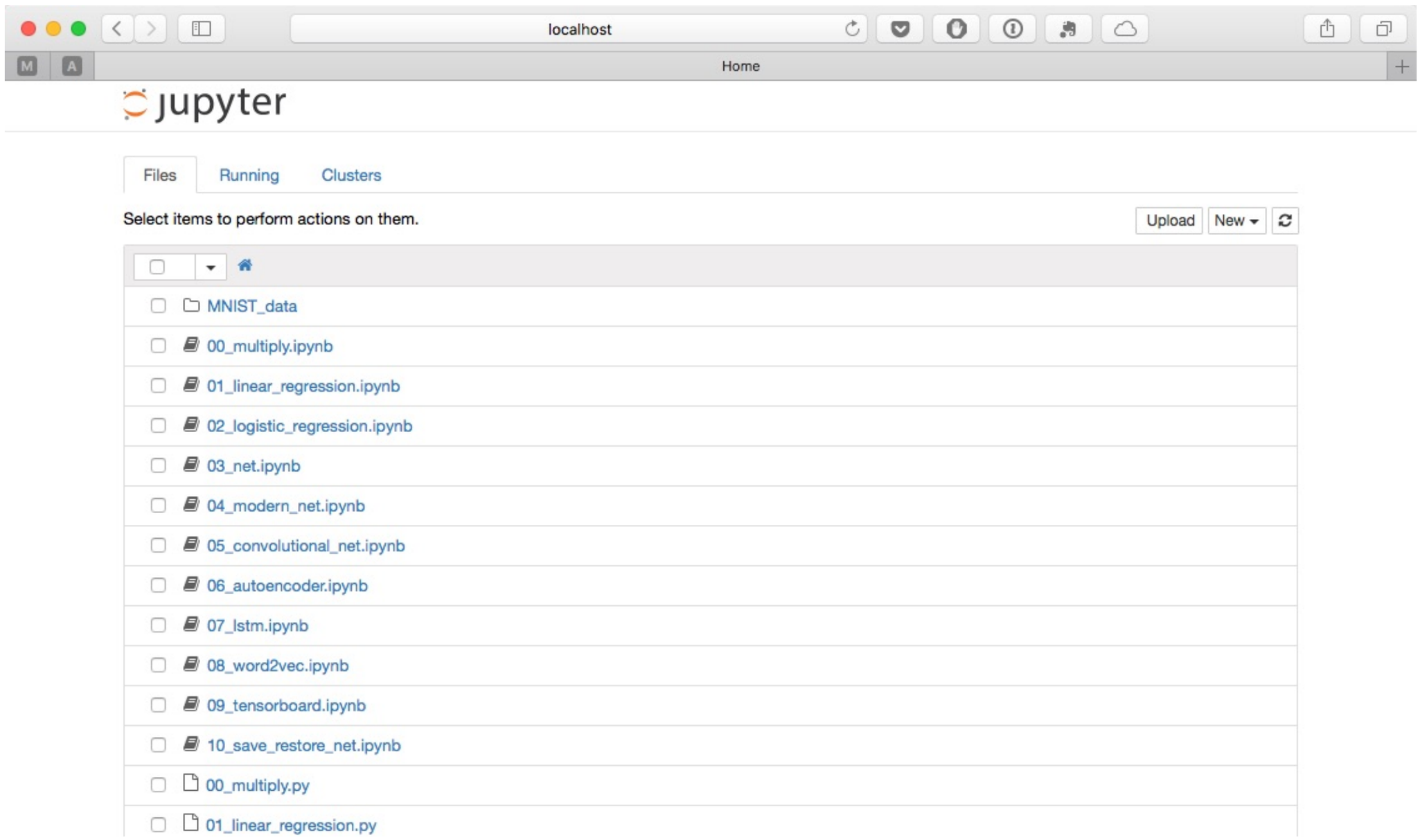
```
lifefeel@auc-dnn1:~/TensorFlow-Tutorials$ ipython notebook --port=8900
[TerminalIPythonApp] WARNING | Subcommand `ipython notebook` is deprecated and will be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter notebook` in the future
[I 00:20:31.630 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook_cookie_secret
[W 00:20:31.641 NotebookApp] Widgets are unavailable. Please install widgetsnbextension or ipywidgets 4.0
[I 00:20:31.651 NotebookApp] Serving notebooks from local directory: /home/lifefeel/TensorFlow-Tutorials
[I 00:20:31.651 NotebookApp] 0 active kernels
[I 00:20:31.651 NotebookApp] The Jupyter Notebook is running at: http://localhost:8900/
[I 00:20:31.651 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 00:20:31.651 NotebookApp] No web browser found: could not locate runnable browser.
```

- 만약 사용중인 포트를 이용하여 실행할 경우 아래와 같은 메시지가 발생합니다.
- 이런 경우엔 임의의 포트로 실행되므로 가능한 서로 다른 포트를 사용하시는 것이 좋습니다.
    - (다른 포트로 사용하고자 할 경우 PuTTY에서 forwarding 설정을 변경해야 함)

```
lifefeel@auc-dnn1:~/TensorFlow-Tutorials$ ipython notebook --port=8900
[TerminalIPythonApp] WARNING | Subcommand `ipython notebook` is deprecated and will be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter notebook` in the future
[W 00:21:53.430 NotebookApp] Widgets are unavailable. Please install widgetsnbextension or ipywidgets 4.0
[I 00:21:53.434 NotebookApp] The port 8900 is already in use, trying another port.
[I 00:21:53.437 NotebookApp] Serving notebooks from local directory: /home/lifefeel/TensorFlow-Tutorials
[I 00:21:53.438 NotebookApp] 0 active kernels
[I 00:21:53.438 NotebookApp] The Jupyter Notebook is running at: http://localhost:8901/
[I 00:21:53.438 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 00:21:53.438 NotebookApp] No web browser found: could not locate runnable browser.
```

ipython 실행하기(2/5)

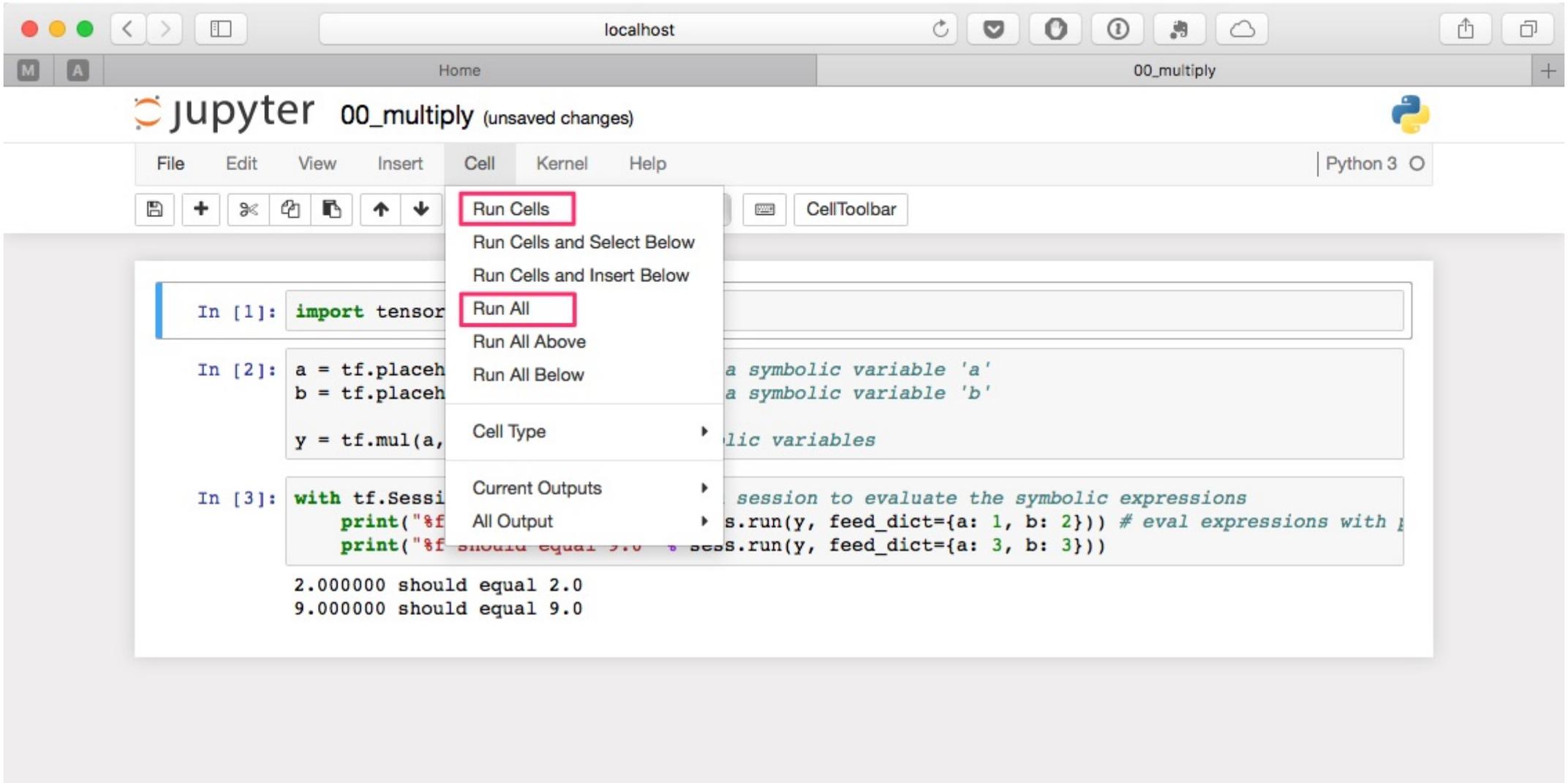
- 웹브라우저를 열어서 다음을 입력하면 ipython이 실행되는 것을 확인하실 수 있습니다.
  - http://localhost:8900



ipython 실행하기(3/5)

- 셀 단위로 코드를 실행시키기 위해서는 Cell->Run을 클릭하거나 셀 위에서 Shift + 엔터키를 누르시면 실행 할 수 있습니다.

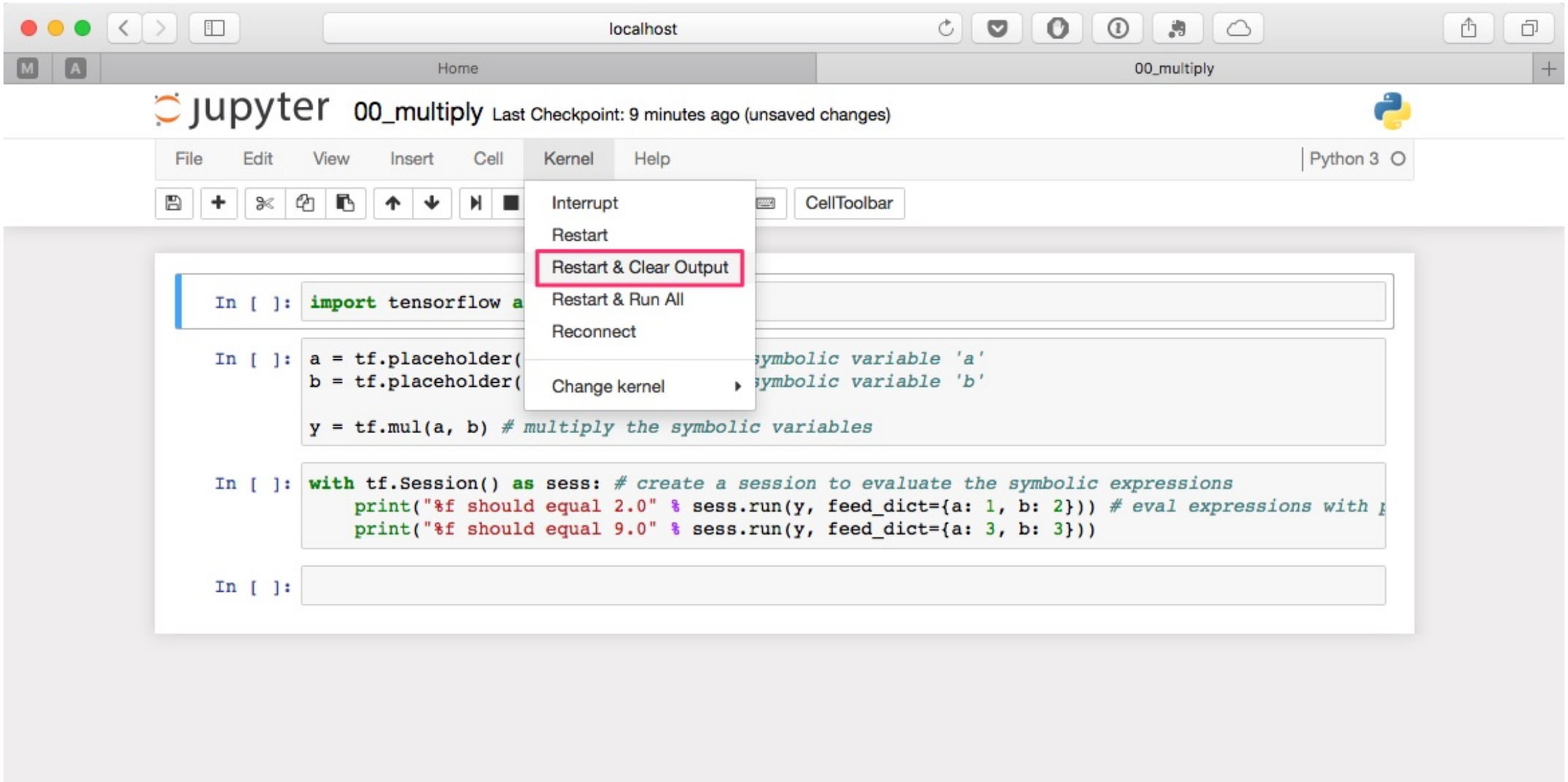
- 전체 코드를 실행하기 위해서는 Cell->Run All을 클릭합니다.



ipython 실행하기(4/5)

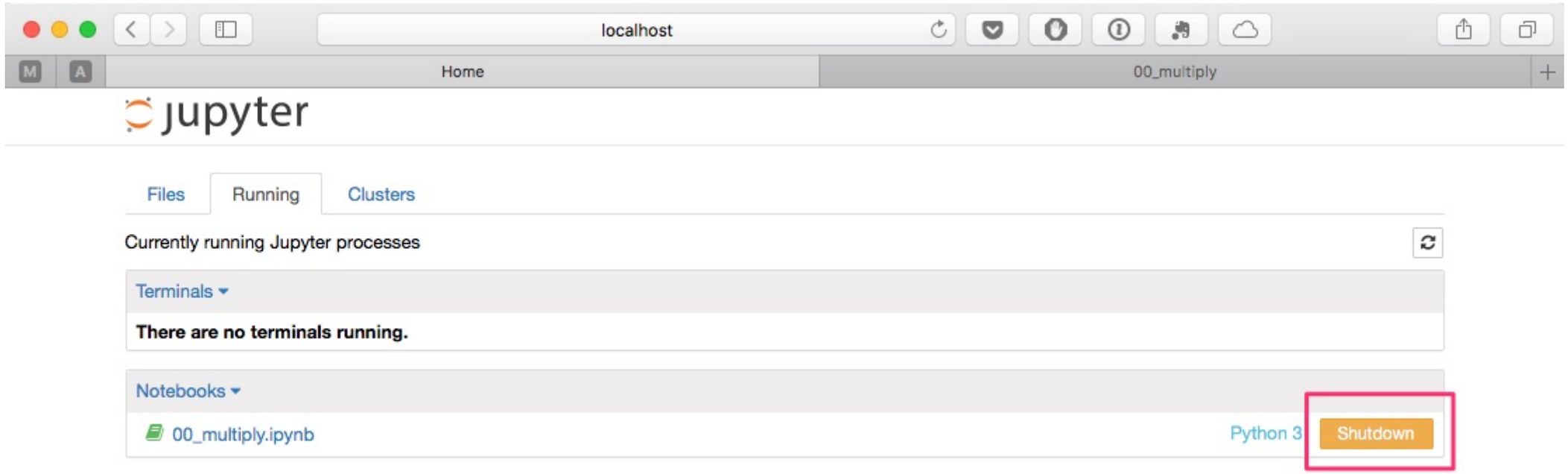
- ipython은 종료하기 전 까지는 점유한 메모리를 해지 하지 않습니다.

- 따라서 ipython이 실행된 상태에서 파일 브라우저를 통해 다른 코드를 실행하시려면 창을 닫으시기 전에 Kernel -> Restart & Clear Output을 실행하여야 합니다.



ipython 실행하기(5/5)

- 메모리를 해제하는 다른 방법으로는 파일 브라우저에서 Running 탭을 클릭하신 후 현재 실행중인 Notebooks를 Shutdown 하여 해제하실 수 있습니다.



사용중인 GPU 및 프로세스 확인

- Tensorflow를 기본값을 이용하여 사용할 경우, GPU:0 만 사용합니다.
- 아래 명령을 통해 현재 사용중인 GPU 및 프로세스를 확인 가능합니다.
    - nvidia-smi
    - watch -n 0.5 nvidia-smi (상태를 계속 관찰하고자 할 경우)
- GPU를 사용하고 있을 경우 Memory-Usage에 메모리 사용량과 GPU-Util 항목에 GPU 사용률이 표시됩니다.

```
auc@auc-dnn1 ~ » nvidia-smi
Thu Oct 27 16:29:42 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.48                 Driver Version: 367.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX TIT...  Off  | 0000:05:00.0      On |                  N/A |
| 22%   52C    P8    16W / 250W |     56MiB / 12201MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX TIT...  Off  | 0000:06:00.0     Off |                  N/A |
| 22%   53C    P8    16W / 250W |      1MiB / 12206MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 1080    Off  | 0000:09:00.0     Off |                  N/A |
|  0%   54C    P8    16W / 240W |      1MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce GTX 1080    Off  | 0000:0A:00.0     Off |                  N/A |
|  0%   52C    P8    16W / 240W |      1MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      1224    G   /usr/lib/xorg/Xorg                             55MiB |
+-----------------------------------------------------------------------------+
```

Tensorflow 메모리 할당 설정

- Tensorflow를 기본값을 이용하여 사용할 경우, 설치된 모든 GPU의 Maximum 메모리를 할당하기 때문에 동시에 여러명이 돌릴 수 없습니다.

- 아래의 옵션을 주어 Tensorflow가 필요한 만큼만 할당하도록 설정 가능합니다.

  - session_conf = tf.ConfigProto()

  - session_conf.gpu_options.allow_growth = **True**

  - session = tf.Session(config=session_conf)

Tensorflow 사용할 GPU 지정

- GPU 번호는 nividia-smi 에서 표시되는 것과 Tensorflow에서 사용하는 번호가 다르며 Tensorflow로 구현한 코드 실행 시 아래와 같이 GPU 번호와 실제 디바이스 매핑 정보를 확인 할 수 있습니다.

```
855] cannot enable peer access from device ordinal 3 to device ordinal 1
972] DMA: 0 1 2 3
982] 0:   Y Y N N
982] 1:   Y Y N N
982] 2:   N N Y Y
982] 3:   N N Y Y
1041] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 1080, pci bus id: 0000:0a:00.0)
1041] Creating TensorFlow device (/gpu:1) -> (device: 1, name: GeForce GTX 1080, pci bus id: 0000:09:00.0)
1041] Creating TensorFlow device (/gpu:2) -> (device: 2, name: GeForce GTX TITAN X, pci bus id: 0000:06:00.0)
1041] Creating TensorFlow device (/gpu:3) -> (device: 3, name: GeForce GTX TITAN X, pci bus id: 0000:05:00.0)
```

- 아래 코드를 통해 GPU 번호를 지정 가능합니다.

  - **with** tf.device('/gpu:0'):
    a = tf.placeholder("float") # Create a symbolic variable 'a'
    b = tf.placeholder("float") # Create a symbolic variable 'b'

    y = tf.mul(a, b) # multiply the symbolic variables

- tf 로 시작하는 것에 대해서는 tf.device 의 하위로 넣어야 지정한 GPU를 사용하게 되며, 설정되지 않은 연산에 대해서는 기본 GPU (gpu:0)을 사용하게 되어 데이터 교환에 대한 비효율이 발생할 수 있습니다.

Tensorflow Multiply 예제 (옵션 미적용)

- **import** tensorflow **as** tf

  a = tf.placeholder(**"float"**) # Create a symbolic variable 'a'
  b = tf.placeholder(**"float"**) # Create a symbolic variable 'b'

  y = tf.mul(a, b) # multiply the symbolic variables

  with tf.Session() **as** sess: # create a session to evaluate the symbolic expressions
  print(**"%f should equal 2.0"** % sess.run(y, feed_dict={a: 1, b: 2})) # eval expressions with parameters for a and b
  print(**"%f should equal 9.0"** % sess.run(y, feed_dict={a: 3, b: 3}))

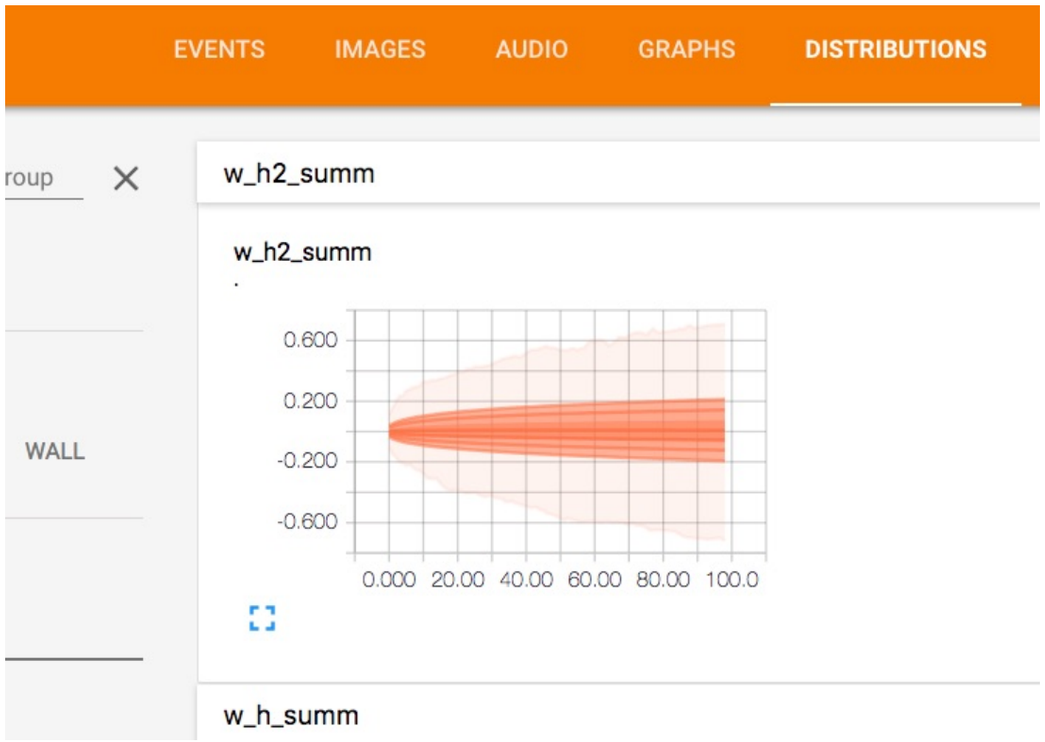
Tensorflow Multiply 예제 (옵션 적용)

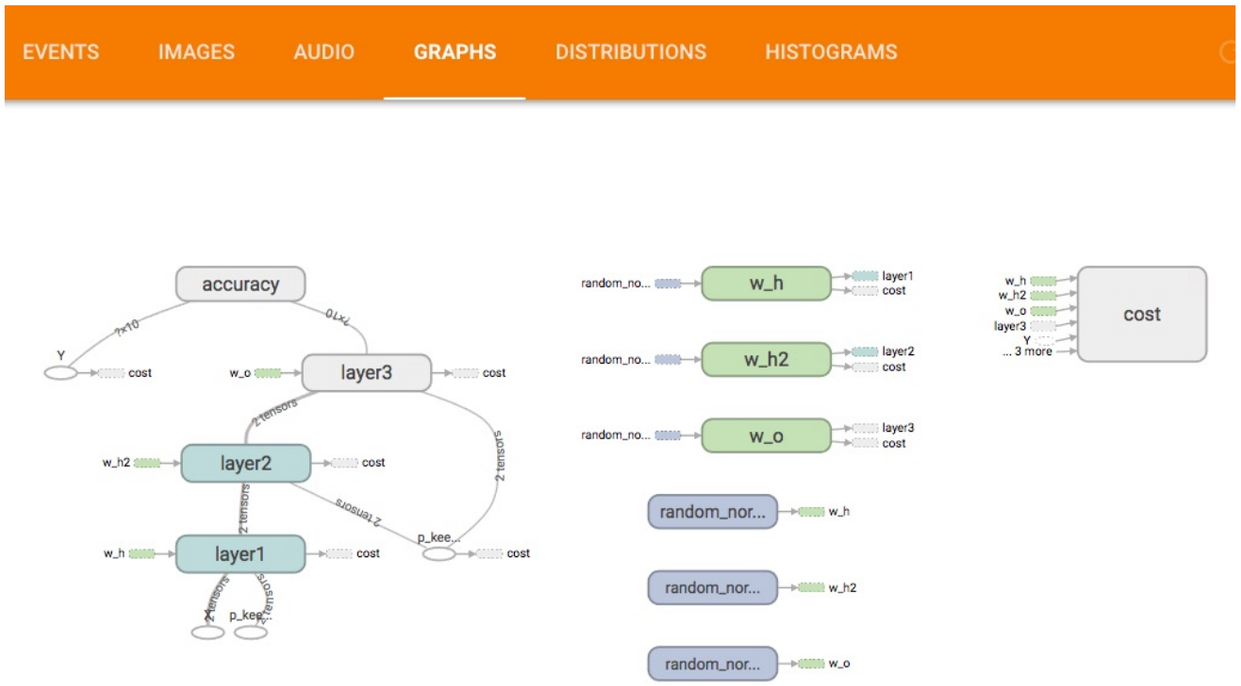
- **import** tensorflow **as** tf

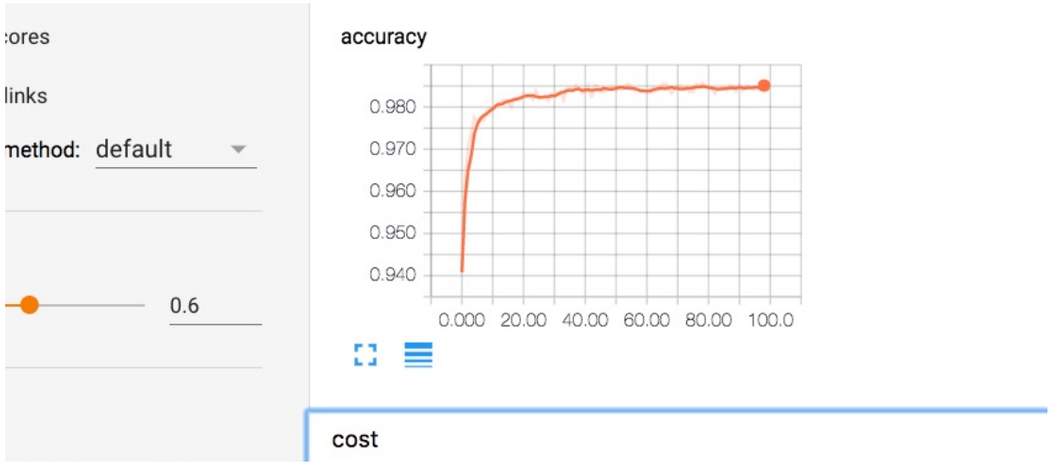
```
with tf.device('/gpu:1'):
    a = tf.placeholder("float") # Create a symbolic variable 'a'
    b = tf.placeholder("float") # Create a symbolic variable 'b'

    y = tf.mul(a, b) # multiply the symbolic variables

session_conf = tf.ConfigProto()
session_conf.gpu_options.allow_growth = True

with tf.Session(config=session_conf) as sess: # create a session to
evaluate the symbolic expressions
    print("%f should equal 2.0" % sess.run(y, feed_dict={a: 1, b: 2})) #
eval expressions with parameters for a and b
    print("%f should equal 9.0" % sess.run(y, feed_dict={a: 3, b: 3}))
```

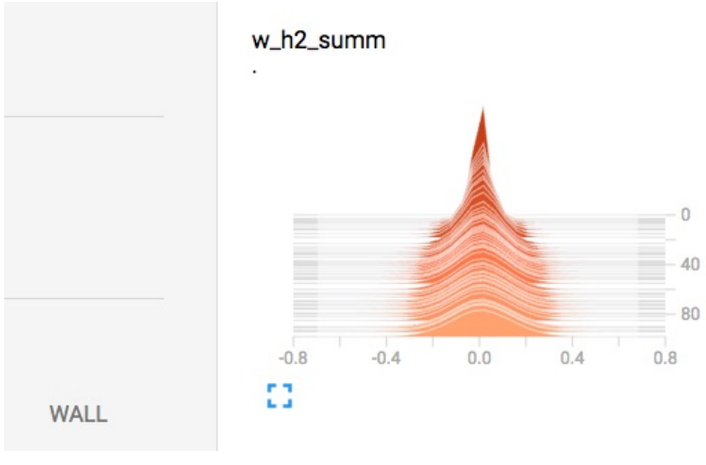
Tensorboard

- tensorboard는 tensorflow를 통해 훈련한 모델 및 파라미터에 대해서 콘솔 환경이 아닌 비주얼 환경에서 모니터링 할 수 있도록 제공합니다.









Tensorboard를 위한 logdir 지정

- ipython 브라우저 목록에서 09_tensorboard.ipynb 파일을 클릭합니다.

- 아래 코드를 보시면 이전에 보신 코드들과 다르게 tensorboard를 위한 name이나 histogram_summary가 설정되어있습니다.

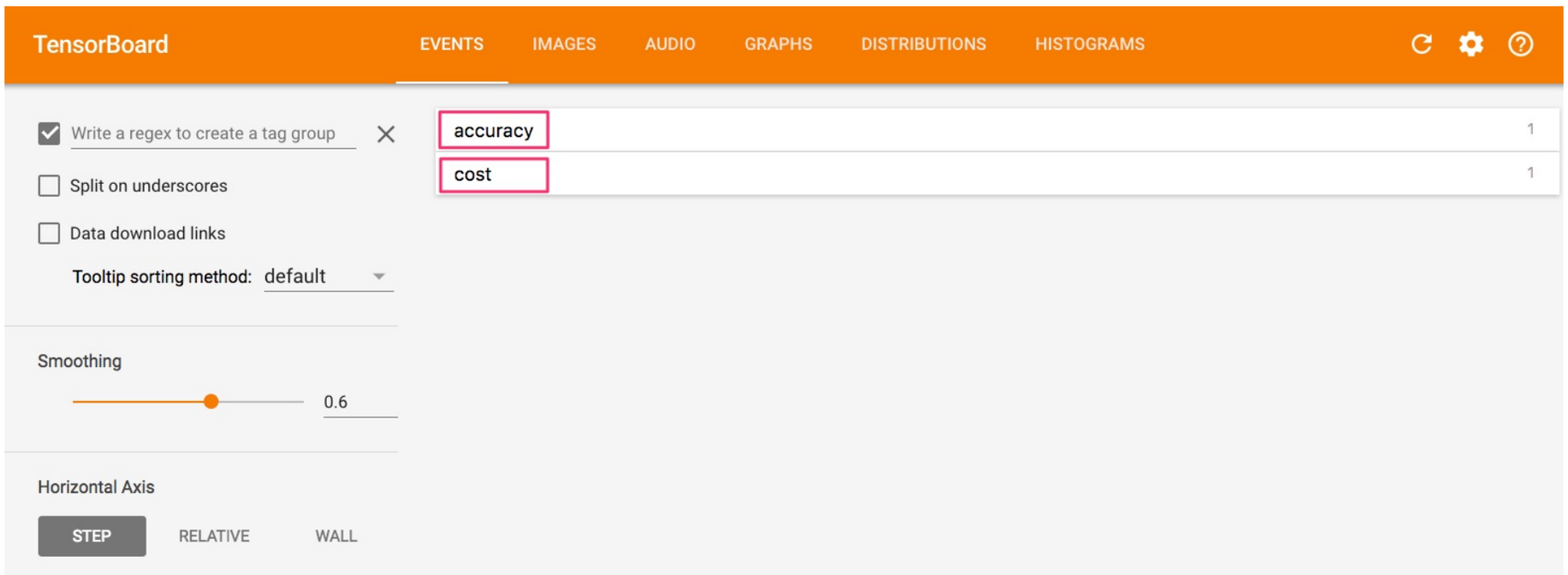
In [3]:
```
X = tf.placeholder("float", [None, 784], name="X")
Y = tf.placeholder("float", [None, 10], name="Y")

w_h = init_weights([784, 625], "w_h")
w_h2 = init_weights([625, 625], "w_h2")
w_o = init_weights([625, 10], "w_o")

# Add histogram summaries for weights
tf.histogram_summary("w_h_summ", w_h)
tf.histogram_summary("w_h2_summ", w_h2)
tf.histogram_summary("w_o_summ", w_o)

p_keep_input = tf.placeholder("float", name="p_keep_input")
p_keep_hidden = tf.placeholder("float", name="p_keep_hidden")
py_x = model(X, w_h, w_h2, w_o, p_keep_input, p_keep_hidden)

with tf.name_scope("cost"):
```

- 아래의 SummaryWriter에 설정된 로그 디렉토리를 잘 확인 하세요.

```
with tf.Session() as sess:
    # create a log writer. run 'tensorboard --logdir=./logs/nn_logs'
    writer = tf.train.SummaryWriter("./logs/nn_logs", sess.graph) # for 0.8
    merged = tf.merge_all_summaries()
```

Tensorboard 실행하기

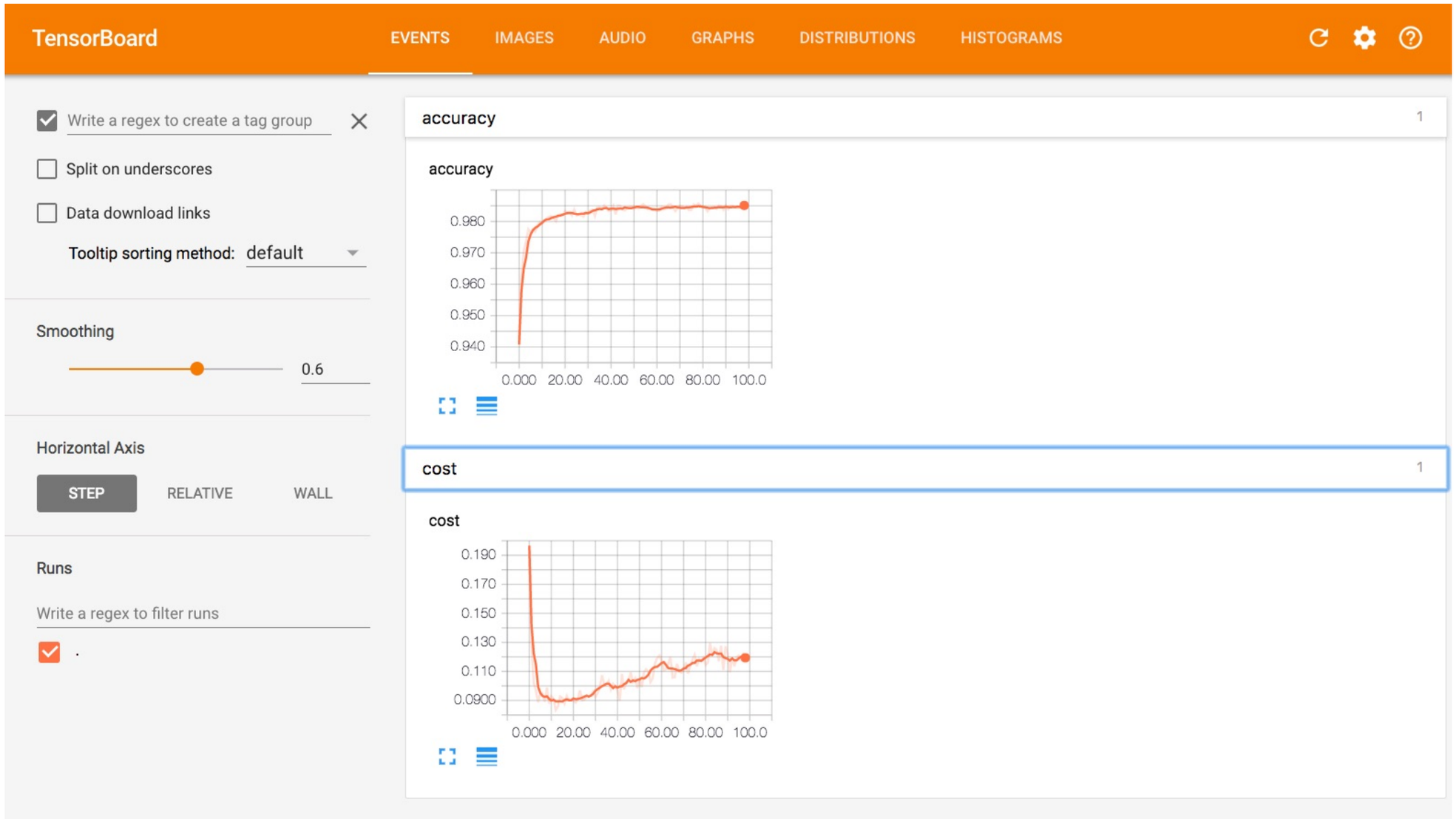
- tensorboard를 실행하기 위해서는 이전에 코드를 실행할 때 로그파일이 저장되도록 지정한 위치와 사용하고자 하는 포트번호를 같이 입력하여 실행합니다. (포트포워딩 참고)
    - tensorboard --logdir logs/nn_logs --port 8901
- 브라우저를 열어 다음을 입력하여 tensorboard에 접속하실 수 있습니다.
    - http://localhost:8901
- 접속이 되면 아래와 같은 화면을 보실 수 있습니다. 각 variable을 클릭해 보세요.



Tensorboard 실행하기

- epoch이 지남에 따라 accuracy와 cost값이 어떻게 변하는지, 수렴하는지 여부 등을 비주얼로 확인하실 수 있습니다.

- 그 외, GRAPH, DISTRIBUTIONS, HISTOGRAMS 등을 다양하게 확인하실 수 있습니다.



감사합니다.

Alluser.net Corp.

Contact : alluser.net@gmail.com