**Create and Manage an Azure Machine Learning Workspace**

**Understand the Azure Machine Learning Service**

* **Get Access to Azure:**
  + Sign in to the Azure portal and get access to an Azure subscription.
  + Create a resource group within your subscription.
  + Create an Azure Machine Learning service to provision a workspace.
* **Provisioned Resources:**
  + **Azure Storage Account:** Stores files, notebooks, and job metadata.
  + **Azure Key Vault:** Manages secrets securely.
  + **Application Insights:** Monitors predictive services.
  + **Azure Container Registry:** Stores images for environments when needed.

**Create the Workspace**

* **Methods to Create:**
  + **Azure Portal:** Use the UI to create the service.
  + **Azure Resource Manager (ARM) Template:** Deploy an ARM template.
  + **Azure Command Line Interface (CLI):** Use CLI with the Machine Learning CLI extension.
  + **Azure Machine Learning Python SDK:** Example code:

**from azure.ai.ml.entities import Workspace**

workspace\_name = "mlw-example"

ws\_basic = Workspace(

name=workspace\_name,

location="eastus",

display\_name="Basic workspace-example",

description="This example shows how to create a basic workspace",

)

**ml\_client.workspaces.begin\_create(ws\_basic)**

**Explore the Workspace**

* **Azure Portal:**
  + Access the workspace from the portal's overview page.
  + Launch **Azure Machine Learning Studio** for managing resources.

**Manage Access**

* **Role-Based Access Control (RBAC):**
  + **Owner:** Full access and can grant access.
  + **Contributor:** Full access but cannot grant access.
  + **Reader:** View-only access.
* **Azure Machine Learning Specific Roles:**
  + **AzureML Data Scientist:** Full workspace actions except compute creation/deletion.
  + **AzureML Compute Operator:** Manages compute resources.
* **Custom Roles:** Create custom roles if built-in roles do not meet specific needs.

**Organize Workspaces**

* **Single vs. Multiple Workspaces:**
  + Initially, use a single workspace.
  + For large-scale projects, use multiple workspaces based on projects, environments (test/production), or teams.

**Identifying Azure Machine Learning Resources**

**1. The Workspace**

* **Role:** Central resource for Azure Machine Learning.
* **Contents:** Logs, metrics, outputs, models, and code snapshots.
* **Access:** Controlled to ensure data scientists can train and track models and deploy them to endpoints without having unrestricted access.

**2. Compute Resources**

Compute resources are essential for training and deploying models. They include:

* **Compute Instances:**
  + Description: Cloud-based virtual machines managed by the workspace.
  + Use Case: Development environment for running (Jupyter) notebooks.
* **Compute Clusters:**
  + Description: On-demand clusters of CPU or GPU compute nodes.
  + Use Case: Production workloads that require scalable resources.
* **Kubernetes Clusters:**
  + Description: Create or attach Azure Kubernetes Service (AKS) clusters.
  + Use Case: Deploy trained models in production.
* **Attached Computes:**
  + Description: Attach external Azure compute resources like Azure Databricks or Synapse Spark pools.
  + Use Case: Integrate with other Azure compute services for flexible resource use.
* **Serverless Compute:**
  + Description: Fully managed, on-demand compute for training jobs.
  + Use Case: Cost-efficient, scalable training without manual resource management.
  + Note: Managed by Azure Machine Learning and not listed in the compute page.

**3. Datastores**

* **Role:** References to Azure data services, storing connection information in Azure Key Vault.
* **Pre-configured Datastores:**
  + **workspaceartifactstore:**
    - Connects to: Azureml container of the Azure Storage account.
    - Use: Stores compute and experiment logs.
  + **workspaceworkingdirectory:**
    - Connects to: File share of the Azure Storage account.
    - Use: Used by the Notebooks section of the studio for file uploads.
  + **workspaceblobstore:**
    - Connects to: Blob Storage of the Azure Storage account (azureml-blobstore-... container).
    - Use: Default datastore for data assets and uploads.
  + **workspacefilestore:**
    - Connects to: File share of the Azure Storage account (azureml-filestore-... file share).
    - Use: General file storage needs within the workspace.
* **Custom Datastores:**
  + Can be created to connect to other Azure data services like Azure Storage Accounts or Azure Data Lake Storage (Gen2).

**Summary**

* **Workspace:** Central resource for managing ML resources.
* **Compute Resources:** Include instances, clusters, Kubernetes, attached computes, and serverless compute.
* **Datastores:** Pre-configured and custom datastores for storing data and logs.

**Identifying Azure Machine Learning Assets**

**1. Models**

* **Purpose:** The end product of training a machine learning model.
* **Storage Formats:**
  + Python pickle file (.pkl).
  + MLflow MLModel format.
* **Management:**
  + Models are registered in the workspace with a name and version for easy tracking and deployment.
  + Versioning helps in maintaining multiple iterations of models.

**2. Environments**

* **Purpose:** Ensure that code runs consistently across different compute resources.
* **Components:**
  + Software packages.
  + Environment variables.
  + Software settings.
* **Management:**
  + Environments are specified with a name and version.
  + Stored as images in the Azure Container Registry created with the workspace.
  + Ensures necessary components are installed on the compute before code execution, providing robustness and reusability.

**3. Data**

* **Purpose:** Refer to specific files or folders that contain the data needed for machine learning tasks.
* **Management:**
  + Data assets point to files or folders in datastores.
  + Created with specified paths, names, and versions for easy access without repeated authentication.

**4. Components**

* **Purpose:** Reusable pieces of code that can be shared across projects.
* **Usage:** Often represent steps in a machine learning pipeline (e.g., data normalization, model training, model testing).
* **Management:**
  + Components are created with a name, version, code, and environment needed for execution.
  + Facilitate code reuse and sharing within the workspace.

**Training Models in the Workspace**

**1. Automated Machine Learning**

* **Purpose:** Experiment with various algorithms and hyperparameter values to find the best-performing model.
* **Process:**
  + Iterates through algorithm-feature pairs to identify optimal configurations.
  + Speeds up the model selection process significantly.

**2. Run a Notebook**

* **Purpose:** Develop and execute code interactively using Jupyter notebooks.
* **Features:**
  + Accessible through the Notebooks page in the Azure Machine Learning studio.
  + All files are stored in the file share of the Azure Storage account linked with the workspace.
  + Utilizes compute instances, ideal for development and experimentation.

**3. Run a Script as a Job**

* **Purpose:** Prepare code for production use and automate machine learning workloads.
* **Features:**
  + Scripts can be submitted as jobs to the workspace, with inputs and outputs stored for reproducibility.
  + Different job types available:
    - **Command:** Execute a single script.
    - **Sweep:** Perform hyperparameter tuning for a script.
    - **Pipeline:** Execute a sequence of scripts or components.

The Python SDK for Azure Machine Learning is an ideal tool for data scientists that can be used in any Python environment. Whether you normally work with Jupyter notebooks, Visual Studio Code, you can install the Python SDK and connect to the workspace.

pip install azure-ai-ml

After the Python SDK is installed, you'll need to connect to the workspace. By connecting, you're authenticating your environment to interact with the workspace to create and manage assets and resources.

To authenticate, you need the values to three necessary parameters:

* subscription\_id: Your subscription ID.
* resource\_group: The name of your resource group.
* workspace\_name: The name of your workspace.

Next, you can define the authentication by using the following code:

**from azure.ai.ml import MLClient ``**

**from azure.identity import DefaultAzureCredential ``**

ml\_client = MLClient(

DefaultAzureCredential(), subscription\_id, resource\_group, workspace``

)

For example, you'll connect to the workspace when you create a new job to train a model:

from azure.ai.ml import command

# configure job

job = command(

code="./src",

command="python train.py",

environment="AzureML-sklearn-0.24-ubuntu18.04-py37-cpu@latest",

compute="aml-cluster",

experiment\_name="train-model"

)

# connect to workspace and submit job

returned\_job = ml\_client.create\_or\_update(job)``

 **URIs in Azure Machine Learning:**

* URIs reference the location of data in Azure Machine Learning.
* URIs need to be prefixed with the appropriate protocol for Azure Machine Learning to connect to the data.

 **Common Protocols:**

* **http(s):**
  + Used for data stores in Azure Blob Storage or publicly accessible http(s) locations.
* **abfs(s):**
  + Used for data stores in Azure Data Lake Storage Gen 2.
* **azureml:**
  + Used for data stored in a datastore within Azure Machine Learning.

 **Accessing Data in Azure Blob Storage:**

* Data can be stored in containers within Azure Blob Storage.
* Accessible via URIs with the http(s) protocol.
* If the container is private, authentication like a Shared Access Signature (SAS) is required.

 **Datastore in Azure Machine Learning:**

* A reference to an existing storage account on Azure.
* Stores connection and authentication information within the workspace.
* Accessed using the azureml protocol, eliminating the need for authentication in the code.

**Benefits of Datastores:**

* Provide easy-to-use URIs for data storage.
* Facilitate data discovery within Azure Machine Learning.
* Securely store connection information without exposing secrets and keys.

**Authentication Methods for Datastores:**

* **Credential-based:** Utilize a service principal, shared access signature (SAS) token, or account key for authentication.
* **Identity-based:** Use Microsoft Azure Active Directory (Azure AD) identities or managed identities.

**Types of Supported Datastores:**

* Azure Blob Storage
* Azure File Share
* Azure Data Lake (Gen 2)

**Built-in Datastores:**

* Each workspace includes four built-in datastores, two for Azure Storage blob containers and two for Azure Storage file shares.
* These are system storages used by Azure Machine Learning.

**Creating a Datastore:**

* Datastores are attached to workspaces and store connection information to storage services.
* Simplify connection to storage services without repetitive authentication.
* Provide a protective layer for accessing data without direct access to the storage service.

**Creating a Datastore for Azure Blob Storage:**

* Datastores can be created via GUI, Azure CLI, or Python SDK.
* Different authentication options are available depending on the storage service used

**Understanding Compute Targets in Azure Machine Learning:**

* **Compute Instances:**
  + Similar to virtual machines, ideal for running notebooks.
  + Best for experimentation and development.
  + Continuously running, managed by Azure Machine Learning.
* **Compute Clusters:**
  + Multi-node clusters of virtual machines.
  + Automatically scale up or down based on demand.
  + Cost-effective for large data processing and parallel workloads.
  + Suitable for running scripts and batch jobs.
* **Kubernetes Clusters:**
  + Based on Kubernetes technology for more control over compute configuration and management.
  + Can attach self-managed Azure Kubernetes Service (AKS) clusters or on-premises Arc Kubernetes clusters.
  + Ideal for scenarios requiring custom configurations and control.
* **Attached Compute:**
  + Allows existing compute resources like Azure VMs or Azure Databricks clusters to be attached to your workspace.
  + Leverages pre-existing infrastructure.
* **Serverless Compute:**
  + Fully managed, on-demand compute.
  + Suitable for training jobs without the need to manage compute resources.

**When to Use Each Type of Compute:**

* **Experimentation:**
  + Use **Compute Instances** for a Jupyter notebook experience, suitable for small datasets and experimentation.
  + Alternatively, use **Spark serverless compute** for distributed compute power in notebooks.
* **Production:**
  + Use **Compute Clusters** for training models with large datasets. Clusters scale on-demand for efficiency.
  + For an alternative without the need for management, use **Azure Machine Learning's serverless compute**.
* **Deployment:**
  + **Batch Predictions:** Use compute clusters or serverless compute for scalable, on-demand pipeline jobs.
  + **Real-time Predictions:** Use containers for lightweight, cost-efficient deployments. Managed online endpoints can be used for automatic container management, or attach Kubernetes clusters for real-time prediction workloads.

**Summary of Best Practices:**

* **Compute Instances** for experimentation and development.
* **Compute Clusters** for production training and batch jobs.
* **Kubernetes Clusters** for custom-configured and managed deployments.
* **Attached Compute** to leverage existing infrastructure.
* **Serverless Compute** for fully managed, on-demand training jobs.

**Assigning a Compute Instance to a User in Azure Machine Learning:**

* **Single User Assignment:**
  + A compute instance can only be assigned to one user at a time.
  + Ensure you have appropriate permissions to assign compute instances to users.

**Minimize Compute Time:**

* **Start and Stop Manually:**
  + Users can manually start and stop the compute instance from the Azure Machine Learning studio.
* **Scheduling Start and Stop:**
  + Create a schedule for the compute instance to automatically start and stop at specific times.
  + This helps in saving costs by ensuring the compute instance isn't running when not in use.
* **Auto-shutdown on Idle:**
  + Configure the compute instance to automatically shut down after a specified idle period.
  + This further optimizes cost by preventing the compute instance from running unnecessarily.

**Using a Compute Instance:**

* **Azure Machine Learning Studio:**
  + Use the integrated notebooks experience within the Azure Machine Learning studio.
  + Directly attach the compute instance to notebooks to run cells.
* **Visual Studio Code:**
  + For enhanced source control and development experience, use Visual Studio Code.
  + Attach the compute instance to run notebook cells remotely from Visual Studio Code.

 **Azure Machine Learning Environments:**

* Define runtime configurations for experiments, specifying Python runtime versions and installed packages.
* Automatically created curated environments are available upon creating a workspace.
* Custom environments can be created, managed, and registered in the workspace for consistent, reusable contexts.

 **Environment Definition:**

* Python code runs in a virtual environment, managed using conda or pip.
* Environments are typically created in Docker containers hosted on various compute targets (development computers, VMs, cloud clusters).

 **Building and Using Environments:**

* Azure Machine Learning builds environment definitions into Docker images and conda environments.
* Environments are built on the Azure Container registry associated with the workspace.

 **Viewing and Managing Environments:**

* Environments can be listed and managed via the Azure Machine Learning studio, Azure CLI, or Python SDK.
* Example to list environments using Python SDK:

envs = ml\_client.environments.list()

for env in envs:

print(env.name)

* Example to review a specific environment by its name:

env = ml\_client.environments.get(name="my-environment", version="1")

print(env)

**Curated Environments in Azure Machine Learning**

**Overview:**

* Curated environments are prebuilt for common machine learning workloads.
* These environments use the prefix **AzureML-**.
* They support popular frameworks and tools like Scikit-Learn, TensorFlow, and PyTorch.

**Exploring Curated Environments:**

* You can view curated environments in the Azure Machine Learning studio, using Azure CLI, or Python SDK.
* Example to retrieve description and tags using Python SDK:

env = ml\_client.environments.get("AzureML-sklearn-0.24-ubuntu18.04-py37-cpu", version=44)

print(env.description, env.tags)

**Using Curated Environments:**

* Typically used when running scripts as command jobs.
* Reference an environment by its name and version.
* Example of configuring a command job with Python SDK:

from azure.ai.ml import command

# configure job

job = command(

code="./src",

command="python train.py",

environment="AzureML-sklearn-0.24-ubuntu18.04-py37-cpu@latest",

compute="aml-cluster",

display\_name="train-with-curated-environment",

experiment\_name="train-with-curated-environment"

)

# submit job

returned\_job = ml\_client.create\_or\_update(job)

**Testing and Troubleshooting Curated Environments:**

* Curated environments ensure faster deployment times.
* Verify necessary packages are included by reviewing environment details.
* Test by running the script with the curated environment.