

Tech Summit 2018

Hands-on lab

使用 Azure ML 以及 Batch AI 分布式模型训练

总览	3
Part 1: 配置 AzureML Notebook 所需环境	5
Task 0 – 在 Azure Portal 中找到实验所需的 Azure 订阅	5
Task 1 – 下载本实验相关 AzureML 样本, 启动 Notebook Server	5
Task 2 – 在本地创建 Azure ML Notebook 运行所需的配置文件	5
Task 3 – 通过 Azure Portal 查找 AzureML 相关资源.....	7
Part 2: 通过 AzureML 使用 BatchAI 群集进行分布式训练	12
Task 1 – 进入 Notebook 文件并初始化 Workspace	12
Task 2 – 创建 BatchAI 虚拟机群集作为运算对象	12
Task 3 – 创建本地源目录并放入训练脚本	13
Task 4 – 定义实验及分布式 Tensorflow 作业	14
Task 5 – 提交并监督训练作业	15

总览

微软 **Azure Machine Learning** 服务是一款可用于开发和部署机器学习模型的云服务。使用 **Azure** 机器学习服务，你可以在生成、培训、部署和管理模型时对其进行跟踪，所有这些都以云提供的广泛规模完成。**Azure** 机器学习服务提供了一个基于云的环境，你可以使用这一环境来开发、培训、测试、部署、管理和跟踪机器学习模型。

Azure 机器学习服务完全支持开放源代码技术，因此，你可以使用数以万计的开放源代码 Python 包与机器学习组件（如 **TensorFlow** 和 **scikit-learn**）。借助丰富的工具（如 **Jupyter Notebook** 或 **Visual Studio Code Tools for AI**），可以交互式地轻松探索数据、转换数据，然后开发和测试模型。此外，**Azure** 机器学习服务还包括自动化模型生成和优化的功能，能够帮助你轻松、高效和准确地创建模型。

使用 **Azure** 机器学习服务，你可以先在本地计算机上开始培训，然后扩大到云。通过提供对 **Azure Batch AI** 的本机支持和高级超参数优化服务，你可以使用云的强大功能更快地生成更好的模型。

实验目的

本次实验覆盖以下内容:

- 配置 AzureML Notebook 所需环境
- 通过 AzureML 使用 BatchAI 群集进行云端分布式训练

系统需求

本次实验需要:

- Windows 10
- Anaconda 及 Python3.6 环境

- AzureML Python SDK
 - Github AzureML Notebook 样本
-

设置

如果您使用的是大会提供的实验设备，请跳过此节。如果您使用自己的设备，请进行以下设置以进行本次实验：

1. 安装 Microsoft Windows 10.
2. 安装 Anaconda 及 Python3.6 环境.
3. 在 Anaconda 中安装 AzureML Python SDK: `pip install azureml-sdk[notebooks]`

详见 <https://docs.microsoft.com/zh-cn/azure/machine-learning/service/quickstart-create-workspace-with-python>

4. 安装 Git 环境
-

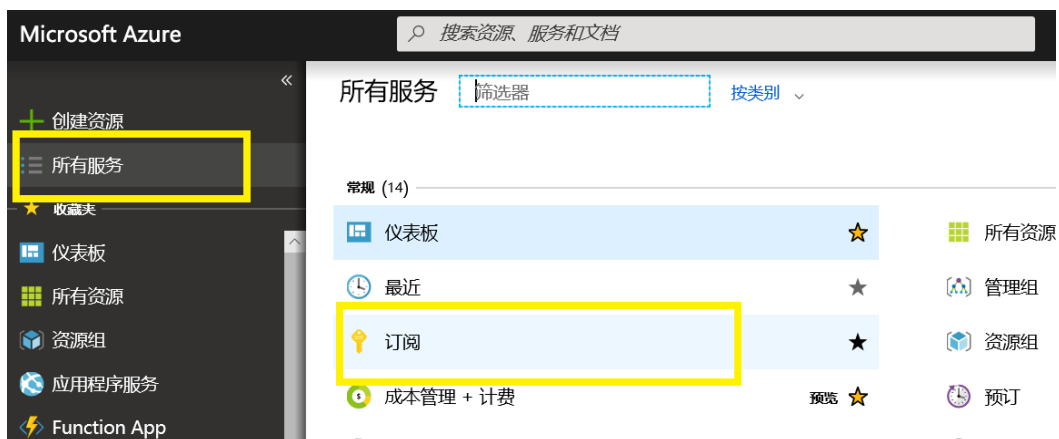
本实验所需时间为 40 分钟。

Part 1: 配置 AzureML Notebook 所需环境

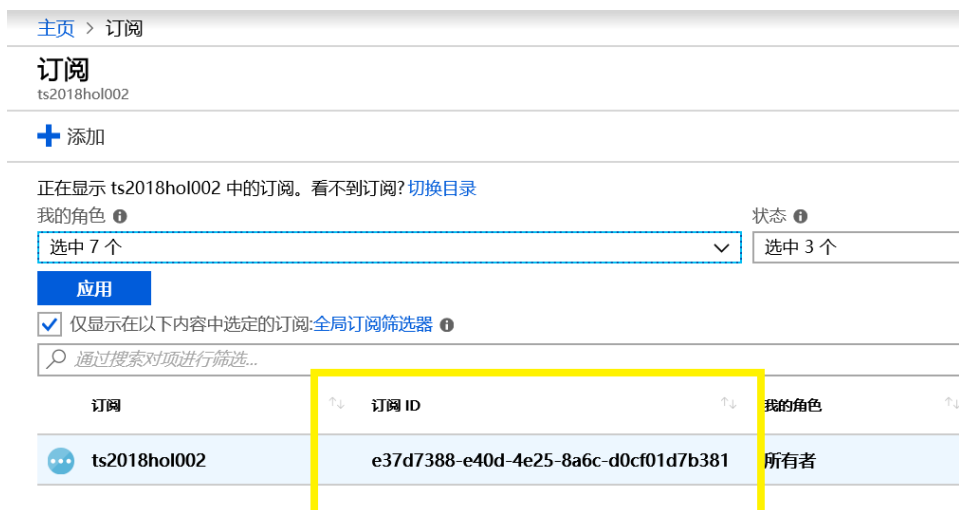
Task 0 –在 Azure Portal 中找到实验所需的 Azure 订阅

每位学员将使用大会提供的 Azure 订阅来进行实验

1. 在桌面点击打开 Edge 浏览器，打开 portal.azure.com 页面
2. 使用大会提供的账号及密码登入
3. 在左上角所有服务选项中选择进入“订阅”



4. 在进入的页面中我们便可以找到实验将使用的订阅 ID

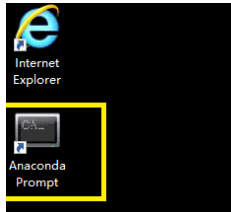


Task 1 – 下载本实验相关 AzureML 样本，启动 Notebook Server

首先，我们需要从 Github 上将本实验相关的 AzureML 样本下载在本地目录

(<https://github.com/Azure/MachineLearningNotebooks>)

5. 在桌面点击“anaconda prompt”，选择对应的程序打开一个新的 Anaconda 终端



6. 输入 `git clone https://github.com/lliimsft/MachineLearningNotebooks.git`
7. 进入目录 `cd MachineLearningNotebooks`
8. 在目录内通过 `jupyter notebook` 开起本地的 Notebook Server

```
Anaconda Prompt - jupyter notebook

(azureml) C:\Users\TechSummitUser>git clone https://github.com/azure/machinelearningnotebooks.git
Cloning into 'machinelearningnotebooks'...
remote: Enumerating objects: 137, done.
remote: Counting objects: 100% (137/137), done.
remote: Compressing objects: 100% (101/101), done.
Remote: Total 747 (delta 64), reused 85 (delta 35), pack-reused 610
Receiving objects: 100% (747/747), 1.17 MiB | 196.00 KiB/s, done.
Resolving deltas: 100% (422/422), done.

(azureml) C:\Users\TechSummitUser>cd machinelearningnotebooks

(azureml) C:\Users\TechSummitUser\machinelearningnotebooks>jupyter notebook
[I 16:53:06.581 NotebookApp] Serving notebooks from local directory: C:\Users\TechSummitUser\machinelearningnotebooks
[I 16:53:06.581 NotebookApp] The Jupyter Notebook is running at:
[I 16:53:06.581 NotebookApp] http://localhost:8888/?token=d8029840d02480f6e801f25400f53c3e6864f0992b156eff
[I 16:53:06.581 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:53:06.597 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=d8029840d02480f6e801f25400f53c3e6864f0992b156eff
[I 16:53:08.815 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

Task 2 – 在本地创建 Azure ML Notebook 运行所需的配置文件

1. 在弹出的 Jupyter notebook 主窗口中，点击进入 0.configuration.ipynb。通过运行这个 Jupyter 脚本，我们将配置本地的 notebook library，并创建一个新的 Azure Machine Learning workspace



2. 请选择默认的 Python3 Kernel。
3. 运行第一个 Cell 查看目前所使用的 AzureML SDK 版本

```
In [1]: import azureml.core
        print("SDK Version:", azureml.core.VERSION)

SDK Version: 0.1.65
```

4. 在第二个 Cell 里，填入本实验所使用的 Azure 订阅 ID，并输入希望创建的 Azure 资源组名及 AzureML Workspace 名称。注意，请尽量使用唯一性较强的 Azure 资源组名，以避免冲突。运行期间浏览器将会弹出交互式登入的提示，请输入你的 Azure 账号相关信息进行登录。

```
In [3]: from azureml.core import Workspace

subscription_id = '029ee373-e6af-40bb-9776-fbade18e8669'
resource_group = 'my_hol_rg'
workspace_name = 'my_first_workspace'

try:
    ws = Workspace(subscription_id = subscription_id, resource_group = resource_group, workspace_name = workspace_name)
    ws.write_config()
    print('Workspace configuration succeeded. You are all set!')
except:
    print('Workspace not found. Run the cells below.')

get_workspace error using subscription_id=029ee373-e6af-40bb-9776-fbade18e8669, resource_group_name=my_hol_rg, workspace_name=my_first_workspace

Workspace not found. Run the cells below.
```

5. 运行接下三个 Cells 创建 Azure 资源组和 AzureML Workspace，并生成本地配置文件。

```
In [3]: workspace_region = "eastus2"

In [4]: import os

subscription_id = os.environ.get("SUBSCRIPTION_ID", subscription_id)
resource_group = os.environ.get("RESOURCE_GROUP", resource_group)
workspace_name = os.environ.get("WORKSPACE_NAME", workspace_name)
workspace_region = os.environ.get("WORKSPACE_REGION", workspace_region)

In [5]: # import the Workspace class and check the azureml SDK version
from azureml.core import Workspace

ws = Workspace.create(name = workspace_name,
                     subscription_id = subscription_id,
                     resource_group = resource_group,
                     location = workspace_region,
                     create_resource_group = True,
                     exist_ok = True)

ws.get_details()
ws.write_config()

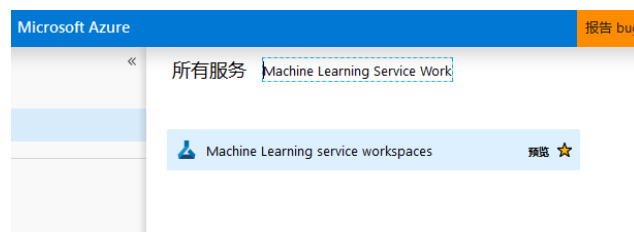
UserWarning: The resource group doesn't exist or was not provided. AzureML SDK is creating a resource group=my_hol_rg in location=eastus2 using subscription=029ee373-e6af-40bb-9776-fbade18e8669.

Wrote the config file config.json to: C:\Users\Li\Documents\MachineLearningNotebooks\aml_config\config.json
```

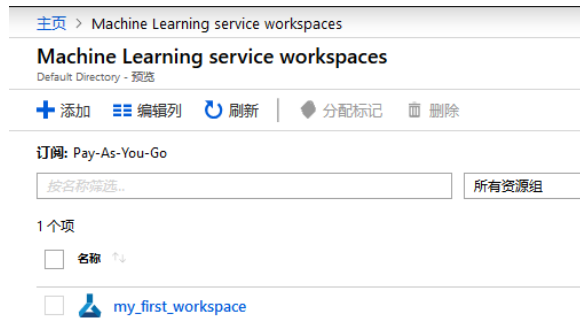
Task 3 – 通过 Azure Portal 查找 AzureML 相关资源

执行完成之前步骤后，对应的 AzureML Workspace 应该已经创建成功。这时，我们可以登入 Azure Portal 查看刚刚创建好的 Workspace。

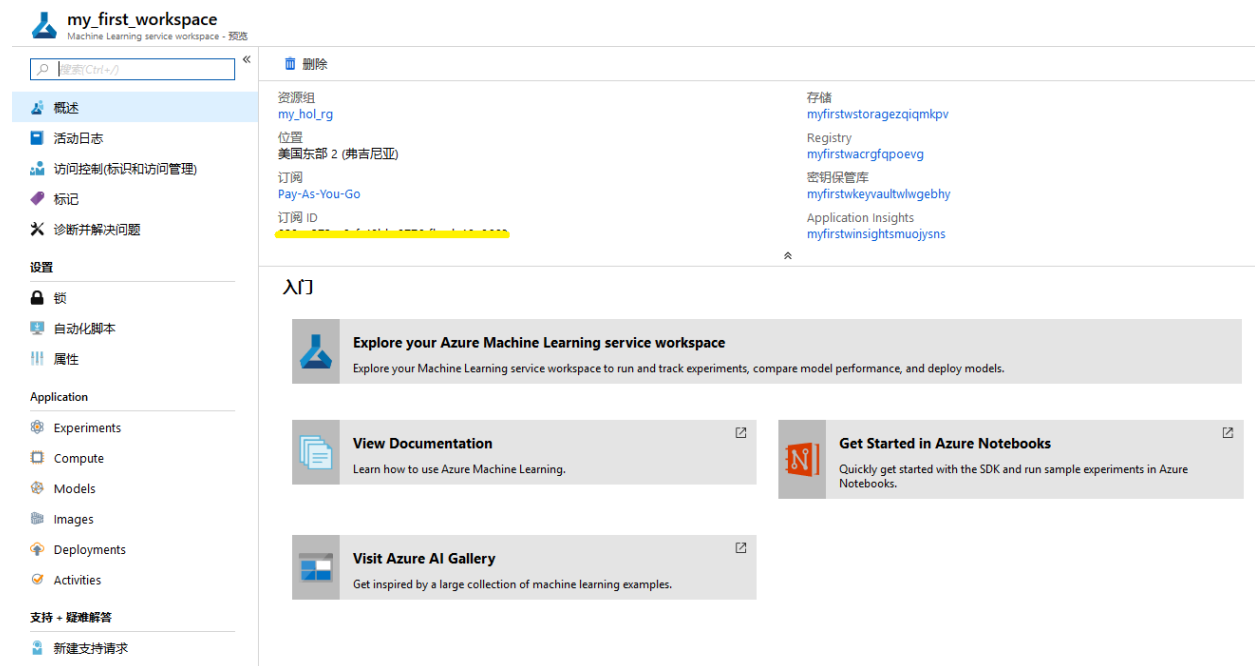
1. 登入你的 Azure Portal 账户，在左上角所有服务选项中，搜索“Machine Learning Service Workspaces”。



2. 在显示的列表中，你将可以找到刚才创建好的 workspace。



用户可以通过 AzureML Workspace 的 Portal 界面查看并管理模型训练的实验(Experiments)以及运算对象(Compute)。在创建 Workspace 时候自动，系统也会自动为用户创建一个 Azure 存储账号及容器注册表，用来存放训练实验相关的文件及 Docker 镜像。



Part 2: 通过 AzureML 使用 BatchAI 群集进行分布式训练

在这一部分，我们将介绍如何通过 AzureML 在 Azure 服务器端进行分布式 Tensorflow 训练。在这个例子中，我们将创建一个自动伸缩的 BatchAI 虚拟机群集，并将这个群集作为计算对象进行基于 MNIST 手写数据的神经网络模型训练。

Task 1 – 进入 Notebook 文件并初始化 Workspace

1. 回到 Notebook 主界面，进入并打开 training/05.distributed-tensorflow-with-parameter-server/05.distributed-tensorflow-with-parameter-server.ipynb
2. 运行第一个 Cell 检查 AzureML SDK 版本。再运行第二个 Cell 载入之前生成的配置文件，获取订阅号，数据中心，资源组及 Workspace 相关信息。

```
In [2]: from azureml.core.workspace import Workspace

ws = Workspace.from_config()
print('Workspace name: ' + ws.name,
      'Azure region: ' + ws.location,
      'Subscription id: ' + ws.subscription_id,
      'Resource group: ' + ws.resource_group, sep = '\n')

Found the config file in: C:\Users\llii\MachineLearningNotebooks\aml_config\config.json
Workspace name: my_first_workspace
Azure region: eastus2
Subscription id: lcbalda6-5a83-45e1-a88e-8b397eb84356
Resource group: llii_hol_10_25
```

Task 2 – 创建 BatchAI 虚拟机群集作为运算对象

在第三个 Cell 中，我们将创建一个 BatchAI 虚拟机群集，并将其作为我们模型训练的计算对象：

- 请将 BatchAI 群集命名为“cpucluster”
- 将群集的虚拟机类型设置为“STANDARD_D2_v2”，即每台虚拟机配有两块 CPU
- 我们启动 BatchAI 群集的自动伸缩功能，规定此群集的最小虚拟机数为 2 台，最大虚拟机数为 4 台：当有训练作业被提交时，群集将自动分配新的虚拟机进行训练，当所有任务完成后，群集也将自动释放闲置的虚拟机以节约资源。

- 如果该名称的 BatchAI 群集已存在，我们将使用已有资源。反之，将创建一个新的群集。

```
In [3]: from azureml.core.compute import ComputeTarget, BatchAiCompute
        from azureml.core.compute_target import ComputeTargetException

        # choose a name for your cluster
        cluster_name = "cpcluster"

        try:
            compute_target = ComputeTarget(workspace=ws, name=cluster_name)
            print('Found existing compute target.')
        except ComputeTargetException:
            print('Creating a new compute target...')
            compute_config = BatchAiCompute.provisioning_configuration(vm_size='STANDARD_D2_V2',
                                                                      autoscale_enabled=True,
                                                                      cluster_min_nodes=2,
                                                                      cluster_max_nodes=4)

            # create the cluster
            compute_target = ComputeTarget.create(ws, cluster_name, compute_config)

            compute_target.wait_for_completion(show_output=True, min_node_count=0)

            # Use the 'status' property to get a detailed status for the current cluster.
            print(compute_target.status.serialize())

        Creating a new compute target...
        Creating
        succeeded.....
        BatchAI wait for completion finished
        Minimum number of nodes requested have been provisioned
        {'allocationState': 'steady', 'allocationStateTransitionTime': '2018-10-24T15:31:44.868000+00:00', 'creationTime': '2018-10-24T15:30:16.313000+00:00', 'currentNodeCount': 2, 'errors': None, 'nodeStateCounts': {'idleNodeCount': 0, 'leavingNodeCount': 0, 'preparingNodeCount': 2, 'runningNodeCount': 0, 'unusableNodeCount': 0}, 'provisioningState': 'succeeded', 'provisioningStateTransitionTime': '2018-10-24T15:30:29.134000+00:00', 'scaleSettings': {'manual': None, 'autoScale': {'maximumNodeCount': 4, 'minimumNodeCount': 2, 'initialNodeCount': 2}}, 'vmPriority': 'dedicated', 'vmSize': 'STANDARD_D2_V2'}
```

当 BatchAI 群集创建成功，我们可以在 Portal 中点击 Workspace 目录下的“Compute”，检查该群集的各项属性。

Task 3 – 创建本地源目录并放入训练脚本

此次训练，我们将使用 AzureML 提供的 Tensorflow 训练脚本

(https://github.com/Azure/MachineLearningNotebooks/blob/master/training/05.distributed-tensorflow-with-parameter-server/tf_mnist_replica.py)。

这个脚本使用基于 parameter server 分布式训练框架，并将自动下载所需的 MNIST 训练数据。

我们在本地创建一个名为“tf-distr-ps”的文件夹作为此次训练的目录，将脚本文件放入此目录中。

```
In [4]: import os

project_folder = './tf-distr-ps'
os.makedirs(project_folder, exist_ok=True)
```

Copy the training script `tf_mnist_replica.py` into this project directory.

```
In [5]: import shutil
shutil.copy('tf_mnist_replica.py', project_folder)

Out[5]: './tf-distr-ps\\tf_mnist_replica.py'
```

Task 4 – 定义实验及分布式 Tensorflow 作业

1. 我们将此次实验命名为“tf-distr-ps”。

```
In [6]: from azureml.core import Experiment

experiment_name = 'tf-distr-ps'
experiment = Experiment(ws, name=experiment_name)
```

2. 下一步，我们将给出此次训练任务的具体定义：
 - a. 定义此次训练任务的 estimator 使用 Tensorflow 框架
 - b. 使用刚才创建的 BatchAI 群集作为计算对象
 - c. 通过“script_params”定义训练脚本的参数：我们设置每个 worker 的 GPU 数改为 0
 - d. 设置此次训练将使用 2 台虚拟机
 - e. 在“distributed_backend”将分布式模块设置为“ps”，即 parameter server，并将 worker 数跟 ps 数分别设置为 2 和 1。这样一来，我们将在两台虚机上分别启动一个 worker，每个 worker 占用一块 CPU
 - f. 因为本作业不使用 GPU，将 use_gpu 设置为 False

```
In [8]: from azureml.train.dnn import TensorFlow

script_params={
    '--num_gpus': 0,
    '--train_steps': 1000
}

estimator = TensorFlow(source_directory=project_folder,
                       compute_target=compute_target,
                       script_params=script_params,
                       entry_script='tf_mnist_replica.py',
                       node_count=2,
                       worker_count=2,
                       parameter_server_count=1,
                       distributed_backend='ps',
                       use_gpu=False)
```

Task 5 – 提交并监督训练作业

运行接下来的 Cell 提交训练作业至 AzureML 服务器。

Submit job

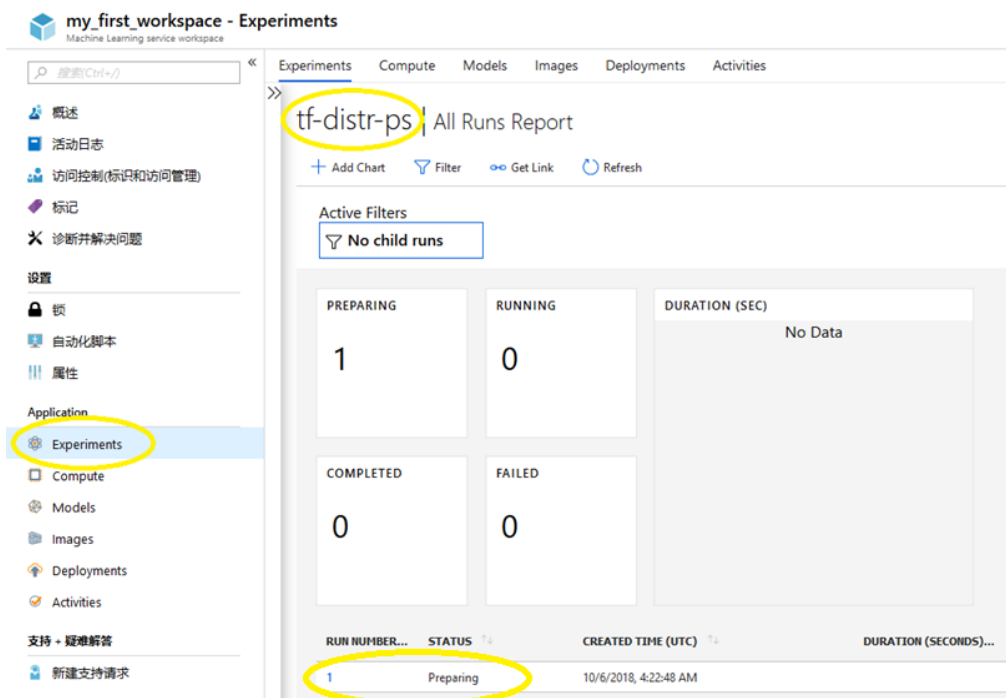
Run your experiment by submitting your estimator object. Note that this call is asynchronous.

```
In [9]: run = experiment.submit(estimator)
print(run.get_details())

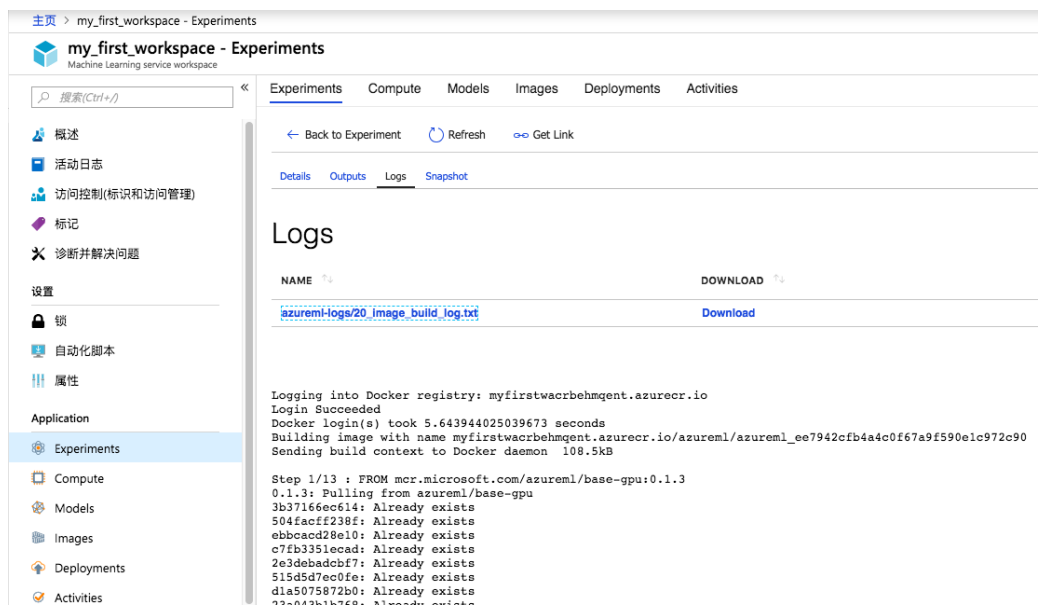
{'runId': 'tf-distr-ps_1538980408142', 'target': 'gpucluster', 'status': 'Preparing', 'properties': {'azureml.runsource': 'experiment', 'ContentSnapshotId': '005ec762-be3a-4db6-aa28-cc426a489658', 'runDefinition': {'Script': 'tf_mnist_replica.py', 'Arguments': ['--num_gpus', '1', '--train_steps', '10000'], 'Framework': 3, 'Target': 'gpucluster', 'DataReferences': {}}, 'JobName': None, 'AutoPrepareEnvironment': True, 'MaxRunDurationSeconds': None, 'Environment': {'Python': {'InterpreterPath': 'python', 'UserManagedDependencies': False, 'CondaDependencies': {'name': 'project_environment', 'dependencies': ['python=3.6.2', {'pip': ['azureml-defaults']], 'tensorflow-gpu=1.9.0'}], 'CondaDependenciesFile': None}, 'EnvironmentVariables': {'EXAMPLE_ENV_VAR': 'EXAMPLE_VALUE', 'NCCL_SOCKET_IFNAME': '^docker0'}, 'Docker': {'BaseImage': 'mcr.microsoft.com/azureml/base-gpu:0.1.3', 'Enabled': True, 'SharedVolumes': True, 'GpuSupport': True, 'Arguments': [], 'BaseImageRegistry': {'Address': None, 'Username': None, 'Password': None}}, 'Spark': {'Repositories': ['https://mmlspark.azureedge.net/maven'], 'Packages': [{'Group': 'com.microsoft.ml.spark', 'Artifact': 'mmlspark_2.11', 'Version': '0.12'}], 'PrecachePackages': True}}, 'History': {'OutputCollection': True}, 'Spark': {'Configuration': {'spark.app.name': 'Azure ML Experiment', 'spark.yarn.maxAppAttempts': '1'}}, 'BatchAI': {'NodeCount': 2}, 'Tensorflow': {'WorkerCount': 2, 'ParameterServerCount': 1}, 'Mpi': {'ProcessCountPerNode': 1}, 'Hdi': {'YarnDeployMode': 2}, 'ContainerInstance': {'Region': None, 'CpuCores': 1, 'MemoryGb': 4}, 'ExposedPorts': None, 'PrepareEnvironment': None}, 'logFiles': {'azureml-logs/20_image_build_log.txt': 'https://myfirstwstora.gezqigmkpv.blob.core.windows.net/azureml/ExperimentRun/tf-distr-ps_1538980408142/azureml-logs/20_image_build_log.txt?sv=2017-04-17&sr=b&sig=neJ3wSjAzeet1tbjP48WegjqIGS1mm45BCtxj%2Bv8Yc%3D&st=2018-10-08T06%3A23%3A32Z&se=2018-10-08T14%3A33%3A32Z&sp=r'}}}
```

我们可以留在 Notebook 实时监管任务进度,同时也可以 Portal 的 Experiments 中查找刚刚创建的实验。

若作业创建成功,我们可以并发现一个正在 Preparing 的作业 (run)。



当作业处于 Preparing 状态的时候，AzureML 正在创建适用于本次训练作业的 docker 镜像，包括 Tensorflow 运行环境及相关 Packages。我们可以在 portal 页面点进这一作业，在“Logs”栏中点击“azureml-logs/20_image_build_log.txt”可以实时查看镜像创建的日志。



Docker 镜像创建大约需要不到 10 分钟的时间，创建好的镜像将被存入 Workspace 附属的容器注册表中，以便今后同类作业使用。

之后，训练作业将从 Preparing 状态进入 Running 状态， BatchAI 群集此时将自动申请 2 台新的虚拟机。虚拟机分配及配置过程大约需要 3 至 4 分钟，完成后训练作业将正式开始执行。

我们同样可以通过实时日志的方式追踪作业进程。刷新 Portal 界面直到更多日志文件的出现。我们可以点击查看“azureml-logs/80_driver_log_worker_0.txt”的内容，这个文件即是 worker 0 的训练日志。同理，“azureml-logs/80_driver_log_worker_1.txt”为 worker 1 的训练日志。

当训练成功完成，作业状态将变为“Completed”。如有错误发生，作业状态会变为“Failed”，可以通过检查日志的方式进行排查。

Experiments	Compute	Models	Images	Deployments	Activities
azureml-logs/80_driver_log_ps_0.txt					Download
azureml-logs/80_driver_log_worker_0.txt					Download
azureml-logs/80_driver_log_worker_1.txt					Download
azureml-logs/azureml.log					Download
driver_log					Download


```

WARNING:tensorflow:From tf_mnist_replica.py:85: read_data_sets (from tensorflow.contrib.learn.python.learn.data
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
WARNING:tensorflow:From /azureml-envs/azureml_079d27a647dac0e3b594f81064adeded/lib/python3.6/site-packages/tens
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From /azureml-envs/azureml_079d27a647dac0e3b594f81064adeded/lib/python3.6/site-packages/tens
Instructions for updating:
Please use urllib or similar directly.
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
WARNING:tensorflow:From /azureml-envs/azureml_079d27a647dac0e3b594f81064adeded/lib/python3.6/site-packages/tens
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting outputs/MNIST/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
WARNING:tensorflow:From /azureml-envs/azureml_079d27a647dac0e3b594f81064adeded/lib/python3.6/site-packages/tens
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting outputs/MNIST/train-labels-idx1-ubyte.gz

```

同样的，我们也可以在 notebook 端使用 Jupyter widget 插件追踪作业进程，查看虚拟机状态及日志文件：

```
[17]: from azureml.train.widgets import RunDetails
RunDetails(run).show()
```

Run Properties		Output Logs
Status	Completed	Distributed run - Show log by process worker_0 1539158913.028963: Worker 1: training step 356 done (global step: 1000) Training ends @ 1539158913.029005 Training elapsed time: 2.332448 s After 1000 training step(s), validation cross entropy = 619.888 The experiment completed successfully. Finalizing run... Logging experiment finalizing status in history service Run is completed.
Start Time	10/10/2018 1:00:00 AM	
Duration	0:08:42	
Run Id	tf-distr-ps_1539158399423	
Arguments	--num_gpus 0 --train_steps 1000	

以上为本实验的所有内容，感谢您的参与！