Submit Training Jobs to Azure ML Compute Cluster

Load Libraries

In [14]:
```
# Libraries are only necessary for what you see in this notebook
# The Azure ML Environment in the configuration later defines libraries needed for training
import azureml.core
from azureml.core import Workspace, Experiment, Environment
from azureml.core.conda_dependencies import CondaDependencies
from azureml.core.compute import ComputeTarget, AmlCompute
from azureml.core import ScriptRunConfig
from azureml.core.model import Model
import os
```

```
ml-workspace    northcentralus  rg-c1-bprescott-lab-02  northcentralus
```

Define Training Functions

In [38]:
```
# Loads the current Azure ML Workspace configurations
def load_workspace():
    ws = Workspace.from_config()
    print(ws.name, ws.location, ws.resource_group, ws.location, sep='\t')

# Sets the model's architecture parameters
def model_params(name, filters, densenodes):
    name = name
    filters = filters
    densenodes = densenodes
    modelname = "{}-{}-{}".format(name,filters,densenodes)
    return filters, densenodes, modelname

# Sets the computer cluster's training environment configuration
def environment_params(exp_name, cluster_name, env_name, script_directory, script, filters, densenodes, modelname):
    exp = Experiment(workspace=ws, name=exp_name)
    gpu_cluster = ComputeTarget(workspace=ws, name=cluster_name)
    env = Environment(env_name)
    cd = CondaDependencies.create(
                                    pip_packages=['azureml-dataset-runtime[pandas,fuse]',
                                                  'azureml-defaults',
                                                  'packaging',
                                                  'tensorflow',
                                                  'matplotlib',
                                                  'numpy',
                                                  'pandas',
                                                  'seaborn',
                                                  'scikit-learn',
                                                  'argparse',
                                                  'azureml-core'],

                                    conda_packages=['scikit-learn==0.22.1']
                                )
    env.python.conda_dependencies = cd
    env.register(workspace=ws)
    config = ScriptRunConfig(
                                source_directory=script_directory,
                                script=script,
                                arguments = ['--filters',filters,
                                             '--densenodes',densenodes,
                                             '--modelname',modelname],
                                compute_target=gpu_cluster.name,
                                environment=env
                            )
    return config

# Submits model configuration to computer cluster for training. Monitors run.
def train_model(configvariable):
    config = configvariable
    run = exp.submit(c)
    print(run.get_portal_url())
    run.wait_for_completion(show_output=True)


# Registers the trained model with Azure ML Models repo
def model_register(modelname):
    run.register_model(model_name=modelname,
                       model_path='outputs/{}.h5'.format(modelname),
                       model_framework=Model.Framework.TENSORFLOW,
                       model_framework_version='2.0')
```

Define Parameters and Submit Training Run

```
In [ ]: # Load the Azure ML Workspace settings
        load_workspace()

        # Prompt for model architecture hyperparameters
        fcount, dcount = int(input('How many convolutional filters?')), int(input("How many dense layer nodes?"))
        exp_name, cluster_name, env_name, script_directory, script = (input("Enter an Experiment name:"),
                                                                      input("Enter the Computer Cluster's name:"),
                                                                      input("Enter the Azure ML Environment's name:"),
                                                                      input("Enter the Computer Instance's script directory:"),
                                                                      input("Enter the training script name with .py extension:"))


        # Set the convolutional filter count, fully connected (dense) layer node count, and the name you want for the model
        filters, densenodes, modelname = model_params(name = 'testme', filters = fcount, densenodes = dcount)

        # Set a config variables for the Azure ML training run
        config = environment_params(exp_name = exp_name,            # Azure ML Experiment name to log metrics into
                                    cluster_name = cluster_name,          # Azure ML Compute Cluster's name
                                    env_name = env_name,                  # Which Azure ML Environment set to use during training
                                    script_directory = script_directory,  # Directory in Azure ML Compute Instance where training script
                                    script = script,                      # Main training script with convolutional network architecture
                                    filters = filters,                    # Convolutional filter count from model_params
                                    densenodes = densenodes,              # Dense (FC) node count from model_params
                                    modelname = modelname )               # Future model's name after training from model_params

        # Submit the training run configuration to the compute cluster for training
        train_model(config)

        # # Register the trained model into Azure ML Models repository
        model_register(modelname)
```