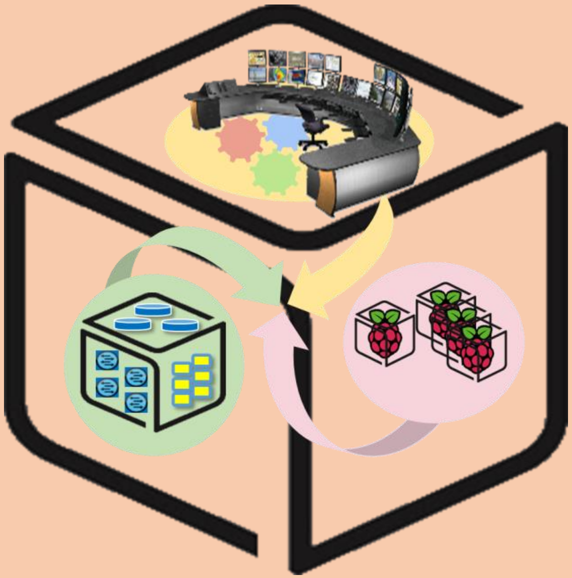
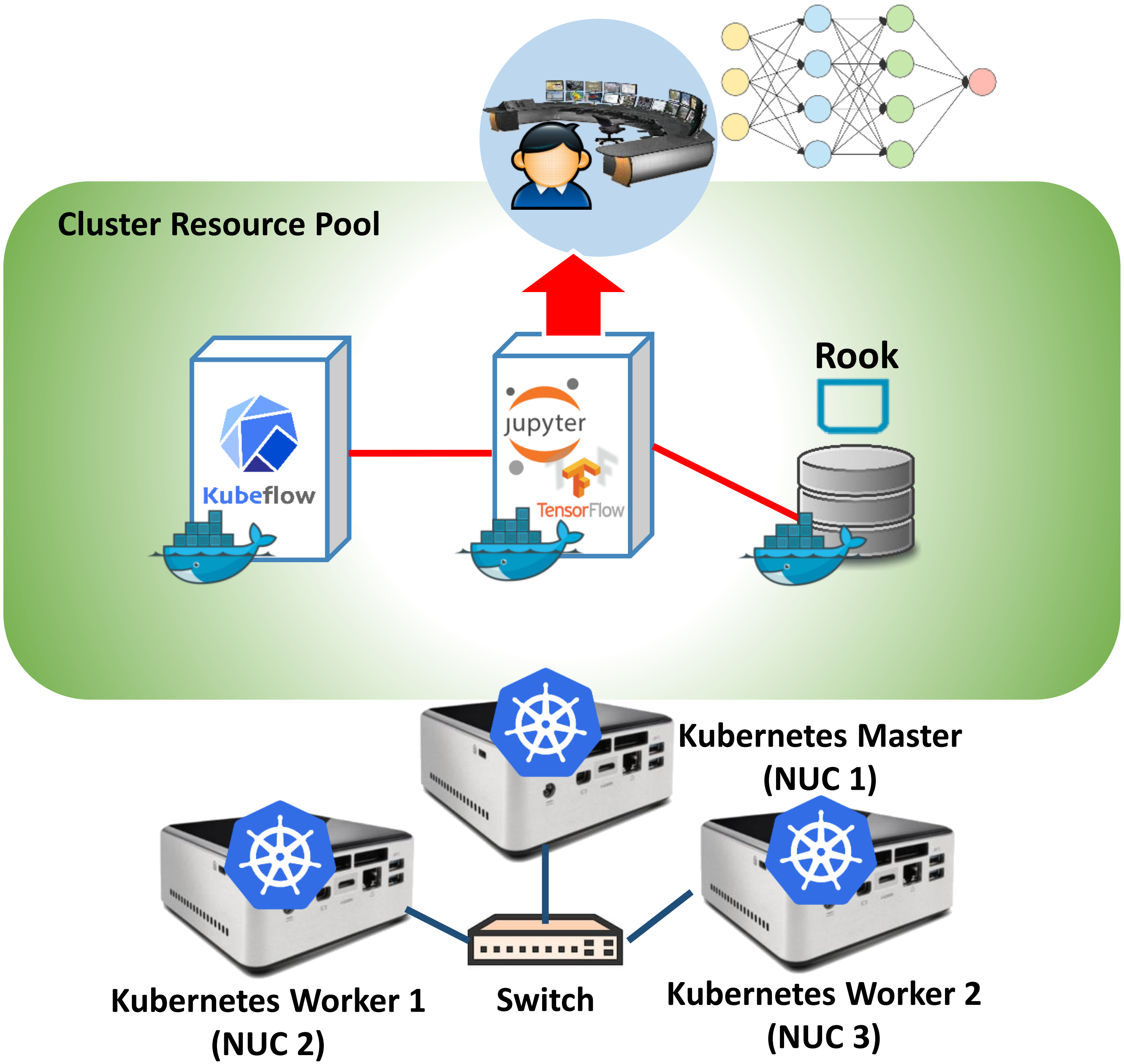
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #6: Analytics

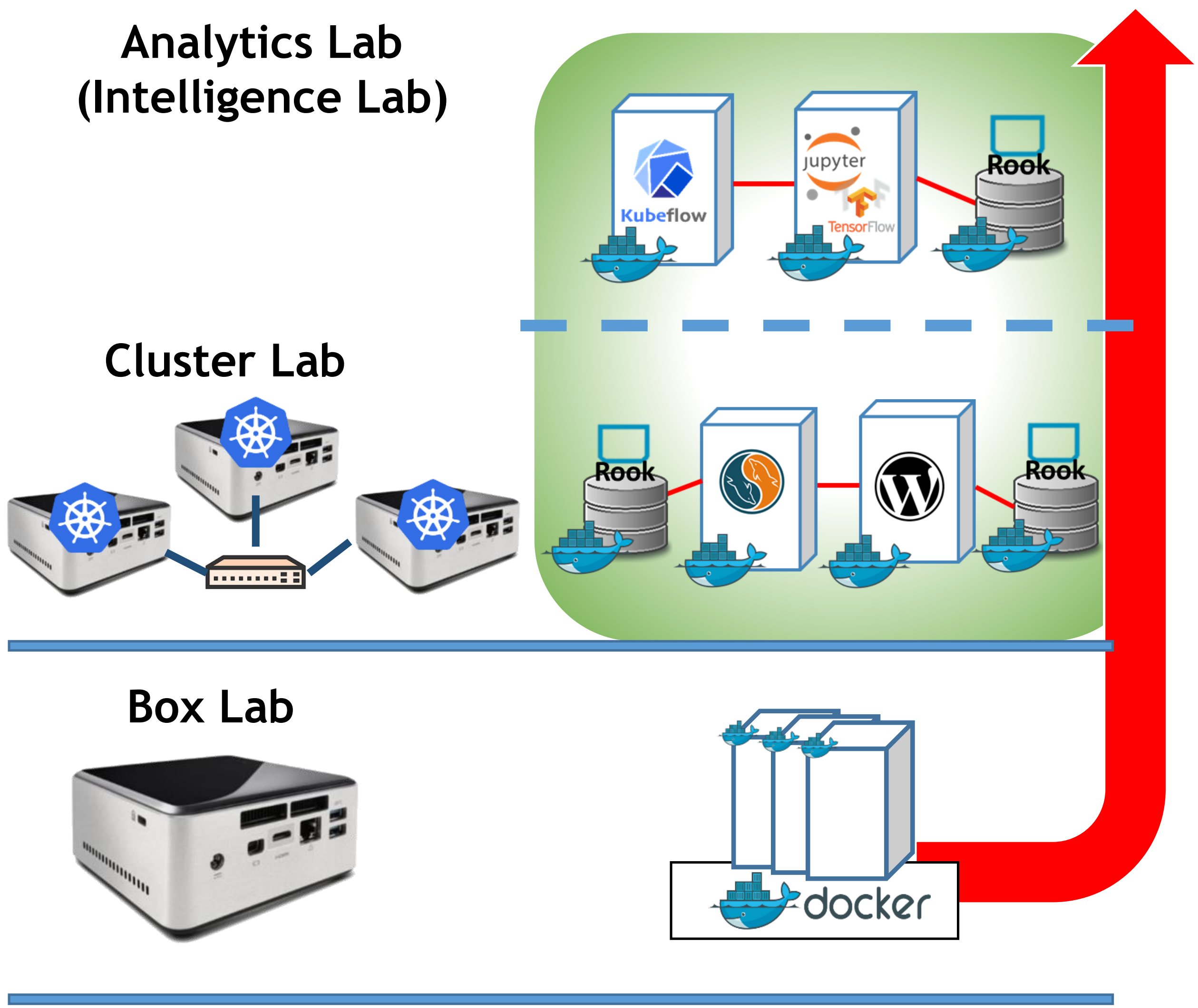


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

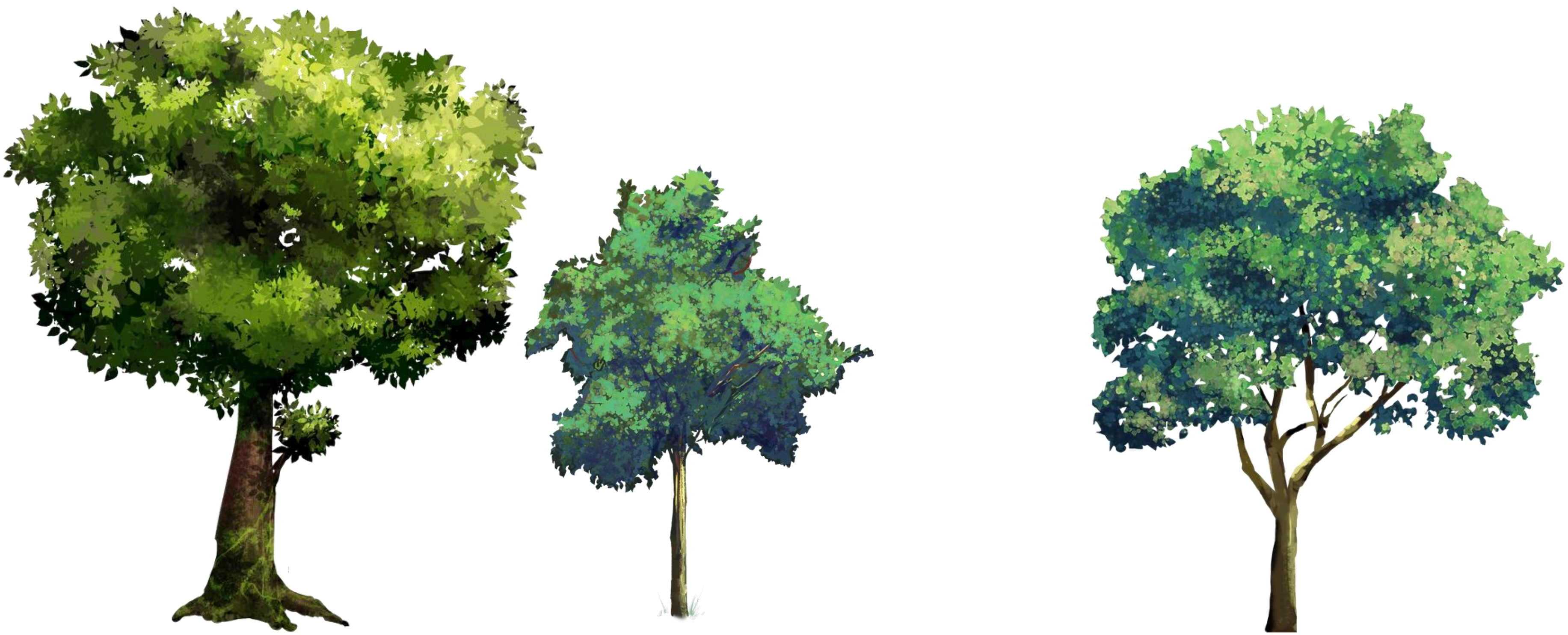
Analytics Lab: Concept



SmartX Labs #1/#5/#6: Relationship

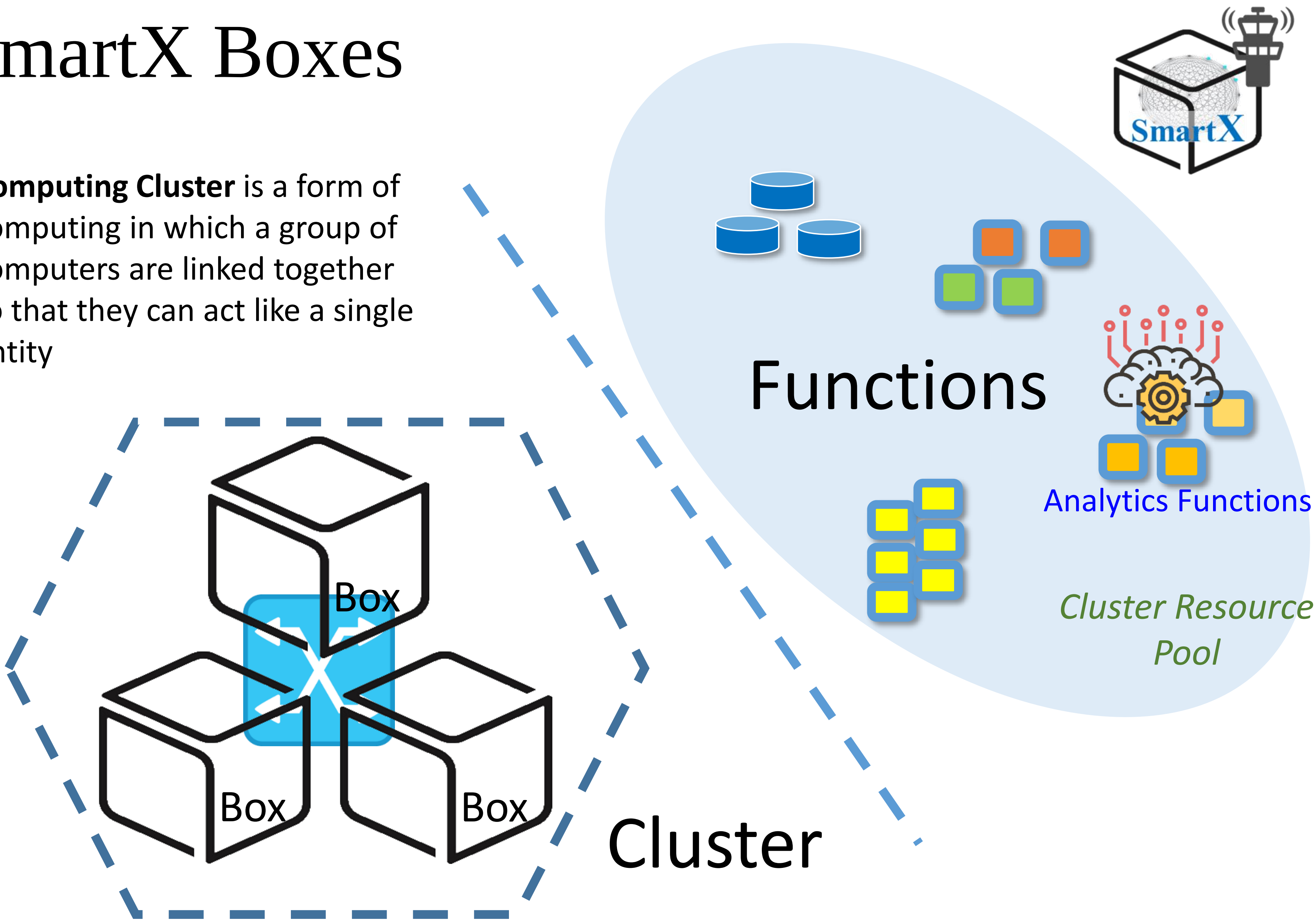


Theory



SmartX Cluster: Inter-connected SmartX Boxes

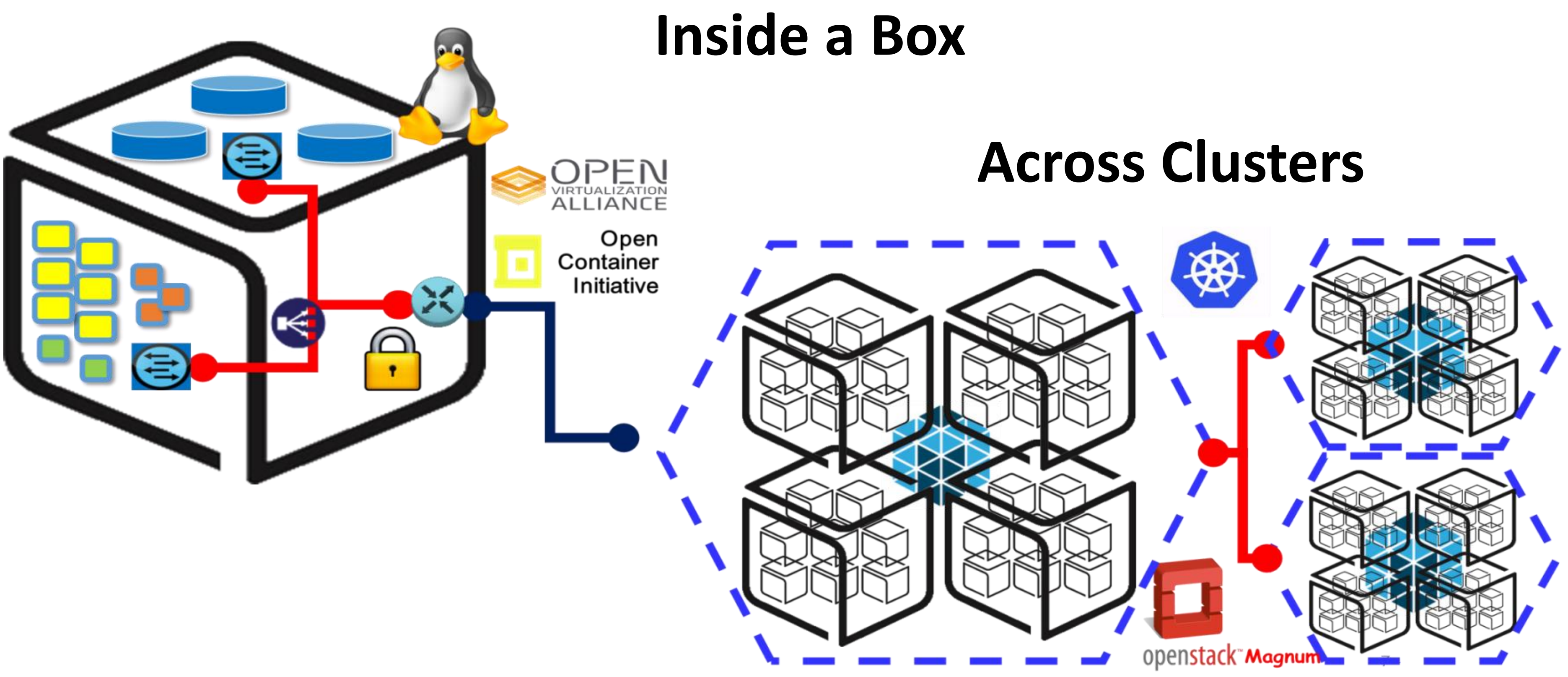
Computing Cluster is a form of computing in which a group of computers are linked together so that they can act like a single entity



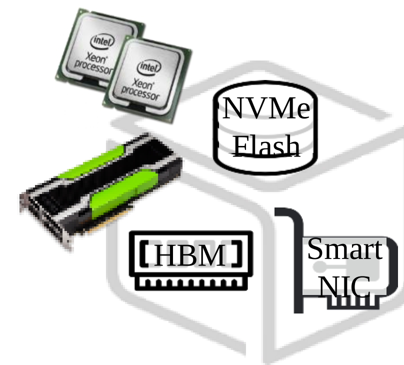
Inter-Connected Functions inside a Box & across Clusters

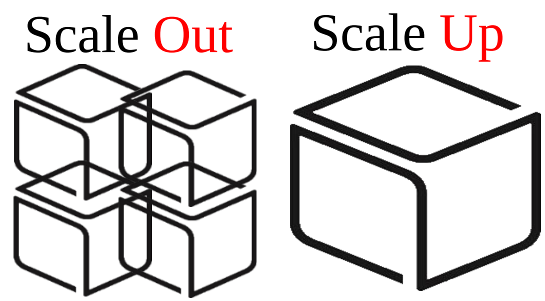
p+v+c Harmonization Challenge:

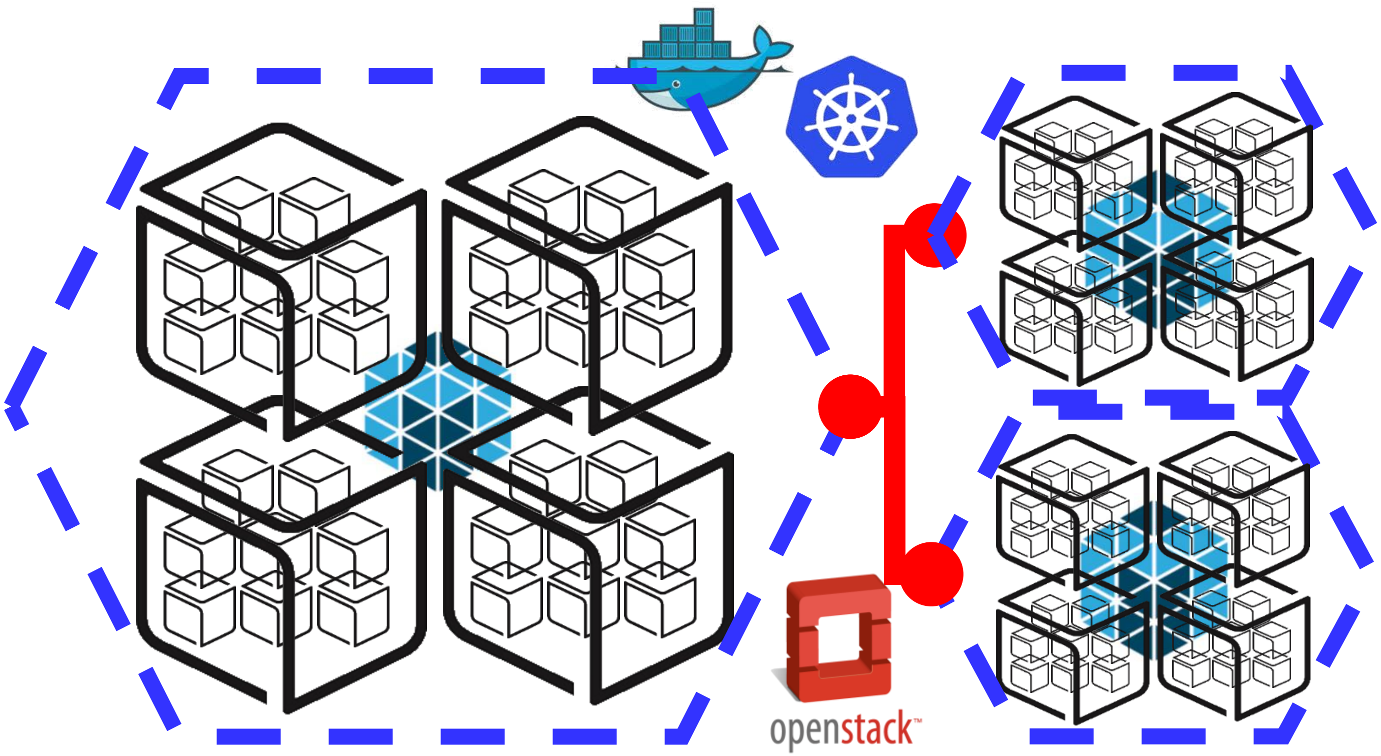
p(Baremetal) + v(VM) + c(Container)



Computer System: Resource Scaling/Pooling with Clustering







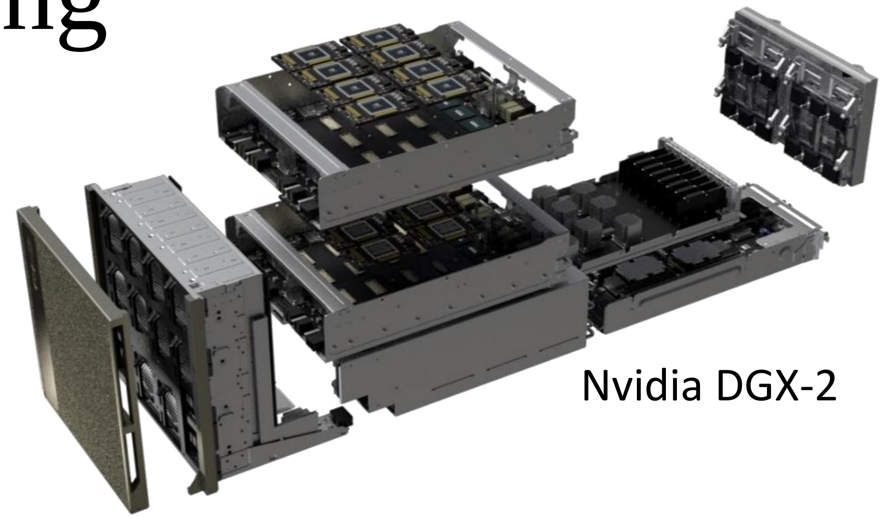
HPC +
HPDA (BigData)

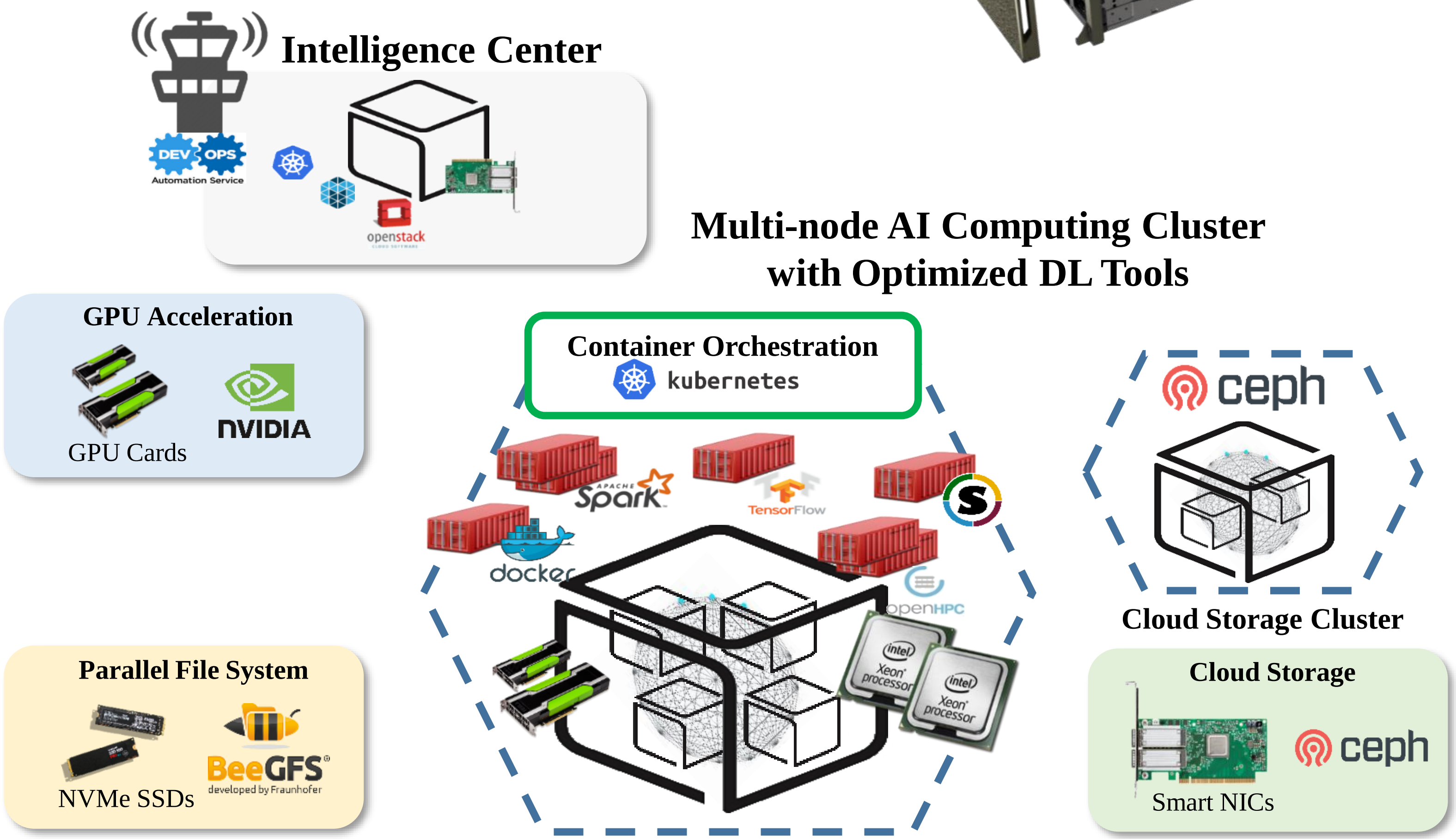
➔ AI (ML/DL)

Cluster for AI Computing

HPC + HPDA (BigData)

→ AI (ML/DL)

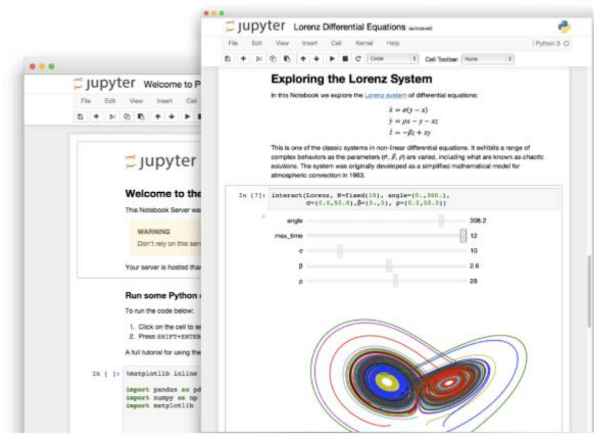




Machine Learning: TensorFlow & Jupyter Notebook

TensorFlow is an **open-source machine learning library** for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud.

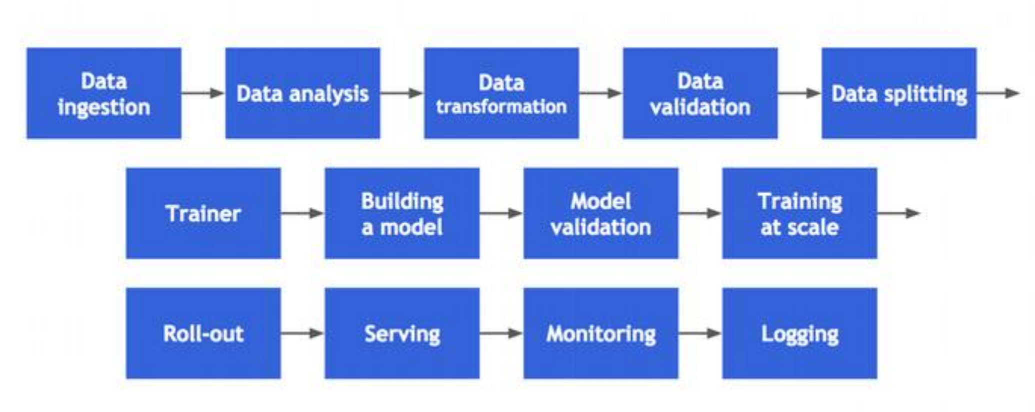
https://github.com/tensorflow/tensorflow



Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, **machine learning**, and much more.

Machine Learning: Kubeflow

Kubeflow



The **Kubeflow** project is dedicated to making deployments of machine learning (ML) workflows on Kubernetes simple, portable and scalable. Our goal is not to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems for ML to diverse infrastructures. Anywhere you are running Kubernetes, you should be able to run Kubeflow.

jupyter

Notebooks

TesorFlow model Training

SELDON

TensorRT

Model serving

AMBASSADOR

argo

Pachyderm

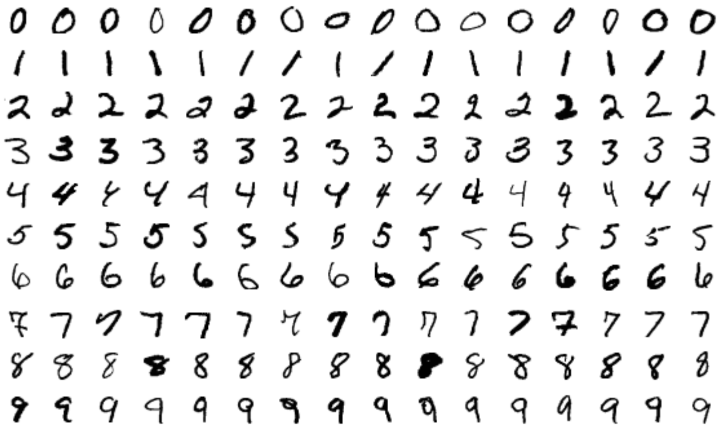
PyTorch

Chainer

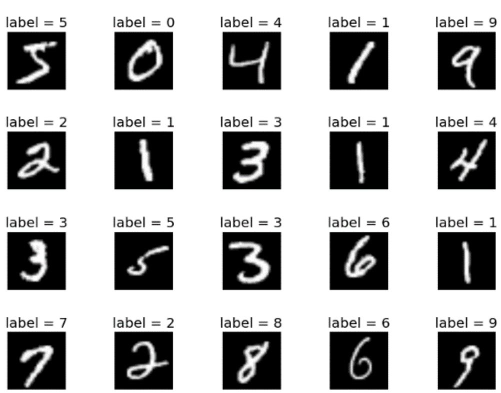
Multi-ML framework

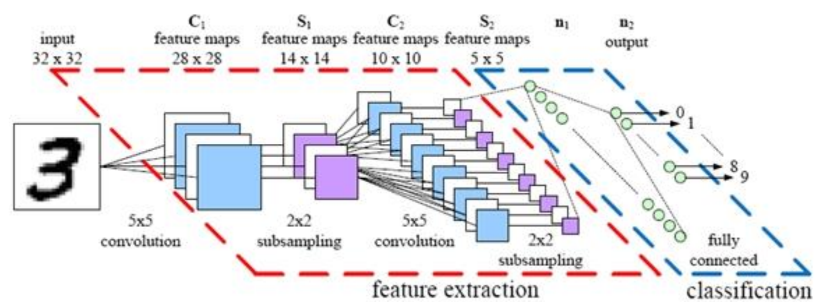
https://www.kubeflow.org/

MNIST handwritten digit Classification



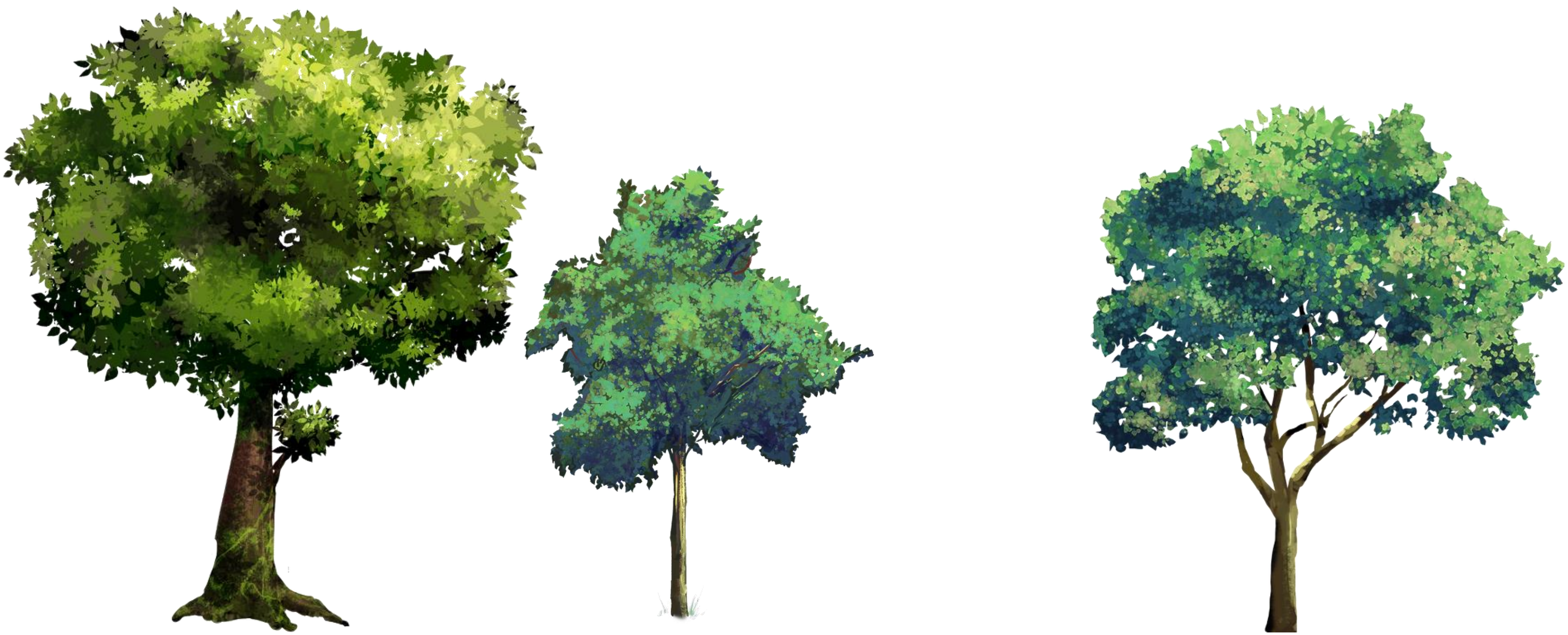
The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.[1][2] The database is also widely used for training and testing in the field of machine learning.



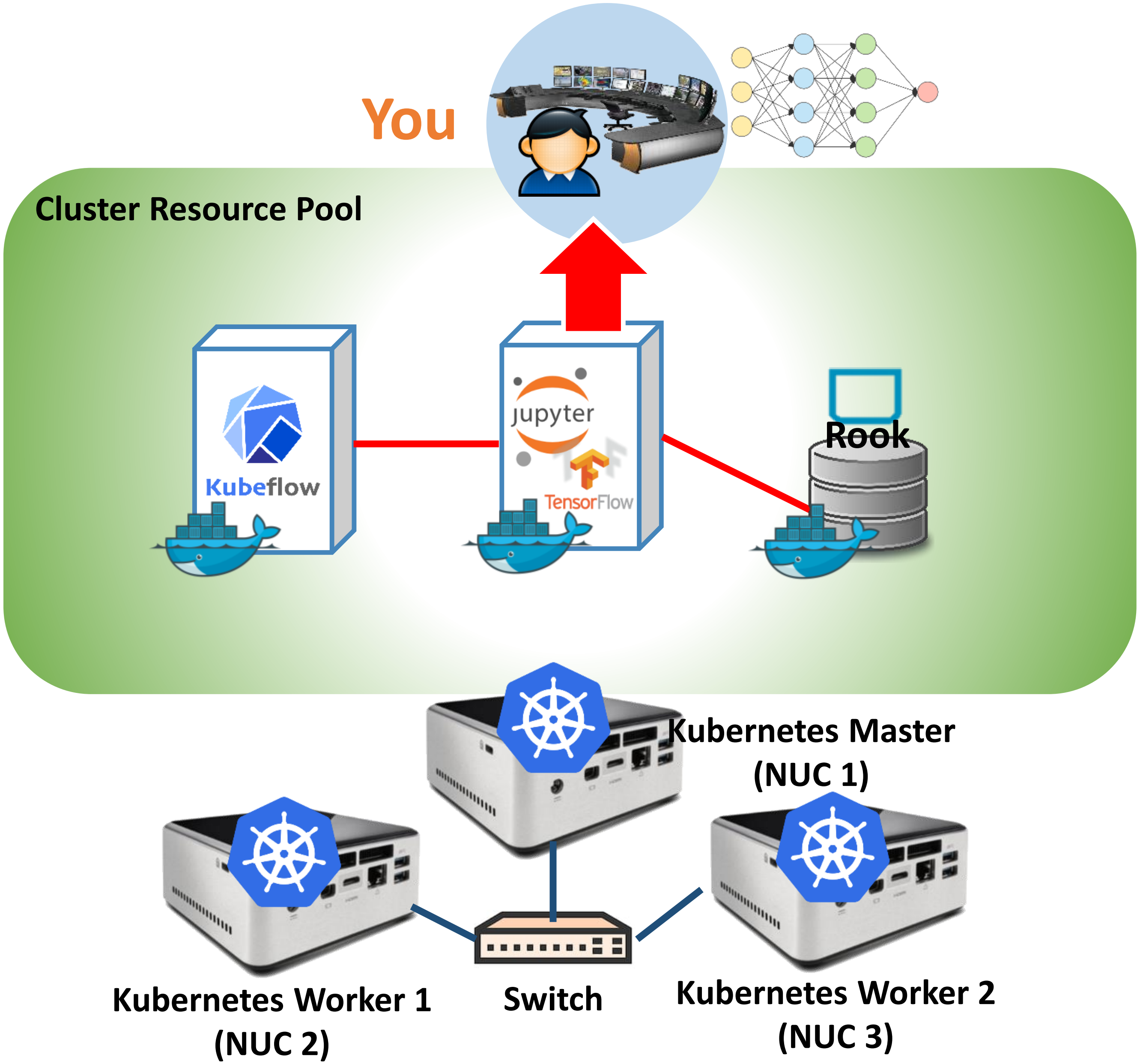


https://towardsdatascience.com

Practice



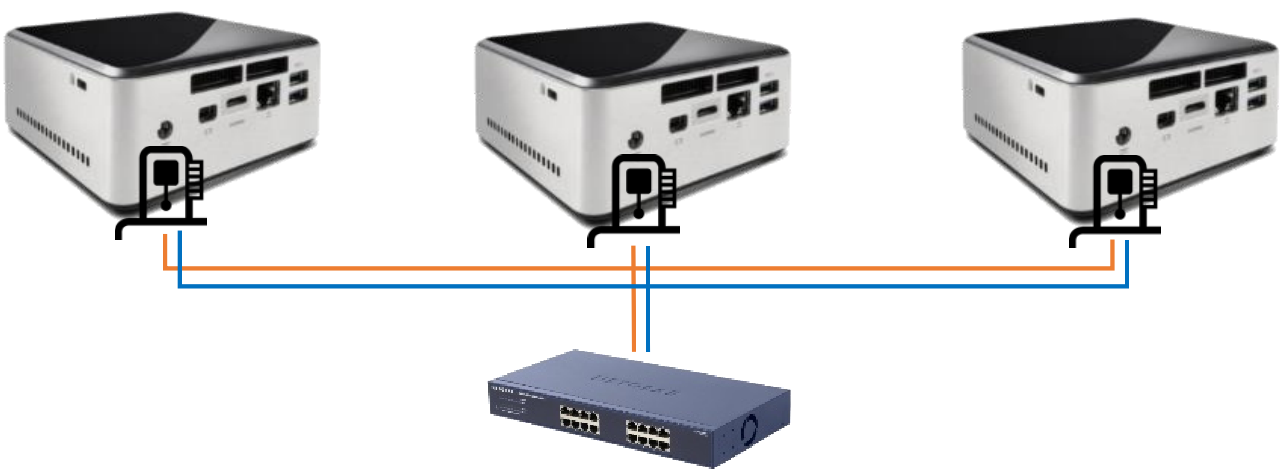
Analytics Lab: Concept



#0 - Lab Preparation (1/2)

Wired connection

NAME: NUC5i5MYHE (NUC PC)
CPU: i5-5300U @2.30GHz
CORE: 4
Memory: 16GB DDR3
HDD: 94GB



NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

#0 - Lab Preparation (2/2)

- Check your cluster is running healthy

For **NUC1**

$ Kubectl get nodes

```
netcs@nuc01:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
nuc01     Ready     master    10d       v1.11.2
nuc02     Ready     <none>    10d       v1.11.2
nuc03     Ready     <none>    10d       v1.11.2
```

Check all nodes are ready.

$ Kubectl get pods –n rook-ceph

```
netcs@nuc01:~$ kubectl get pods -n rook-ceph
NAME                                  READY     STATUS      RESTARTS
rook-ceph-mgr-a-9c44495df-lfs4m       1/1       Running     0
rook-ceph-mon0-5j655                  1/1       Running     0
rook-ceph-mon1-rkggs                  1/1       Running     0
rook-ceph-mon2-vvp2n                  1/1       Running     0
rook-ceph-osd-id-0-8694878c4b-9zz5l   1/1       Running     0
rook-ceph-osd-id-1-756995f97b-9hdhk   1/1       Running     0
rook-ceph-osd-prepare-nuc02-26nj6     0/1       Completed   0
rook-ceph-osd-prepare-nuc03-cf49p     0/1       Completed   0
```

Check Rook are running healthy on your cluster.

#1-1 Cluster Preparations for ML: Kubeflow Installation (1/3)

For **NUC1**

- **Install prerequisites for kubeflow:** Ksonnet Installation

$ sudo su
$ wget https://dl.google.com/go/go1.11.2.linux-amd64.tar.gz
$ tar -C /usr/local -xzf go1.11.2.linux-amd64.tar.gz
$ export PATH=$PATH:/usr/local/go/bin
$ GOPATH=/root/go/
$ go get github.com/ksonnet/ksonnet
$ cd $GOPATH/src/github.com/ksonnet/ksonnet
$ make install
$ exit → Come out of root user

- **Set Rook Storageclass to default for kubeflow**

$ kubectl patch storageclass rook-ceph-block -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
$ kubectl get storageclasses

#1-1 Cluster Preparations for ML: Kubeflow Installation (2/3)

For **NUC1**

- **Install Kubeflow**

$ mkdir ~/kubeflow
$ cd ~/kubeflow
$ export KUBEFLOW_TAG=v0.3.1
$ curl https://raw.githubusercontent.com/kubeflow/kubeflow/${KUBEFLOW_TAG}/scripts/download.sh | bash
$ ~/kubeflow/scripts/kfctl.sh init kubeflow_app --platform none
$ cd kubeflow_app
$ ~/kubeflow/scripts/kfctl.sh generate k8s
$ cd ~/kubeflow/kubeflow_app/ks_app
$ ks param set jupyterhub serviceType NodePort
$ cd ~/kubeflow/kubeflow_apps
$ ~/kubeflow/scripts/kfctl.sh apply k8s

#1-1 Cluster Preparations for ML: Kubeflow Installation (3/3)

For **NUC1**

- **Check Kubeflow is running healthy**

$ kubectl get pods -n kubeflow

- **Check the exposed port to access Jupyter hub**

$ kubectl get services –n kubeflow

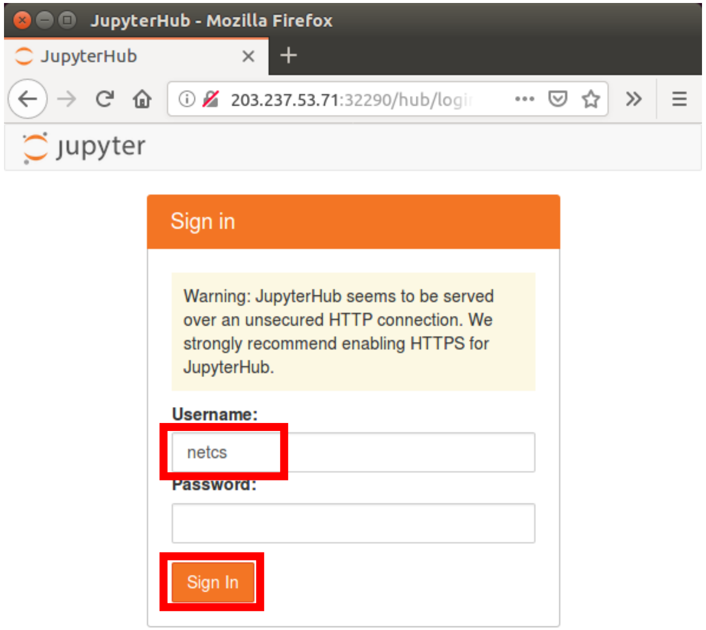
```
netcs@nuc01:~$ kubectl get services -n kubeflow
NAME                                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
ambassador                               ClusterIP   10.111.165.80    <none>        80/TCP
ambassador-admin                         ClusterIP   10.101.217.43    <none>        8877/TCP
argo-ui                                  NodePort    10.101.8.20      <none>        80:30681/TCP
centraldashboard                         ClusterIP   10.105.124.82    <none>        80/TCP
k8s-dashboard                            ClusterIP   10.110.111.206   <none>        443/TCP
modeldb-backend                          ClusterIP   10.104.50.63     <none>        6543/TCP
modeldb-db                               ClusterIP   10.103.246.44    <none>        27017/TCP
modeldb-frontend                         ClusterIP   10.102.138.220   <none>        3000/TCP
statsd-sink                              ClusterIP   10.106.86.124    <none>        9102/TCP
tf-hub-0                                 ClusterIP   None             <none>        8000/TCP
tf-hub-lb                                NodePort    10.107.131.150   <none>        80:32290/TCP
tf-job-dashboard                         ClusterIP   10.111.220.254   <none>        80/TCP
vizier-core                              NodePort    10.96.183.97     <none>        6789:30678/TCP
vizier-db                                ClusterIP   10.99.58.20      <none>        3306/TCP
vizier-suggestion-bayesianoptimization   ClusterIP   10.98.249.26     <none>        6789/TCP
vizier-suggestion-grid                   ClusterIP   10.100.145.210   <none>        6789/TCP
vizier-suggestion-hyperband              ClusterIP   10.108.171.180   <none>        6789/TCP
vizier-suggestion-random                 ClusterIP   10.97.121.224    <none>        6789/TCP
```

You can access Jupyter hub at this address
http://nuc01_IP:Exposed_port

#1-2 Deploy a ML Container: Create a Jupyter Notebook (1/3)

For **NUC1**

Open a web browser and enter the Jupyter hub address
http://nuc01_IP:Exposed_port



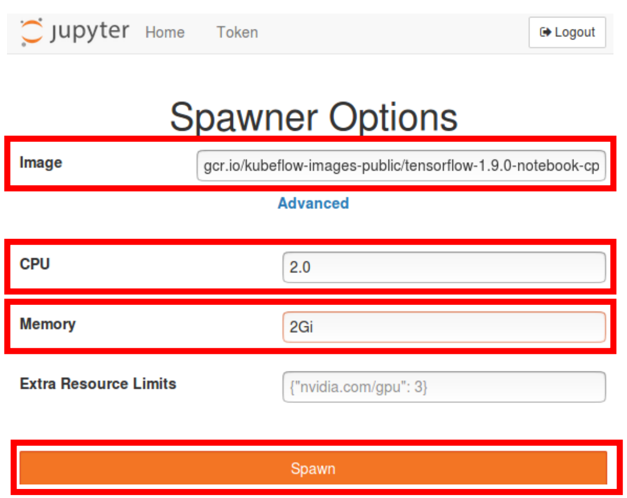
Enter your username and click ‘Sign In’ button (you don’t have to enter a password)

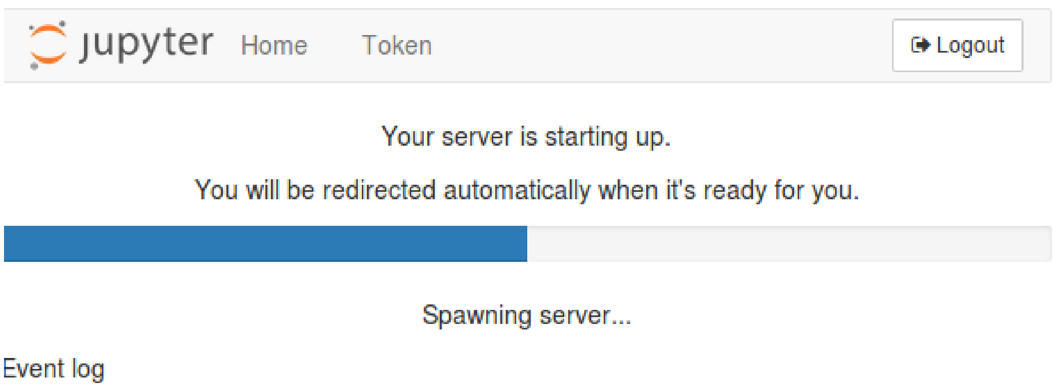
#1-2 Deploy a ML Container: Create a Jupyter Notebook (2/3)

For **NUC1**

In Spawner page, you can choose container image and the size of resources to create a Jupyter Notebook container for Machine Learning

Select or enter the options as below
Image: gcr.io/kubeflow-images-public/tensorflow-1.9.0-notebook-cpu:v0.3.1
CPU: 2
Memory: 2Gi





Click Spawn Button and you need to wait for a while

#1-2 Deploy a ML Container: Create a Jupyter Notebook (3/3)

For **NUC1**

You can see your Jupyter notebook container is deployed as a pod on cluster

$ kubectl get pods –n kubeflow

```
netcs@nuc01:~$ kubectl get pods -n kubeflow
NAME                                  READY   STATUS    RESTARTS
ambassador-c97f7b448-998gd            3/3     Running   0
ambassador-c97f7b448-c2j2b            3/3     Running   0
ambassador-c97f7b448-dvlkv            3/3     Running   0
argo-ui-7495b79b59-l5xh6              1/1     Running   0
centraldashboard-798f8d68d5-5wshd     1/1     Running   0
jupyter-netcs                         1/1     Running   0
jupyter-test                          1/1     Running   0
modeldb-backend-d69695b66-99cqm       1/1     Running   0
modeldb-db-975db58f7-prpzf            1/1     Running   0
```

Now your Jupyter notebook is created…

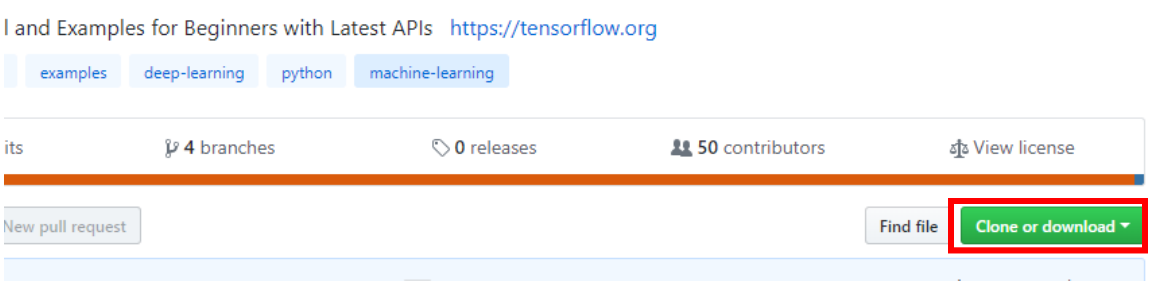
#2-1 Running Analytics code: Running a Sample ML Code (1/3)

For **NUC01**

Download sample notebook including MNIST Machine Learning Code

```
$ git clone https://github.com/aymericdamien/TensorFlow-Examples/
```

Or Download from web browser as below. (you need to unzip the file)



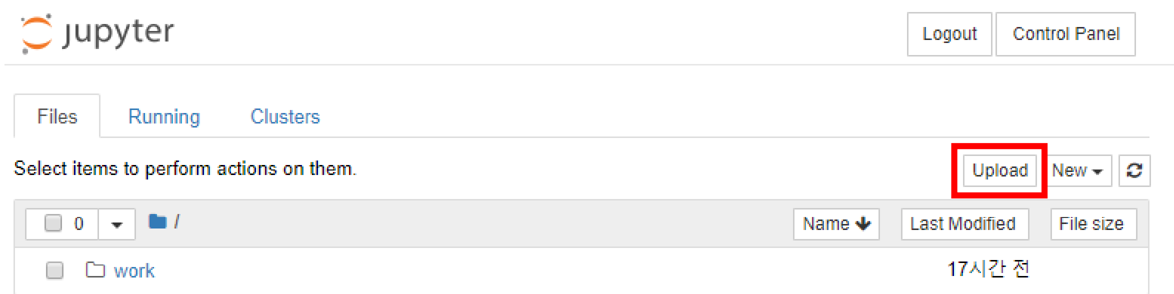
TensorFlow-Examples-master\notebooks\3_NeuralNetwork\convolutional_network.ipynb

We will upload the sample notebook on your Jupyter and run it.
The notebook include MNIST machine learning example code.

https://github.com/aymericdamien/TensorFlow-Examples/

#2-1 Running Analytics code: Running a Sample ML Code (2/3)

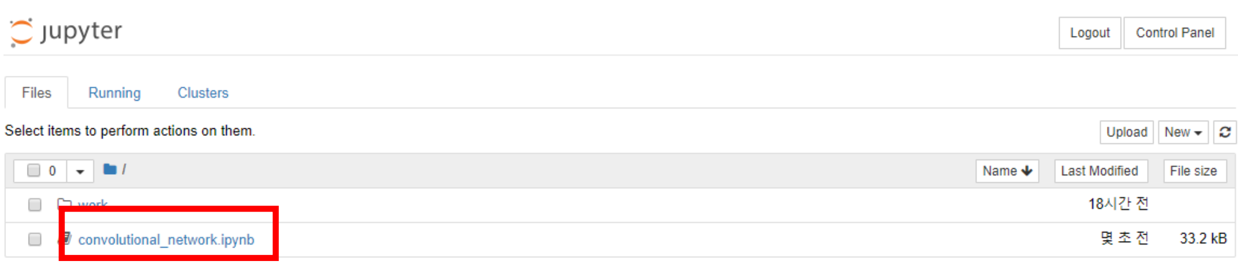
For **NUC1**



Remember! we will use this file

TensorFlow-Examples-master\notebooks\3_NeuralNetwork\convolutional_network.ipynb

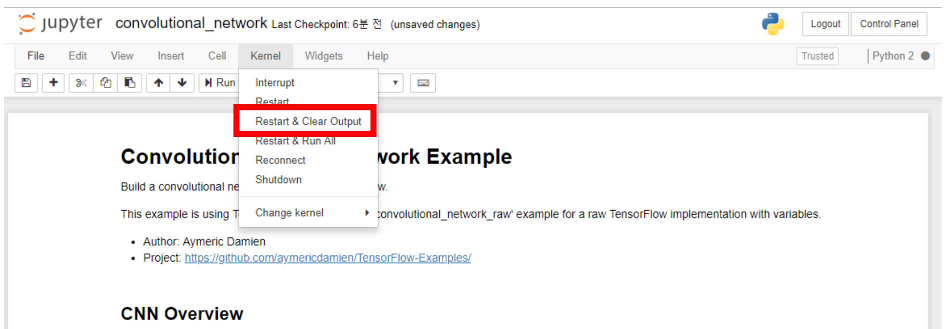
Press upload button to upload the sample notebook



Click it to open the notebook

#2-1 Running Analytics code: Running a Sample ML Code (3/3)

For **NUC1**



Now, you will run the MNIST example code in sample notebook.
Click kernel→"Restart&Clear Output" button

```
In [*]: # Define the input function for training
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': mnist.train.images}, y=mnist.train.labels,
    batch_size=batch_size, num_epochs=None, shuffle=True)
# Train the Model
model.train(input_fn, steps=num_steps)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into /tmp/tmpp9DVpG/model.ckpt.
INFO:tensorflow:loss = 2.3231235, step = 1
INFO:tensorflow:global_step/sec: 7.88675
INFO:tensorflow:loss = 0.09013237, step = 101 (12.681 sec)
INFO:tensorflow:global_step/sec: 7.96045
INFO:tensorflow:loss = 0.087195896, step = 201 (12.562 sec)
INFO:tensorflow:global_step/sec: 7.93652
INFO:tensorflow:loss = 0.07184037, step = 301 (12.600 sec)
INFO:tensorflow:global_step/sec: 7.92234
INFO:tensorflow:loss = 0.15338697, step = 401 (12.622 sec)
⋮
```

The training takes a few minutes.

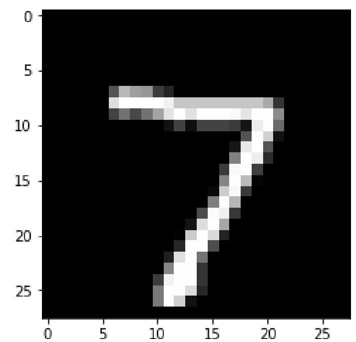
#2-2 Running Analytics code: Check ML Training results

For **NUC1**

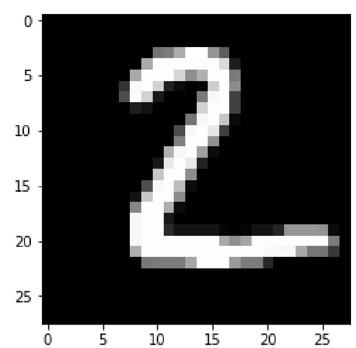
```
In [7]: # Evaluate the Model
        # Define the input function for evaluating
        input_fn = tf.estimator.inputs.numpy_input_fn(
            x={'images': mnist.test.images}, y=mnist.test.labels,
            batch_size=batch_size, shuffle=False)
        # Use the Estimator 'evaluate' method
        model.evaluate(input_fn)

        INFO:tensorflow:Calling model_fn.
        INFO:tensorflow:Done calling model_fn.
        INFO:tensorflow:Starting evaluation at 2018-11-25-07:45:59
        INFO:tensorflow:Graph was finalized.
        INFO:tensorflow:Restoring parameters from /tmp/tmpp9DVpG/model.ckpt-2000
        INFO:tensorflow:Running local_init_op.
        INFO:tensorflow:Done running local_init_op.
        INFO:tensorflow:Finished evaluation at 2018-11-25-07:46:03
        INFO:tensorflow:Saving dict for global step 2000: accuracy = 0.9892, global_step = 20
        00, loss = 0.035650674
        INFO:tensorflow:Saving 'checkpoint_path' summary for global step 2000: /tmp/tmpp9DVp
        G/model.ckpt-2000

Out[7]: {'accuracy': 0.9892, 'global_step': 2000, 'loss': 0.035650674}
```

Check training results

Your model has 98.92% accuracy!



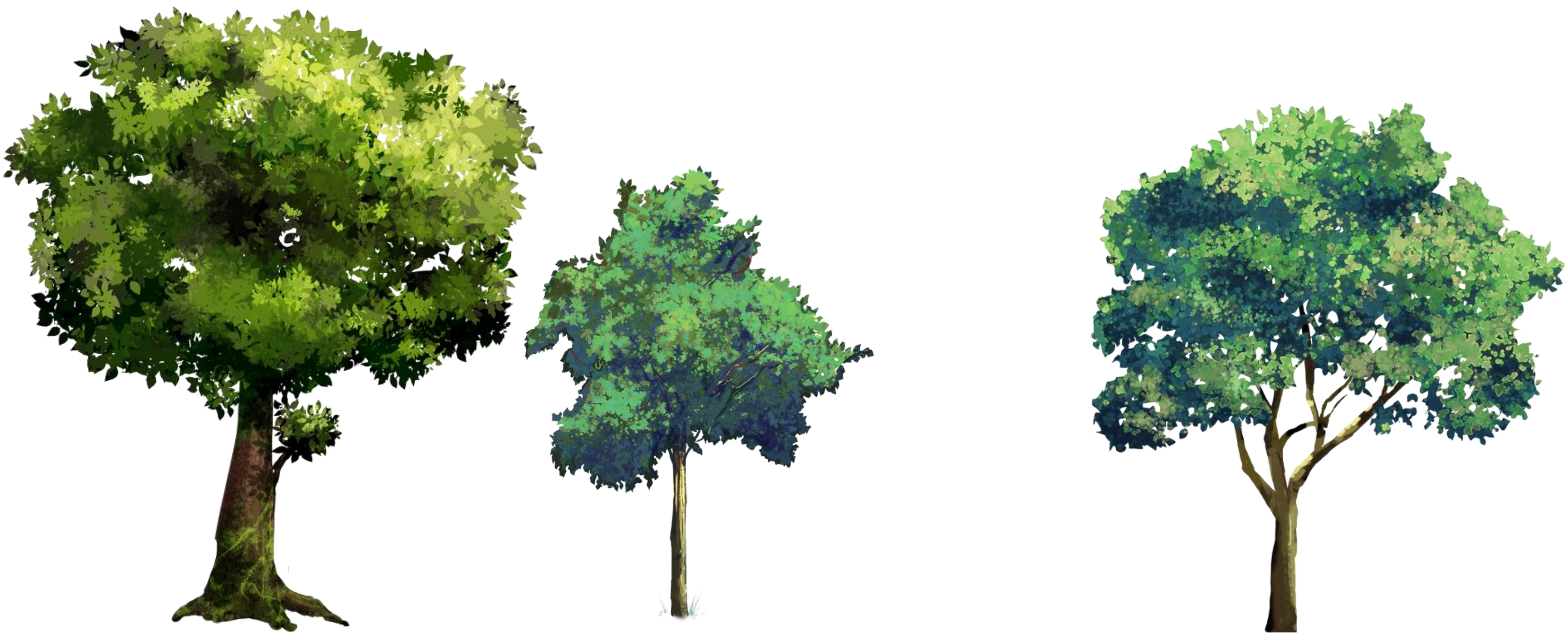
Model prediction: 7



Model prediction: 2

Your Machine Learning model correctly identified the number in the images!

Review



Lab Summary

With Analytics (Intelligence) Lab, you have experimented

1. How to create ML/DL environment on a container-orchestrated cluster? (Kubeflow, …)
2. How to operate desired ML training by testing selected ML code (i.e., neural networks) over the prepared training data?
3. Do you understand the overall workflow for running ML/DL?

Thank You for Your Attention
Any Questions?

mini@smartx.kr