marlabs
driving digital agility

Madvisor -AutoML

Table of content

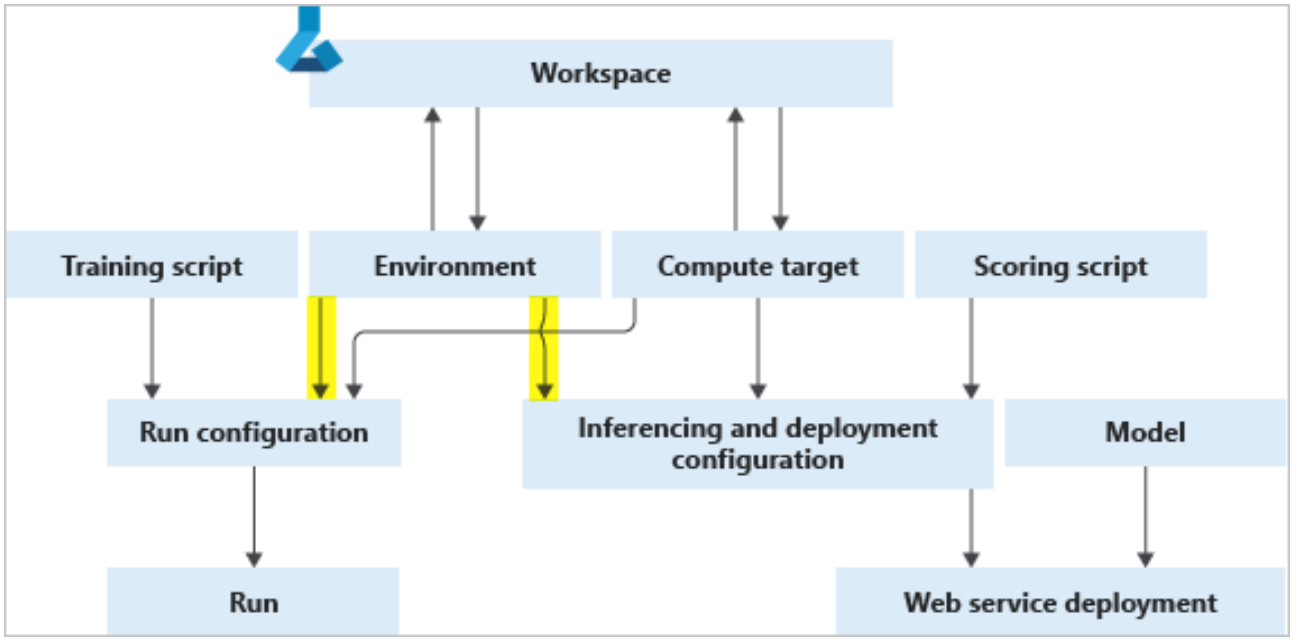
page

Train model using a the AutoML Docker image

In this article, we will create, view, and delete **Azure Machine Learning workspaces** for Azure Machine Learning, using the Azure portal**.**

Prerequisites

An Azure subscription. If you don't have an Azure subscription, create a free account before you begin. Try the **free or paid version of Azure Machine Learning** today.

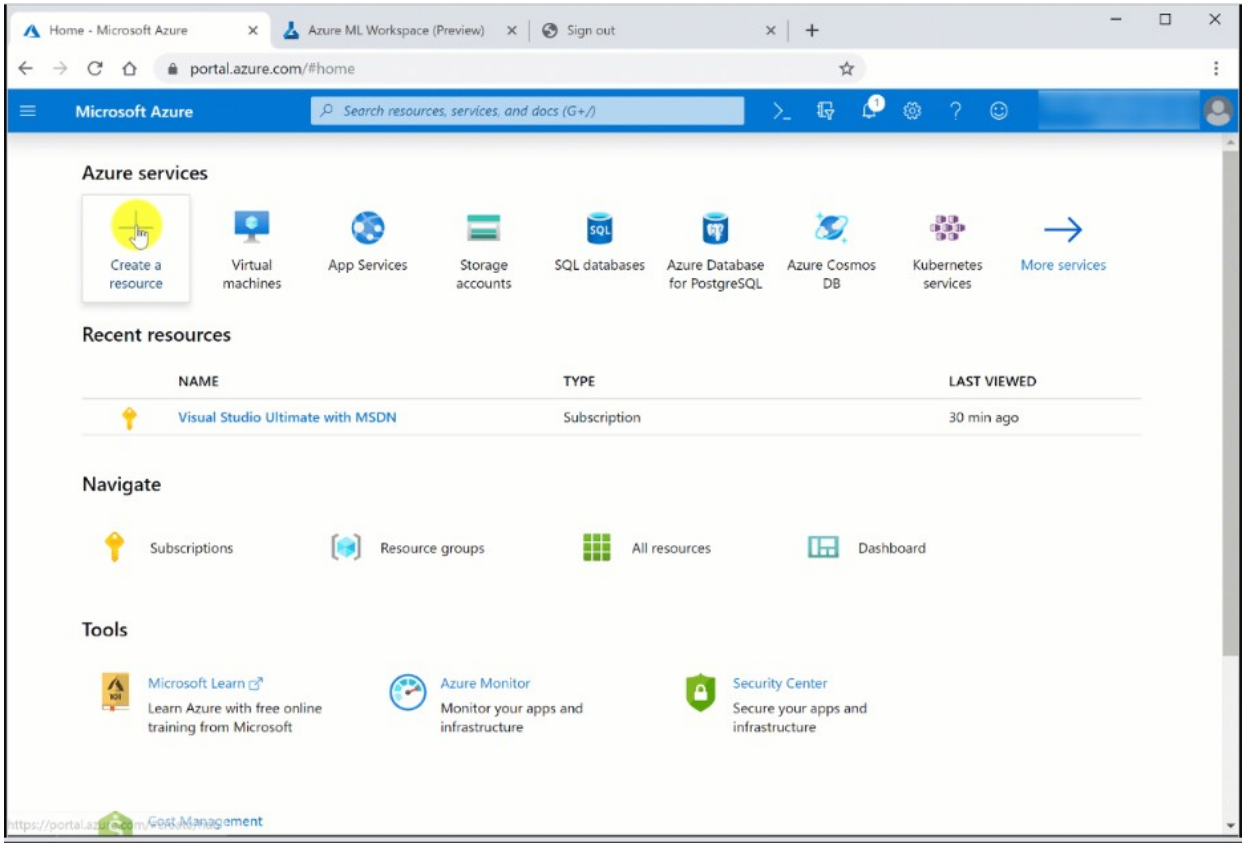


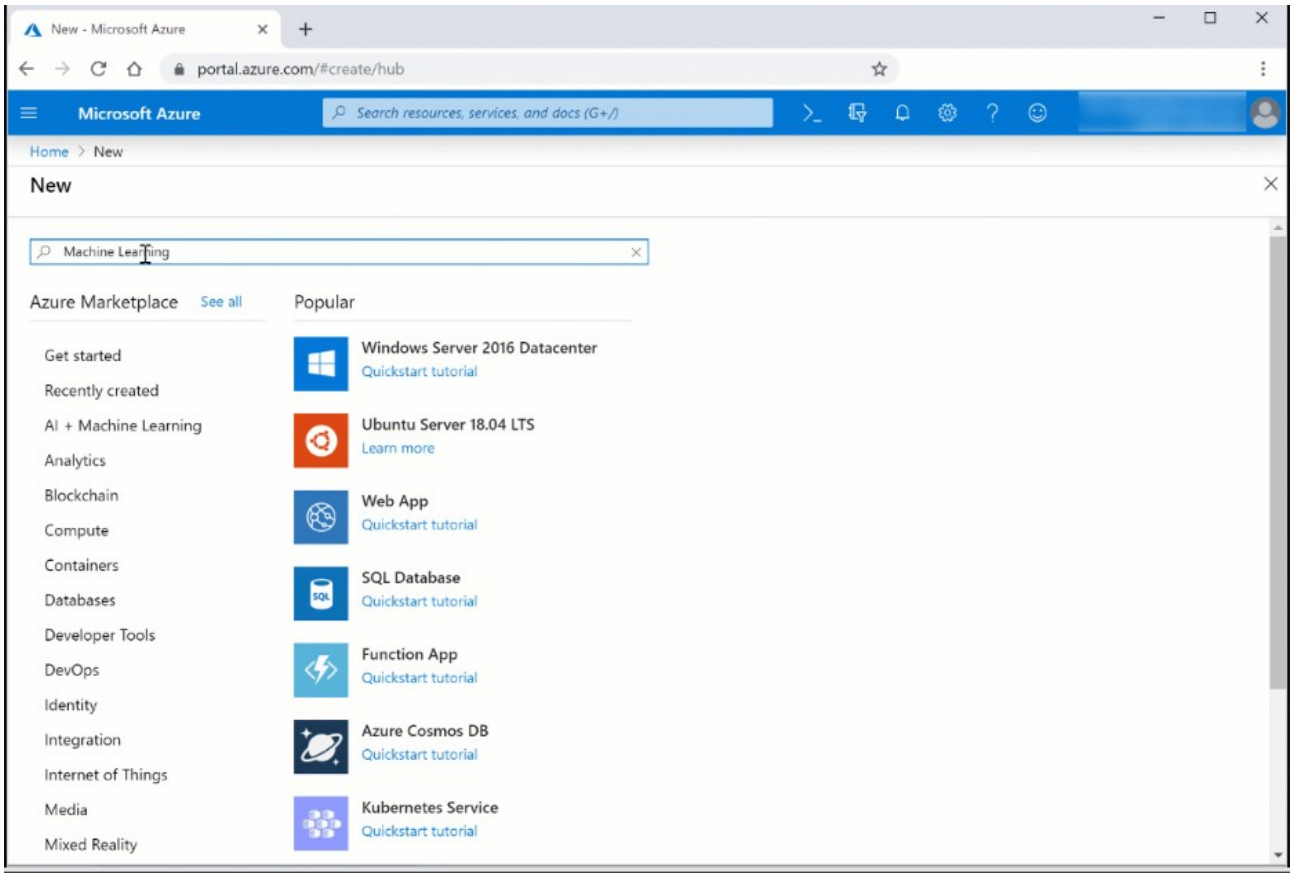
The environment, compute target and training script together form the run configuration: the full specification of a training run.

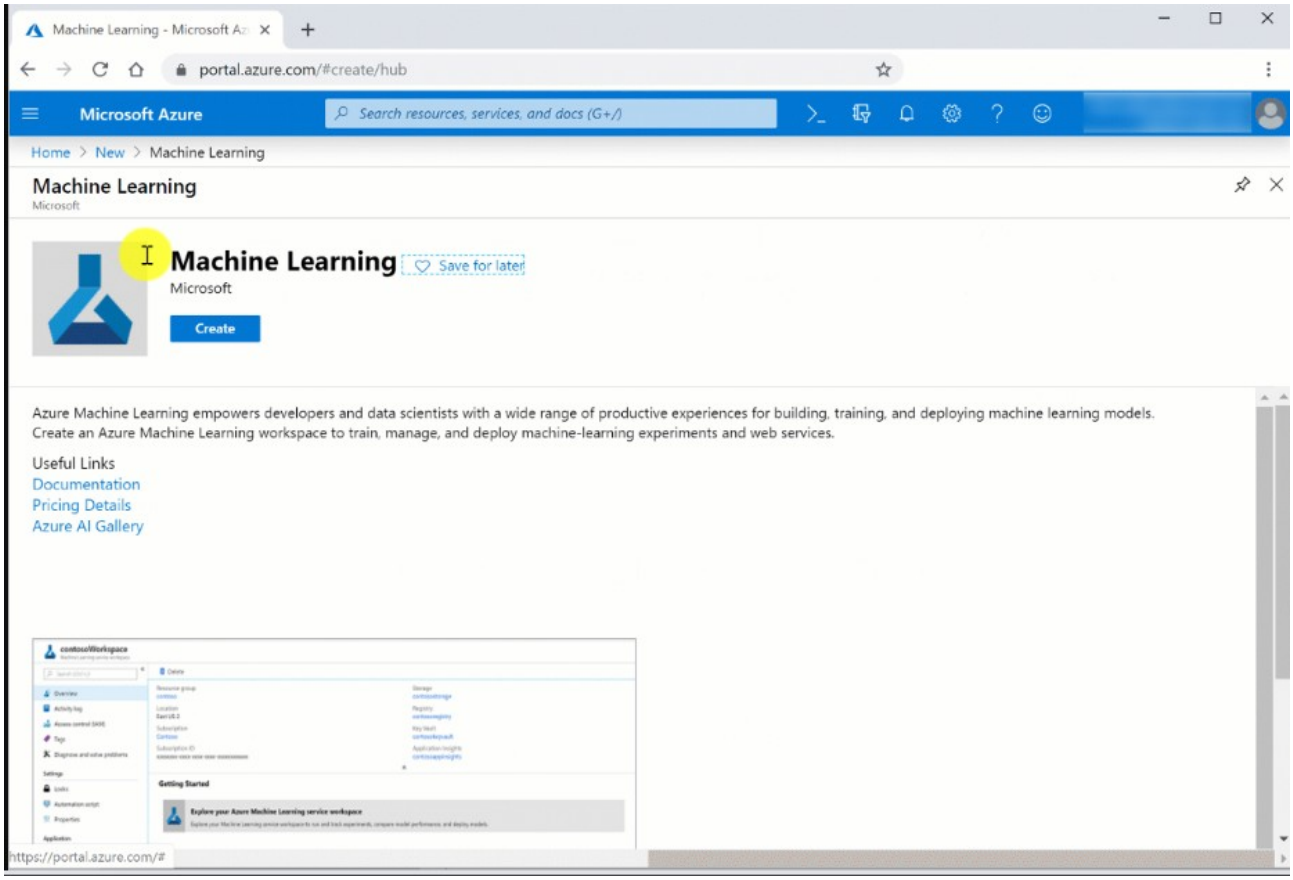
Create a Workspace

*T*he workspace is the top-level resource for Azure Machine Learning, providing a centralized place to work with all the artifacts you create when you use Azure Machine Learning. The workspace keeps a history of all training runs, including logs, metrics, output, and a snapshot of your scripts.

- Sign in to the Azure portal by using the credentials for your Azure subscription.
- In the upper-left corner of Azure portal, select **+ Create a resource**.


Home - Microsoft Azure
Azure ML Workspace (Preview)
Sign out
portal.azure.com/#home
Microsoft Azure
Search resources, services, and docs (G+/)
Azure services
Create a resource
Virtual machines
App Services
Storage accounts
SQL databases
Azure Database for PostgreSQL
Azure Cosmos DB
Kubernetes services
More services
Recent resources
NAME
TYPE
LAST VIEWED
Visual Studio Ultimate with MSDN
Subscription
30 min ago
Navigate
Subscriptions
Resource groups
All resources
Dashboard
Tools
Microsoft Learn
Learn Azure with free online training from Microsoft
Azure Monitor
Monitor your apps and infrastructure
Security Center
Secure your apps and infrastructure
Cost Management


New - Microsoft Azure
portal.azure.com/#create/hub
Microsoft Azure
Search resources, services, and docs (G+/)
Home > New
New
Machine Learning
Azure Marketplace
See all
Get started
Recently created
AI + Machine Learning
Analytics
Blockchain
Compute
Containers
Databases
Developer Tools
DevOps
Identity
Integration
Internet of Things
Media
Mixed Reality
Popular
Windows Server 2016 Datacenter
Quickstart tutorial
Ubuntu Server 18.04 LTS
Learn more
Web App
Quickstart tutorial
SQL Database
Quickstart tutorial
Function App
Quickstart tutorial
Azure Cosmos DB
Quickstart tutorial
Kubernetes Service
Quickstart tutorial



- Use the search bar to find **Machine Learning**.
- Select **Machine Learning**.
- In the **Machine Learning** pane, select **Create** to begin.
- Provide the following information to configure your new workspace

Field	Description
Workspace name	Enter a unique name that identifies your workspace. In this example, we use **docs-ws**. Names must be unique across the resource group. Use a name that's easy to recall and to differentiate from workspaces created by others. The workspace name is case-insensitive.
Subscription	Select the Azure subscription that you want to use.
Resource group	Use an existing resource group in your subscription or enter a name to create a new resource group. A resource group holds related resources for an Azure solution. In this example, we use **docs-aml**. You need *contributor* or *owner* role to use an existing resource group. For more information about access, see Manage access to an Azure Machine Learning workspace.
Region	Select the Azure region closest to your users and the data resources to create your workspace.

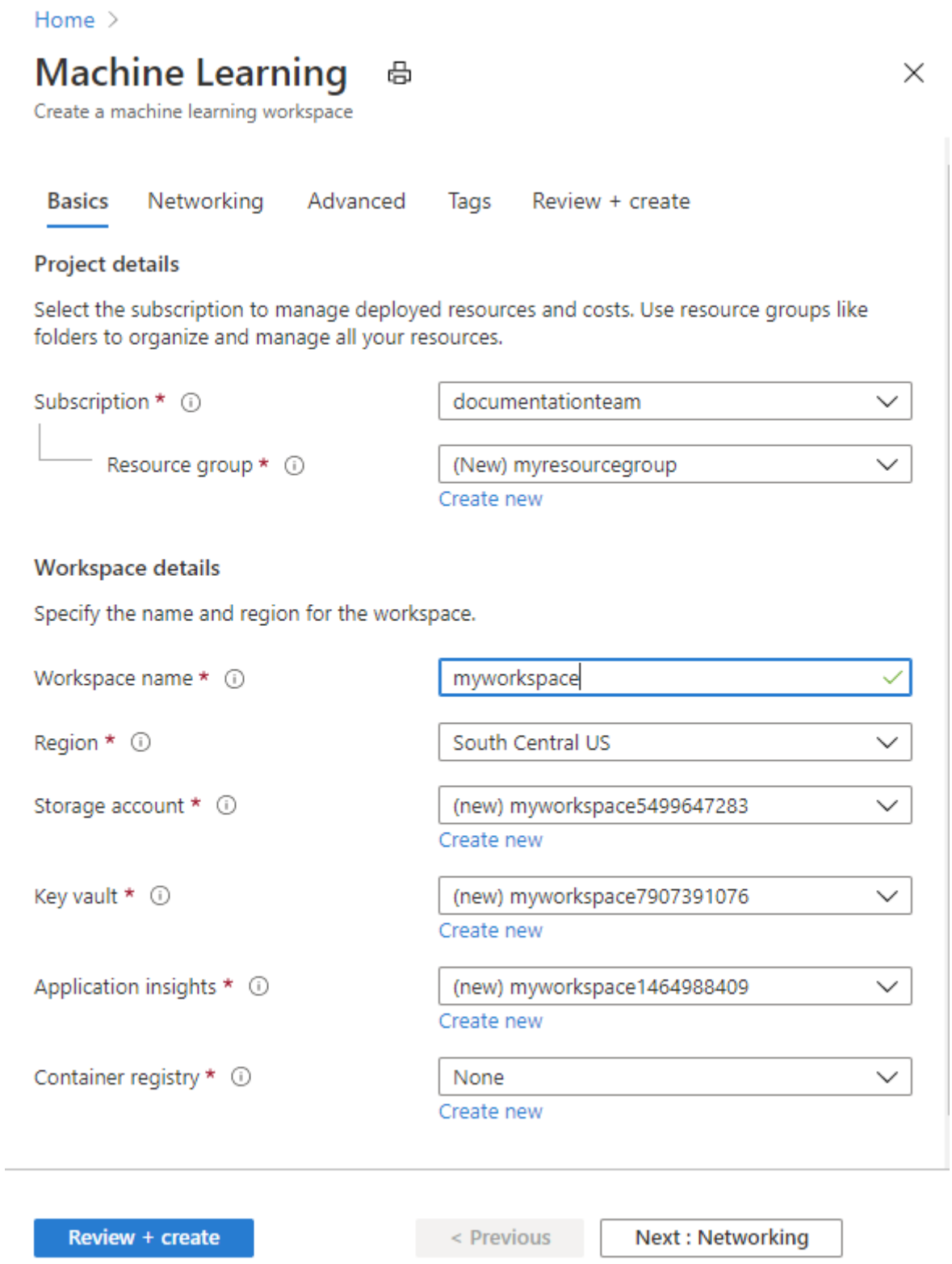
Field	Description
Storage account	The default storage account for the workspace. By default, a new one is created.
Key Vault	The Azure Key Vault used by the workspace. By default, a new one is created.
Application Insights	The application insights instance for the workspace. By default, a new one is created.
Container Registry	The Azure Container Registry for the workspace. By default, a new one is *not* initially created for the workspace. Instead, it is created once you need it when creating a Docker image during training or deployment.

Define the Environment:

Azure Machine Learning environments are an encapsulation of the environment where your machine learning training happens. They specify the Python packages, environment variables, and software settings around your training and scoring scripts.

They also specify run times (Python, Spark, or Docker).

The environments are managed and versioned entities within your Machine Learning workspace that enable reproducible, auditable, and portable machine learning workflows across a variety of compute targets.



- • When you're finished configuring the workspace, select **Review + Creat**
- Review the settings and make any additional changes or corrections. When you're satisfied with the settings, select **Create**.

It can take several minutes to create your workspace in the cloud.

- To view the new workspace, select Go to resource.

Run Jupyter Notebooks in your workspace

See how you can:

- Create Jupyter Notebooks in your workspace
- Run an experiment from a notebook
- Change the notebook environment
- Find details of the compute instances used to run your notebooks

Prerequisites

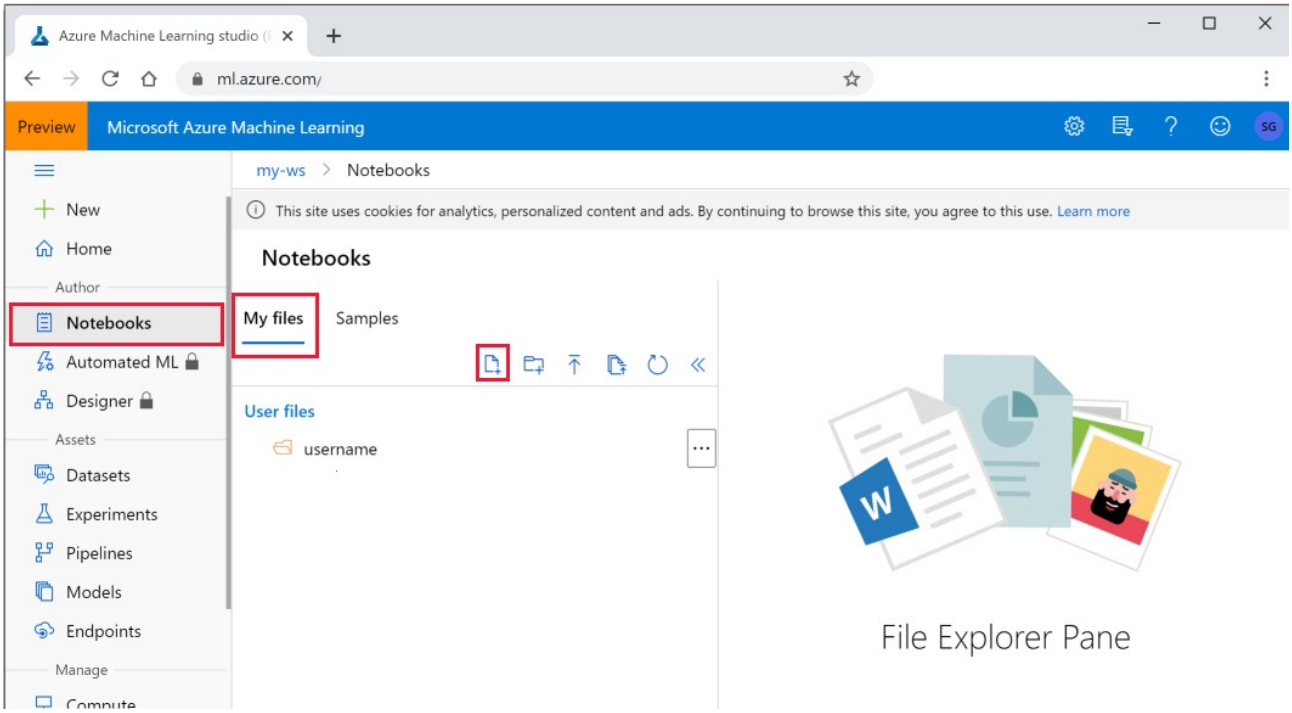
- An Azure subscription. If you don't have an Azure subscription, create a free account before you begin.
- A Machine Learning workspace.

Create notebooks

In your Azure Machine Learning workspace, create a new Jupyter notebook and start working. The newly created notebook is stored in the default workspace storage. This notebook can be shared with anyone with access to the workspace.

To create a new notebook:

1. Open your workspace in Azure Machine Learning studio.
2. On the left side, select **Notebooks**.
3. Select the **Create new file** icon above the list **User files** in the **My files** section.



4. Name the file.
5. For Jupyter Notebook Files, select **Notebook** as the file type.

6. Select a file directory.
7. Select **Create**.

You can create text files as well. Select **Text** as the file type and add the extension to the name (for example, myfile.py or myfile.txt)

You can also upload folders and files, including notebooks, with the tools at the top of the Notebooks page.

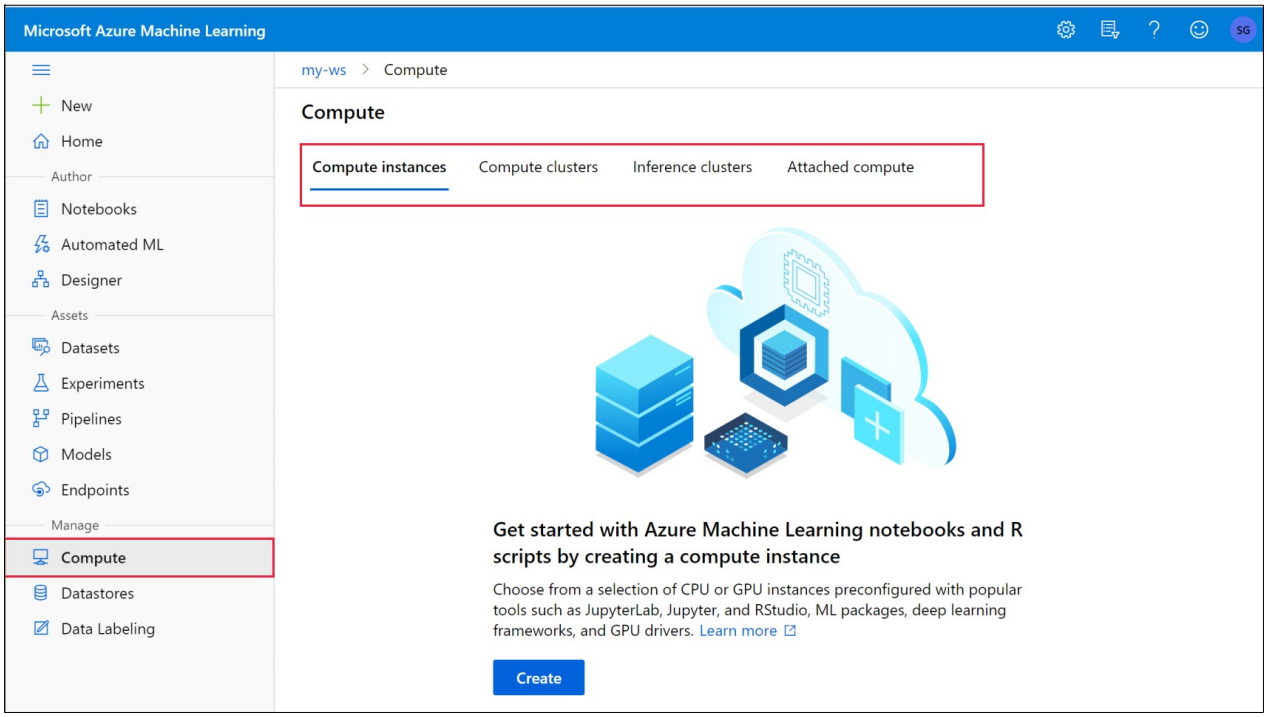
Create and manage an Azure Machine Learning compute instance

Compute instances can run jobs securely in a virtual network environment, without requiring enterprises to open up SSH ports. The job executes in a containerized environment and packages your model dependencies in a Docker container.

View compute targets

To see all compute targets for your workspace, use the following steps:

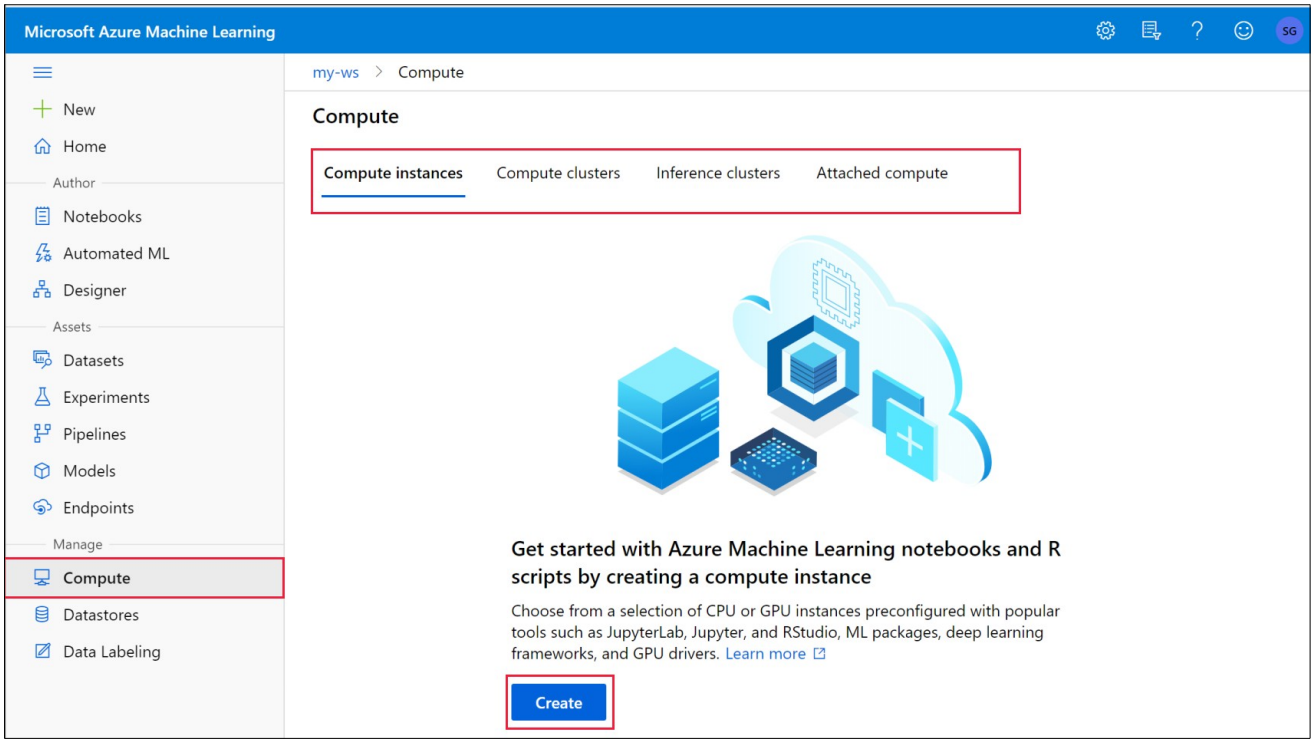
1. Navigate to Azure Machine Learning studio.
2. Under **Manage**, select **Compute**.
3. Select tabs at the top to show each type of compute target.



Create compute target

Follow the previous steps to view the list of compute targets. Then use these steps to create a compute target:

1.Select the tab at the top corresponding to the type of compute you will create.
2. If you have no compute targets, select **Create** in the middle of the page.



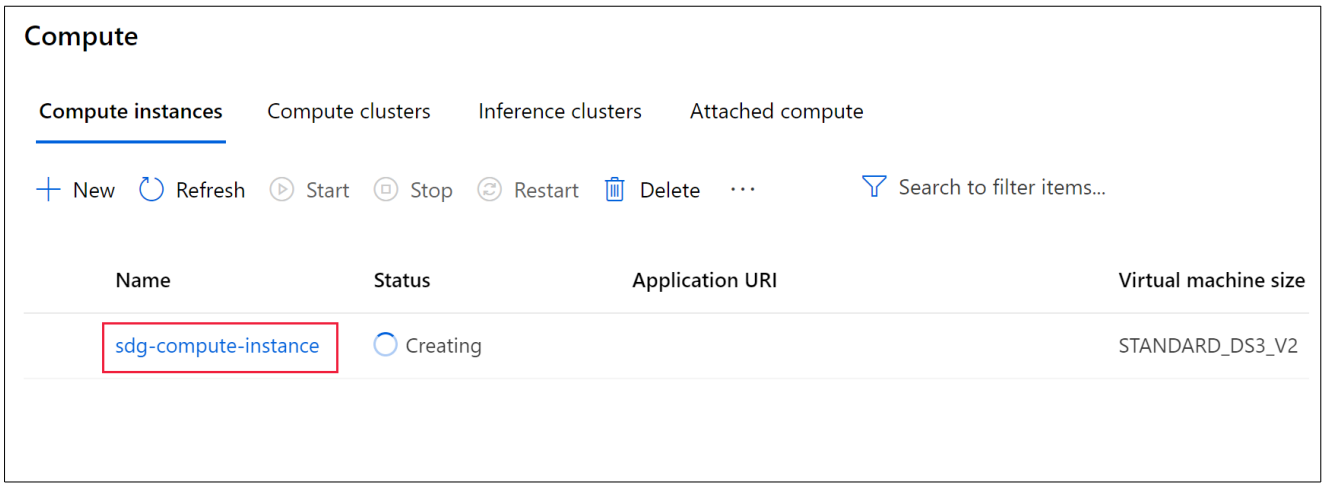
3. If you see a list of compute resources, select **+New** above the list.

4.Fill out the form for your compute type:

- Compute instance
- Compute clusters
- Inference clusters
- Attached compute

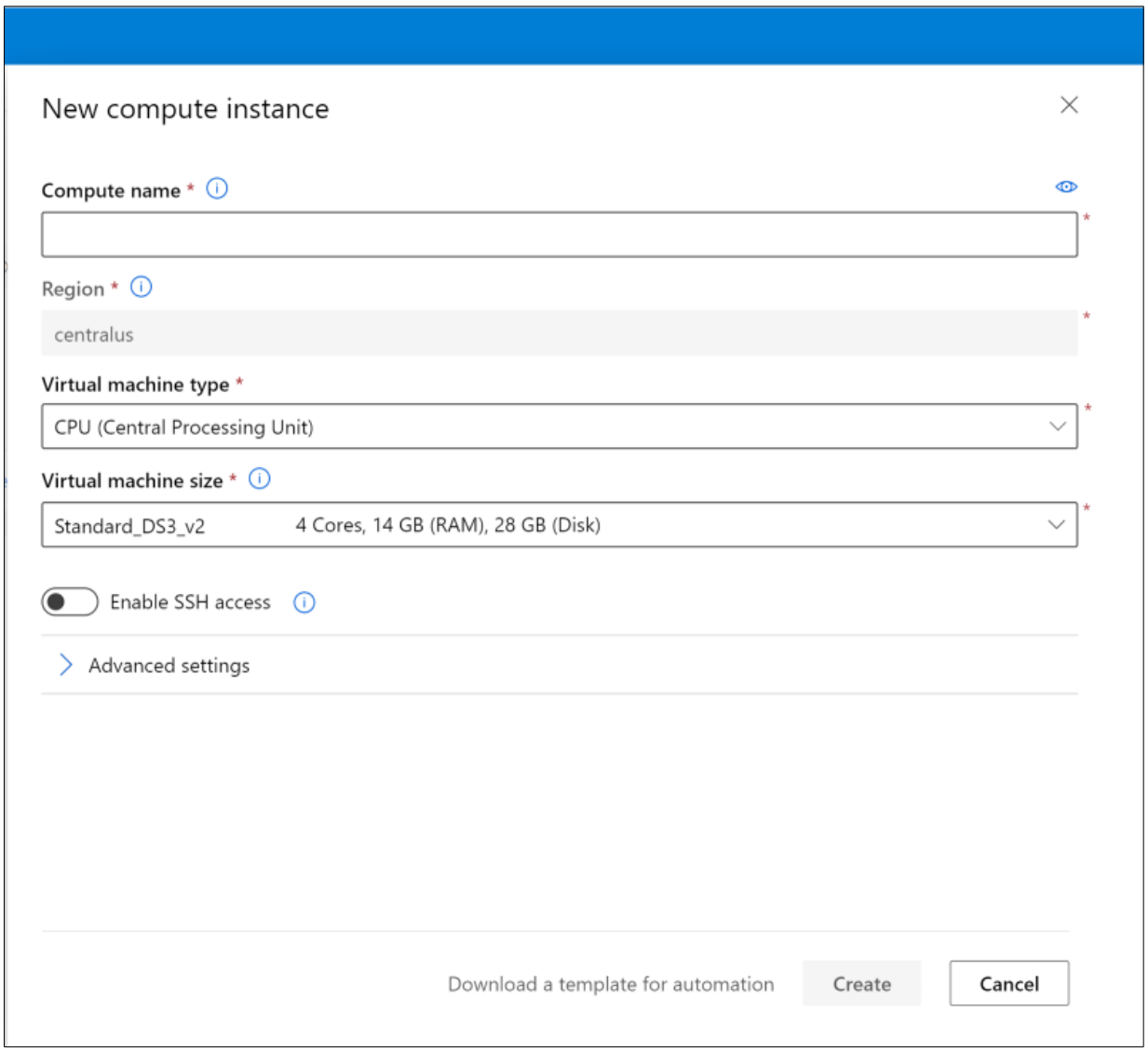
1. Select **Create**.

2.View the status of the create operation by selecting the compute target from the list:



Compute instance

Use the steps above to create the compute instance. Then fill out the form as follows:



COMPUTE INSTANCE

Field	Description
Compute name	• Name is required and must be between 3 to 24 characters long. • Valid characters are upper and lower case letters, digits, and the - character. • Name must start with a letter • Name needs to be unique across all existing computes within an Azure region. You will see an alert if the name you choose is not unique • If - character is used, then it needs to be followed by at least one letter later in the name
Virtual machine type	Choose CPU or GPU. This type cannot be changed after creation
Virtual machine size	Supported virtual machine sizes might be restricted in your region. Check the availability list
Enable/disable SSH access	SSH access is disabled by default. SSH access cannot be. changed after creation. Make sure to enable access if you plan to debug interactively with VS Code Remote
Advanced settings	Optional. Configure a virtual network. Specify the **Resource group**, **Virtual network**, and **Subnet** to create the compute instance inside an Azure Virtual Network (vnet). For more information, see these network requirements for vnet.

Train a model by using a custom Docker image

Now we will learn how to use a custom Docker image when you're training models with Azure Machine Learning.

Set up a training experiment

In this section, you set up your training experiment by initializing a workspace, defining your environment, and configuring a compute target.

The Azure Machine Learning workspace is the top-level resource for the service. It gives you a centralized place to work with all the artifacts that you create. In the Python SDK, you can access the workspace artifacts by creating a Workspace object.

Using following commands:

from azureml.core import Workspace

ws = Workspace.from_config()

Define the compute

Creation of compute takes approximately 5 minutes. If the AmlCompute with that name is already in your workspace the code will skip the creation process.

```
from azureml.core.compute import ComputeTarget, AmlCompute
from azureml.core.compute_target import ComputeTargetException
Choose a name for your cluster.
cluster_name = "define-name-compute"
try:
      compute_target = ComputeTarget(workspace=ws, name=cluster_name)
      print('Found existing compute target.')
except ComputeTargetException:
      print('Creating a new compute target...')
      compute_config = AmlCompute.provisioning_configuration(vm_size='assign-vm-size',
                                                             max_nodes=4)
      #Create the cluster.
      compute_target = ComputeTarget.create(ws, cluster_name, compute_config)
      compute_target.wait_for_completion(show_output=True)
#Use get_status() to get a detailed status for the current AmlCompute.
print(compute_target.get_status().serialize())
```

Use an image with the Azure Machine Learning SDK

Following are the steps to define an environement and to use a private container registry:

To use an image stored in the Azure Container Registry for your workspace, or a container registry that is publicly accessible, set the following Environment attributes:

```
from azureml.core.environment import Environment
# Create the environment
myenv = Environment(name="provide-env-name")
# Enable Docker and reference an image
myenv.docker.enabled = True
```

When you're using your custom Docker image, you might already have your Python environment properly set up. In that case, set the user_managed_dependencies flag to True to use your custom image's built-in Python environment.

```
myenv.python.user_managed_dependencies = True
```

Set the container registry information

To use an image from a private container registry that isn't in your workspace, use docker.base_image_registry to specify the address of the repository and a username and password:

```
myenv.docker.base_image_registry.address = "registry-name"
myenv.docker.base_image_registry.username = "Azure-username"
myenv.docker.base_image_registry.password = "**********************"
myenv.docker.base_image = "image-name-with-tag"
# Define the packages needed by the model and scripts
from azureml.core.conda_dependencies import CondaDependencies
conda_dep = CondaDependencies()
# you must list azureml-defaults as a pip dependency
conda_dep.add_pip_package("azureml-defaults")
myenv.python.conda_dependencies=conda_dep
```

Connect to the defualt storage services on Azure

```
#To get the workspace's default datastore, use this line:
datastore = ws.get_default_datastore()
'''Now let's upload the 2 files(train data and test data) into the default datastore under a path named main_dataset:'''
ds.upload_files(['./traincsv',"./test.csv"], target_path='main_dataset', overwrite=True)
#Downloading the files to compute as a FileDataset object.
from azureml.core import Dataset
ds_paths = [(ds, 'main_dataset/')]
dataset = Dataset.File.from_files(path = ds_paths)
'''Following arguments will be passed to train file.'''
from uuid import uuid4
script_arguments = ['--data-folder',
dataset.as_named_input('main_dataset').as_mount('/tmp/{}'.format(uuid4())),'--Target',
"species","--Train_data","iris.csv","--Test_data","iris_test.csv",'--token',"<please provide your token>"]
```

Submit the experiment.

We are using the following code run train.py file in the container using the compute and environemnt we created above.

```
from azureml.core import ScriptRunConfig
src = ScriptRunConfig(source_directory='./',
                      script='train.py',
                      arguments=script_arguments,
                      compute_target=compute_target,
                      environment= myenv)
from azureml.core import Experiment
run = Experiment(ws,'experiment-name').submit(src)
run.wait_for_completion(show_output=True)
#Get all metris logged in the run
"""Train file ouput will be stored under metrics, to extract the json stored under metrics we are
using the below command.
"""
run.get_metrics()
metrics = run.get_metrics()
```

train.py

#Create a training script

```
# Importing all required libraries
import sourcedefender
import os
import argparse
import sys
import json
import traceback
os.chdir("/automl")
from bi.master import
DataPreprocessingAutoML,FeatureEngineeringAutoML,FeatureSelection,MadvisorAutoMl
from sklearn import datasets
import pandas as pd
import numpy as np
from azureml.core import Run
#reading the arguments from outside the container.
parser = argparse.ArgumentParser()
parser.add_argument('--data-folder', type=str,dest='data_folder', help='data folder')
parser.add_argument('--Target', type=str, dest='Target',help='Target')
parser.add_argument('--Train_data', type=str, dest='Train_data',help='Train_data')
parser.add_argument('--Test_data', type=str, dest='Test_data',help='Test_data')
parser.add_argument('--token', type=str, dest='token',help='token')
args = parser.parse_args()
run = Run.get_context()
def train(): #running training and prediction inside the train function
      print('Starting the training.')
      data = pd.read_csv(os.path.join(args.data_folder, args.Train_data))
      test_data = pd.read_csv(os.path.join(args.data_folder, args.Test_data))
      Target = args.Target
      token =  args.token
      auto_obj  = MadvisorAutoMl(target=Target,\
                                 train_data=data,\
                                 test_data=test_data,\
                                  token=token
                                 )
```

```
    output_of_train=auto_obj.fit()
    if output_of_train == "Please contact for licence":
        return "Please contact for token or Update token"
    else:
        train_slug = output_of_train['model result']['model_slug']
        auto_obj.get_model_summary()
        print("-"*100 +'\n',auto_obj.get_model_summary())
        print('Training completed.\nPrediction started.')
        output_of_test=auto_obj.fit()
        prediction=auto_obj.get_score_dataframe()
        run.log('prediction', prediction)
        print('Prediction complete.')
        print("*" * 100)
        return prediction
if __name__ == '__main__':
    train_result =  train()
    print(train_result)
    # A zero exit code causes the job to be marked a Succeeded.
    sys.exit(0)
```