MLOps

Training In The Sky

Machine Learning Operations

Damian Konrad Kowalczyk, PhD
Applied Scientist, Cloud + AI Division
Microsoft Corporation

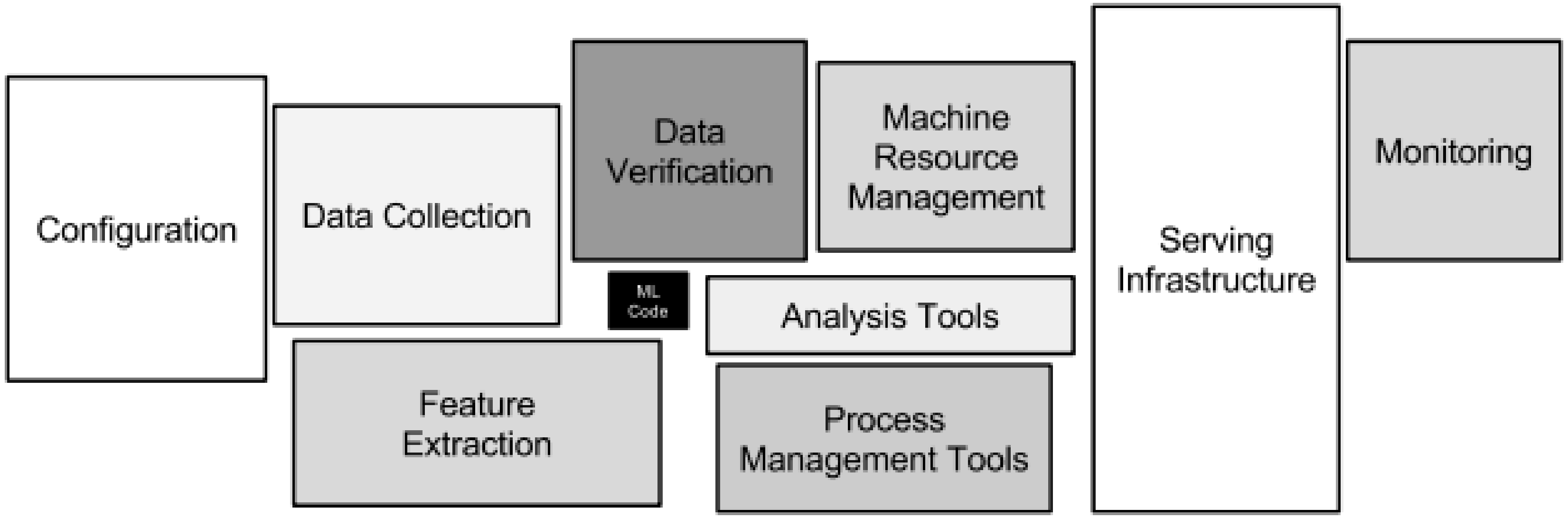
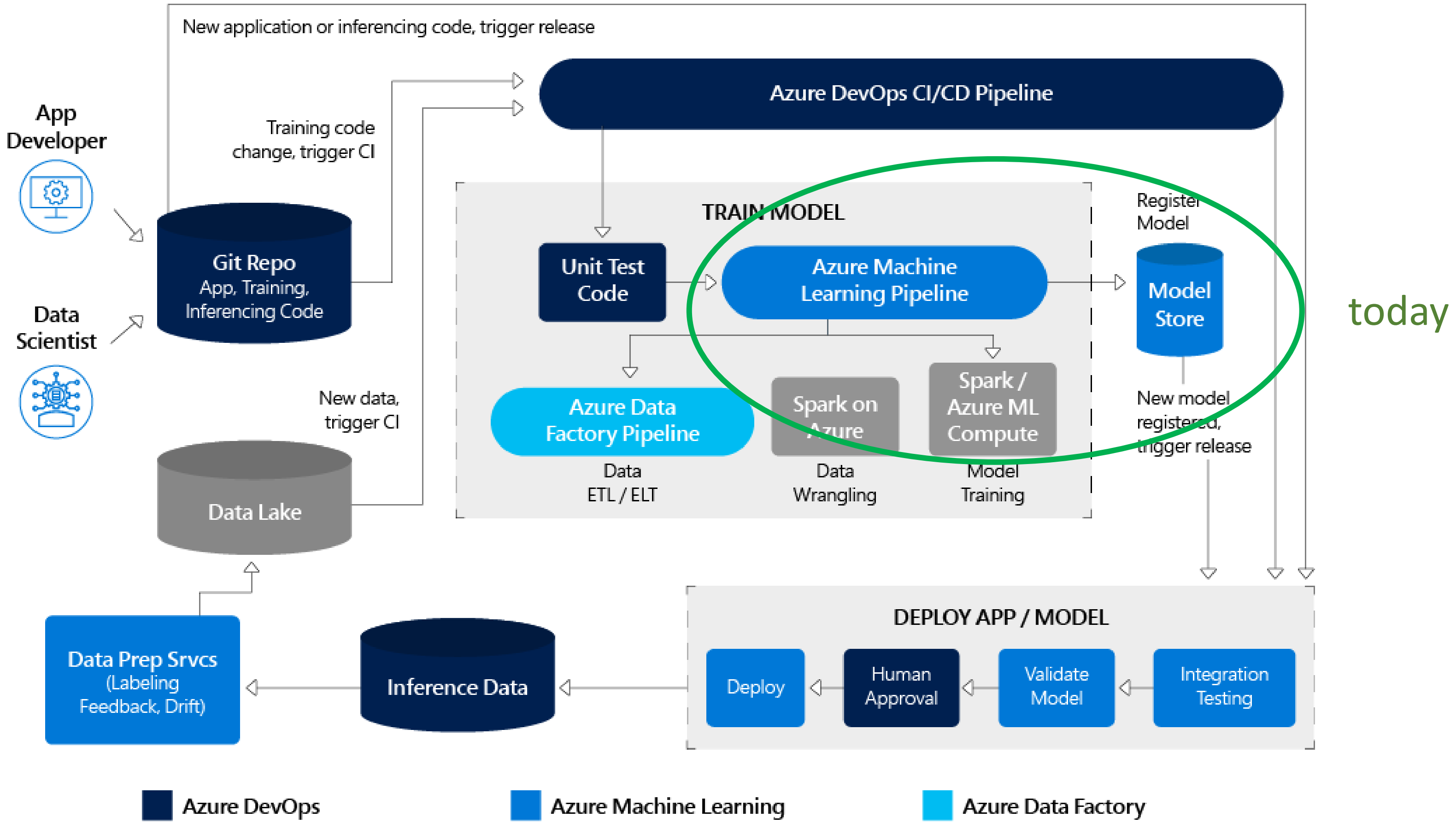


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

source: Hidden Technical Debt in Machine Learning Systems (nips.cc)

science meets engineering: MLOps on Azure



Damian Kowalczyk, Microsoft Corporation

Today

Introduction to Azure Machine Learning

- Provision an Azure Machine Learning workspace.
- Use tools and interfaces to work with Azure Machine Learning.
- Run code-based experiments in an Azure Machine Learning workspace.

Train a machine learning model with Azure Machine Learning

- Use a ScriptRunConfig to run a model training script as an Azure Machine Learning experiment.
- Create reusable, parameterized training scripts.
- Register trained models.

Part of: Build and operate machine learning solutions with Azure Machine Learning

Azure Machine Learning

- Scalable on-demand compute for machine learning workloads.

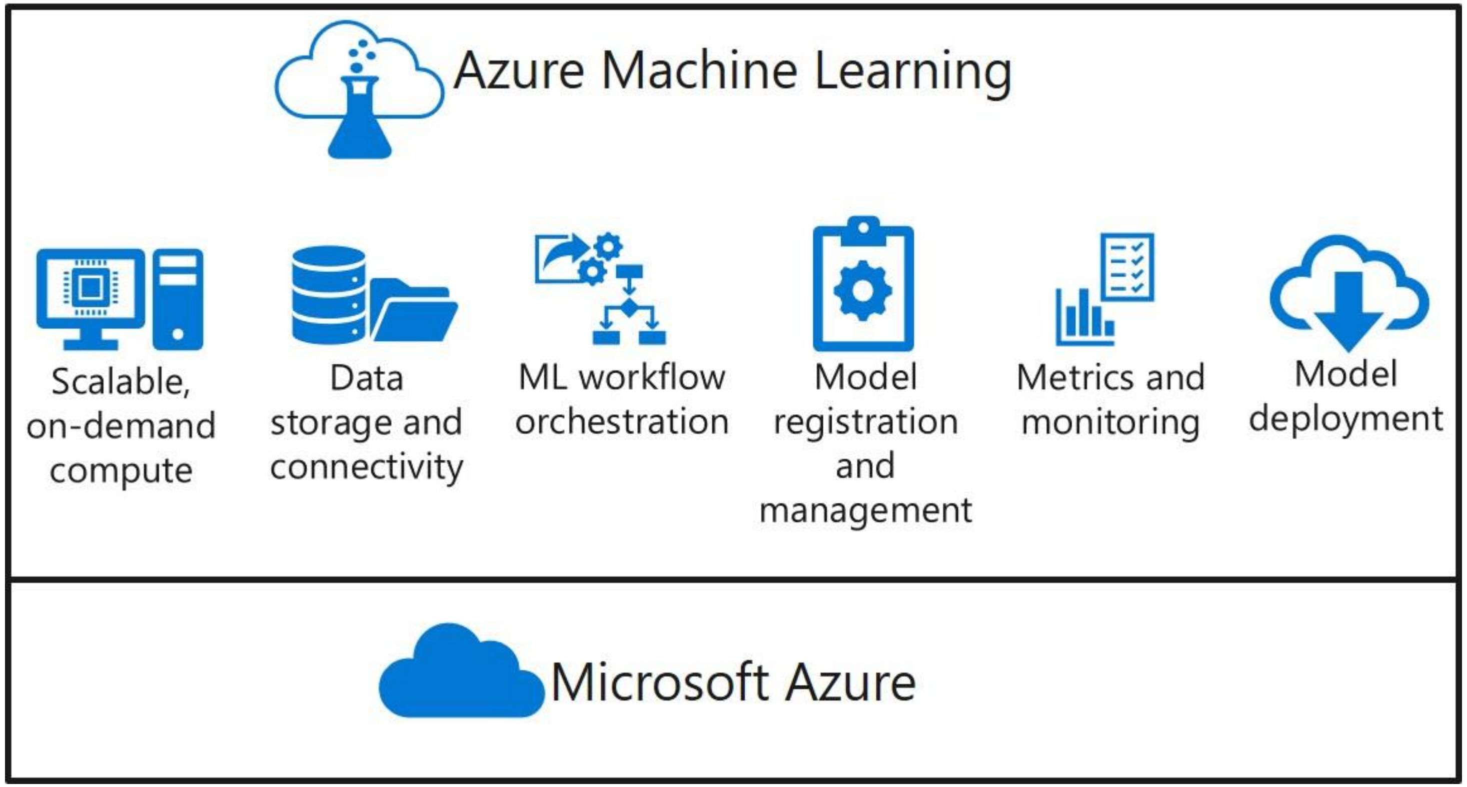
- Data storage and connectivity to ingest data from a wide range sources.

- Machine learning workflow orchestration to automate model training, deployment, and management processes.

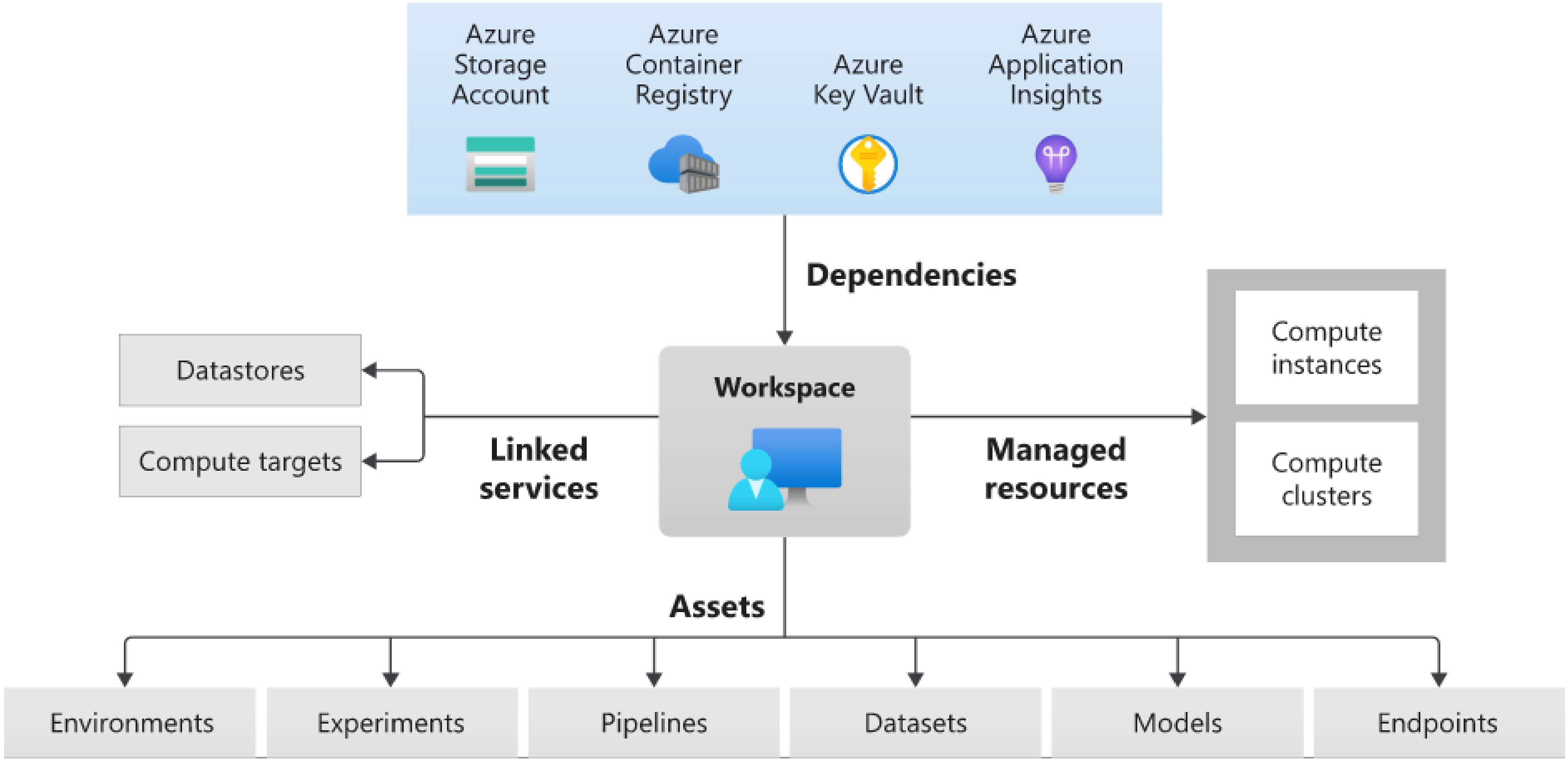
- Model registration and management, so you can track multiple versions of models and the data on which they were trained.

- Metrics and monitoring for training experiments, datasets, and published services.

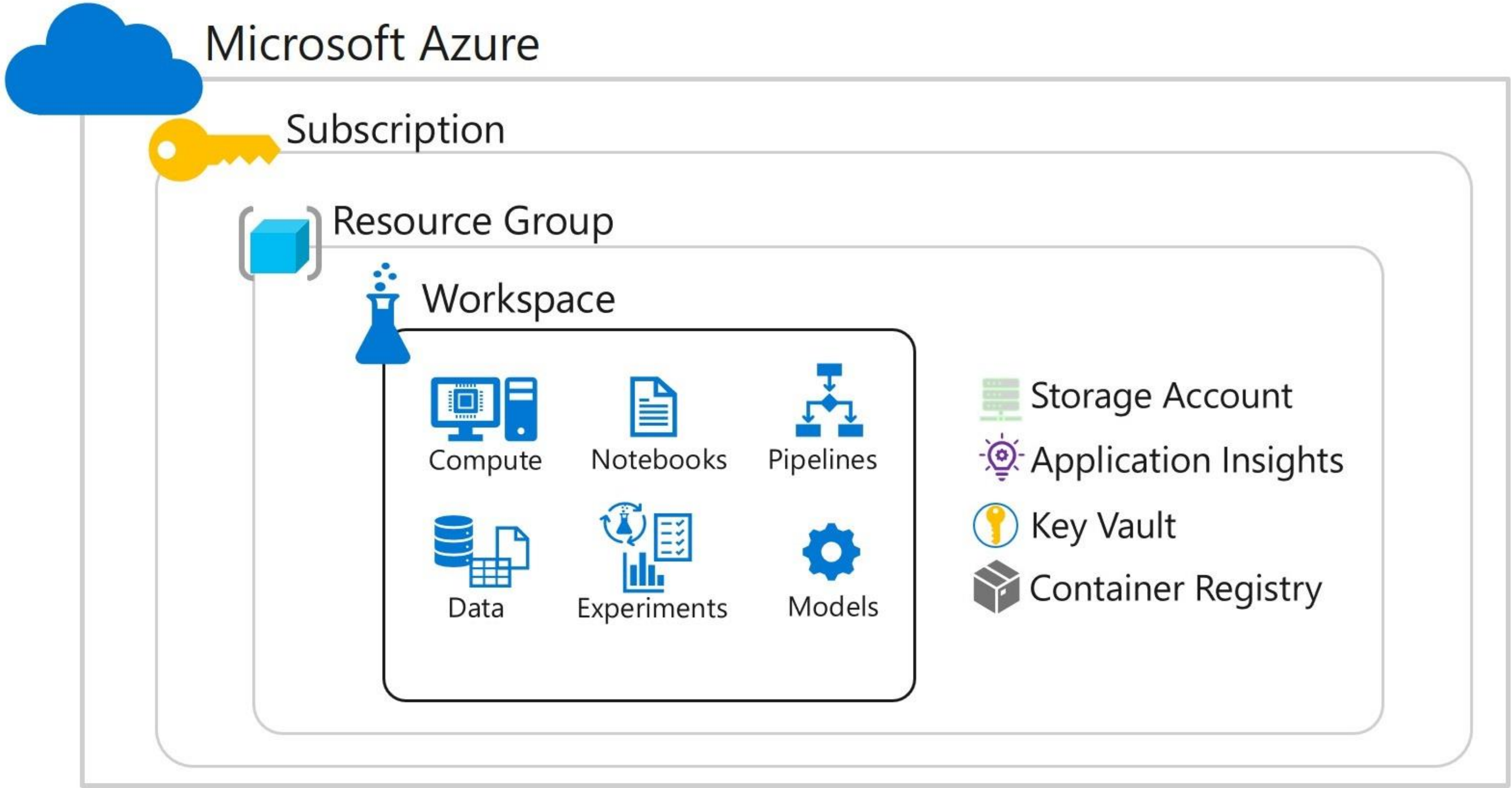
- Model deployment for real-time and batch inferencing.


Azure Machine Learning
Scalable, on-demand compute
Data storage and connectivity
ML workflow orchestration
Model registration and management
Metrics and monitoring
Model deployment
Microsoft Azure

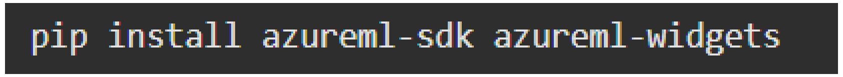
Azure ML Workspace



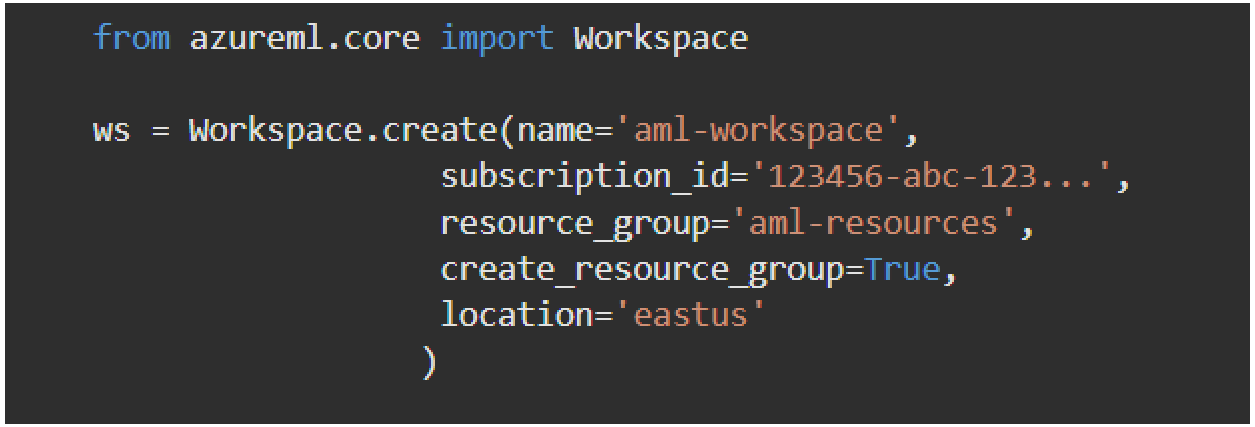
Azure ML Workspace as Azure Resources



Azure ML SDK (Python)



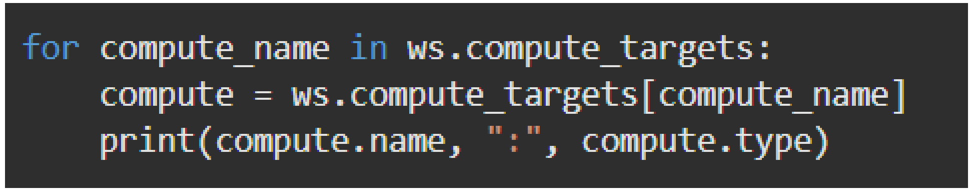
```
pip install azureml-sdk azureml-widgets
```



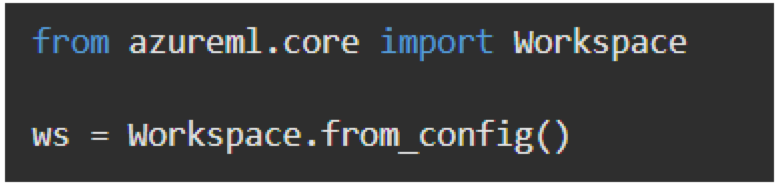
```python
from azureml.core import Workspace

ws = Workspace.create(name='aml-workspace',
                      subscription_id='123456-abc-123...',
                      resource_group='aml-resources',
                      create_resource_group=True,
                      location='eastus'
                     )
```

```python
ws.write_config(path='.azureml')
```

```json
{
"subscription_id": "1234567-abcde-890-fgh...",
"resource_group": "aml-resources",
"workspace_name": "aml-workspace"
}
```

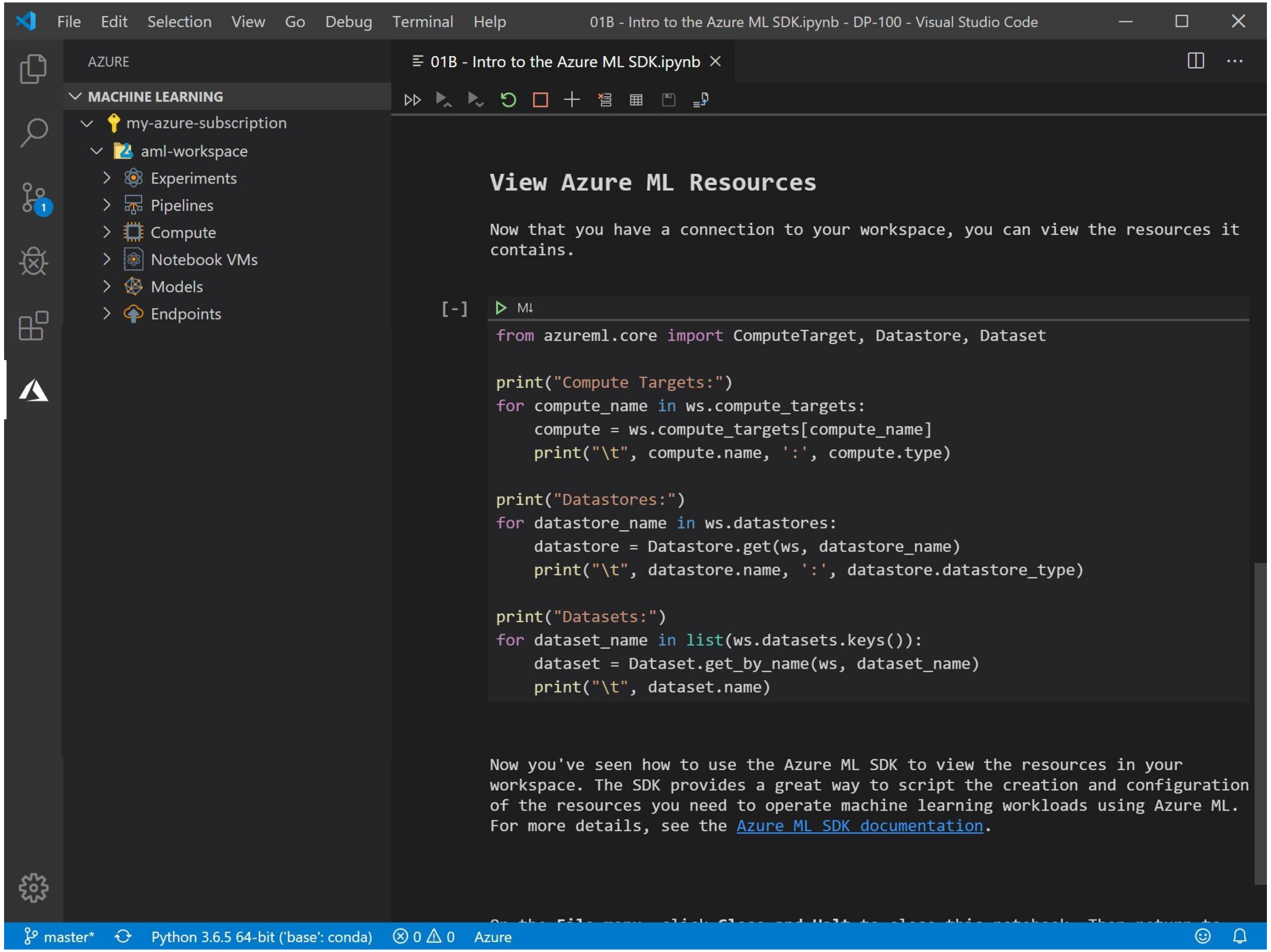


```python
for compute_name in ws.compute_targets:
    compute = ws.compute_targets[compute_name]
    print(compute.name, ":", compute.type)
```



```python
from azureml.core import Workspace

ws = Workspace.from_config()
```

Azure ML from VS Code



View Azure ML Resources

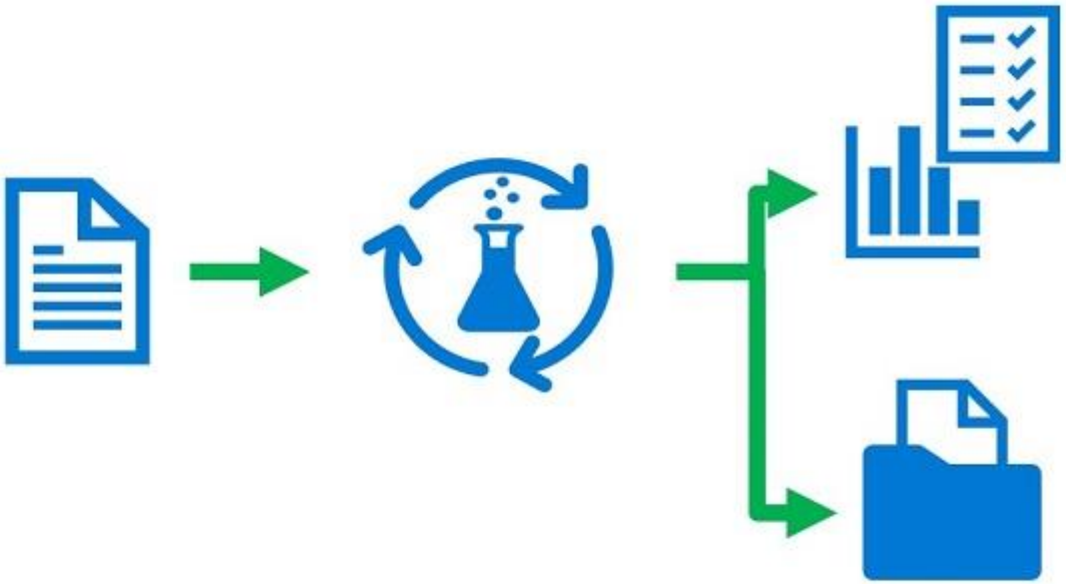
Now that you have a connection to your workspace, you can view the resources it contains.

```
from azureml.core import ComputeTarget, Datastore, Dataset

print("Compute Targets:")
for compute_name in ws.compute_targets:
    compute = ws.compute_targets[compute_name]
    print("\t", compute.name, ':', compute.type)

print("Datastores:")
for datastore_name in ws.datastores:
    datastore = Datastore.get(ws, datastore_name)
    print("\t", datastore.name, ':', datastore.datastore_type)

print("Datasets:")
for dataset_name in list(ws.datasets.keys()):
    dataset = Dataset.get_by_name(ws, dataset_name)
    print("\t", dataset.name)
```

Now you've seen how to use the Azure ML SDK to view the resources in your workspace. The SDK provides a great way to script the creation and configuration of the resources you need to operate machine learning workloads using Azure ML. For more details, see the Azure ML SDK documentation.

Damian Kowalczyk, Microsoft Corporation

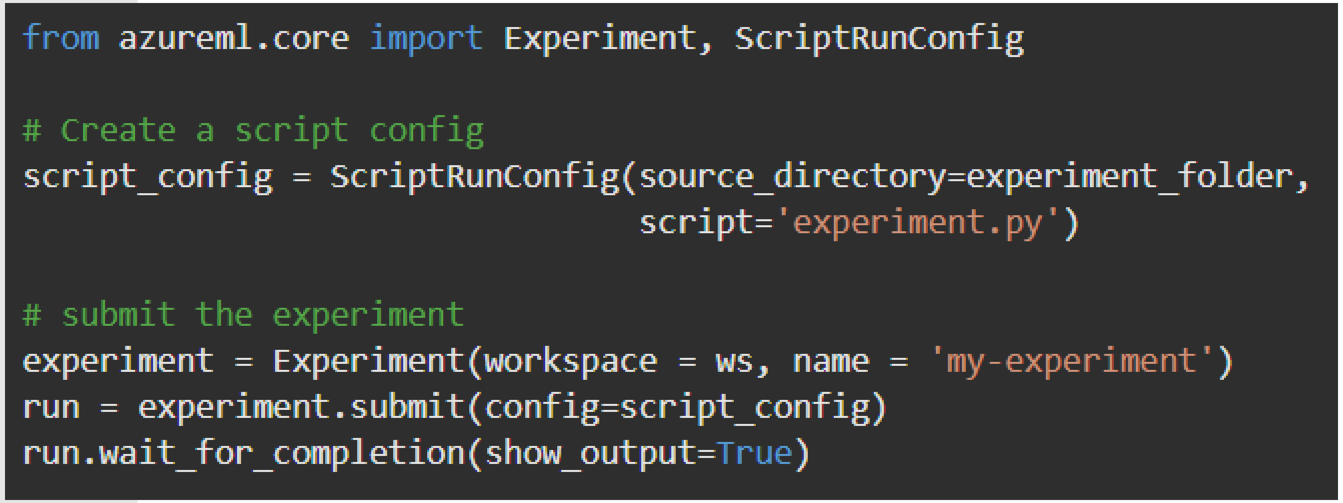
Azure Machine Learning **Experiments**



Running an Experiment: ScriptRunConfig

(experiment.py)

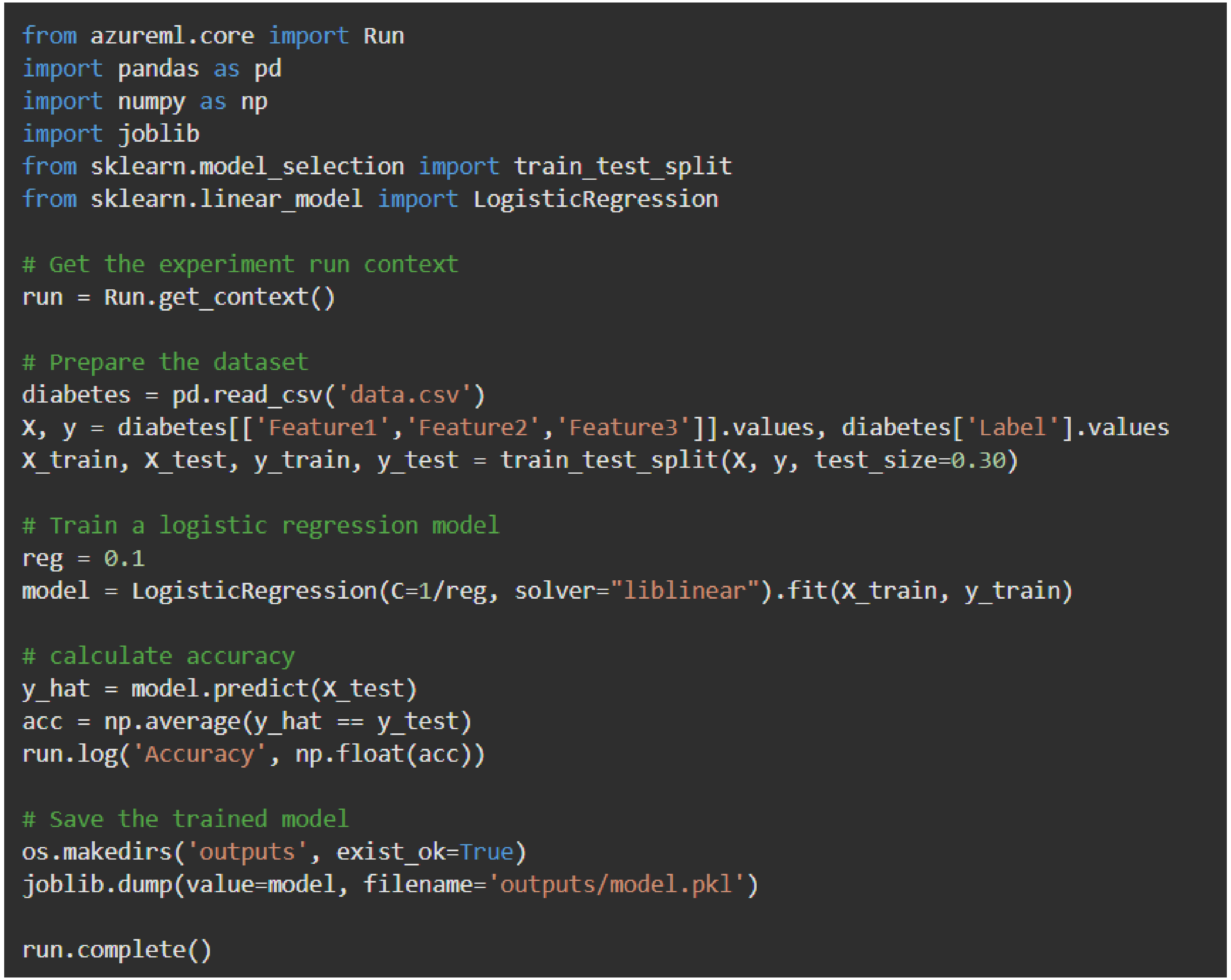
```
from azureml.core import Experiment
import pandas as pd

# Create an Azure ML experiment in your workspace
experiment = Experiment(workspace = ws, name = 'my-experiment')

# Start logging data from the experiment
run = experiment.start_logging()

# load the dataset and count the rows
data = pd.read_csv('data.csv')
row_count = (len(data))

# Log the row count
run.log('observations', row_count)

run.upload_file(name='outputs/sample.csv',
path_or_stream='./sample.csv')

# Complete the experiment
run.complete()
```



```
from azureml.core import Experiment, ScriptRunConfig

# Create a script config
script_config = ScriptRunConfig(source_directory=experiment_folder,
                                script='experiment.py')

# submit the experiment
experiment = Experiment(workspace = ws, name = 'my-experiment')
run = experiment.submit(config=script_config)
run.wait_for_completion(show_output=True)
```

Training a simple model

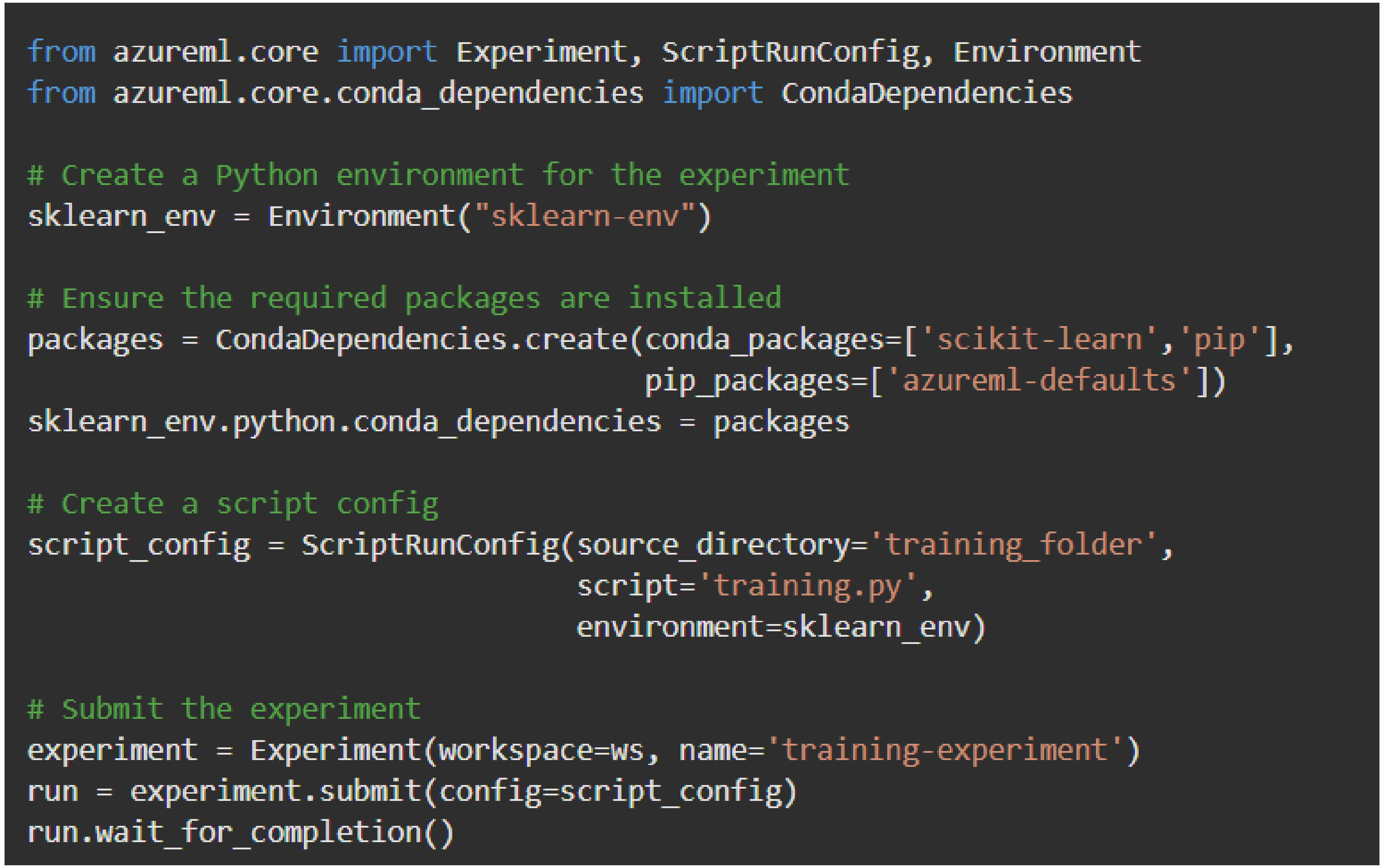
(script)



```
from azureml.core import Run
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Get the experiment run context
run = Run.get_context()

# Prepare the dataset
diabetes = pd.read_csv('data.csv')
X, y = diabetes[['Feature1','Feature2','Feature3']].values, diabetes['Label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

# Train a logistic regression model
reg = 0.1
model = LogisticRegression(C=1/reg, solver="liblinear").fit(X_train, y_train)

# calculate accuracy
y_hat = model.predict(X_test)
acc = np.average(y_hat == y_test)
run.log('Accuracy', np.float(acc))

# Save the trained model
os.makedirs('outputs', exist_ok=True)
joblib.dump(value=model, filename='outputs/model.pkl')

run.complete()
```

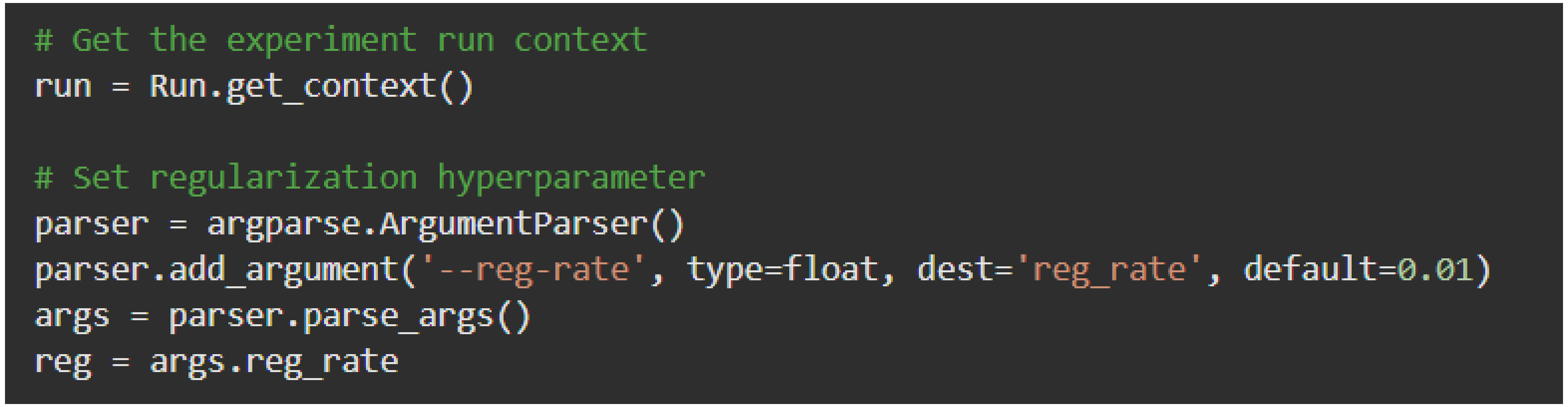
Training a simple model

(experiment)



```python
from azureml.core import Experiment, ScriptRunConfig, Environment
from azureml.core.conda_dependencies import CondaDependencies

# Create a Python environment for the experiment
sklearn_env = Environment("sklearn-env")

# Ensure the required packages are installed
packages = CondaDependencies.create(conda_packages=['scikit-learn','pip'],
                                    pip_packages=['azureml-defaults'])
sklearn_env.python.conda_dependencies = packages

# Create a script config
script_config = ScriptRunConfig(source_directory='training_folder',
                                script='training.py',
                                environment=sklearn_env)

# Submit the experiment
experiment = Experiment(workspace=ws, name='training-experiment')
run = experiment.submit(config=script_config)
run.wait_for_completion()
```

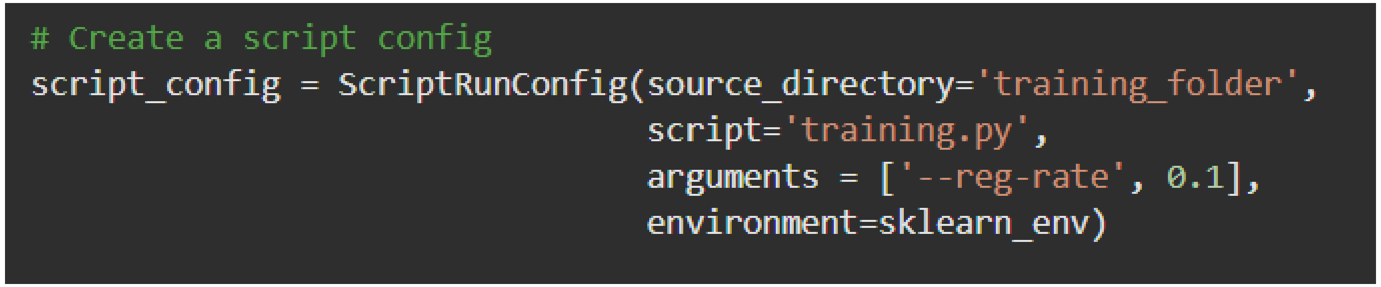
Parametrized scripts



```
# Get the experiment run context
run = Run.get_context()

# Set regularization hyperparameter
parser = argparse.ArgumentParser()
parser.add_argument('--reg-rate', type=float, dest='reg_rate', default=0.01)
args = parser.parse_args()
reg = args.reg_rate
```

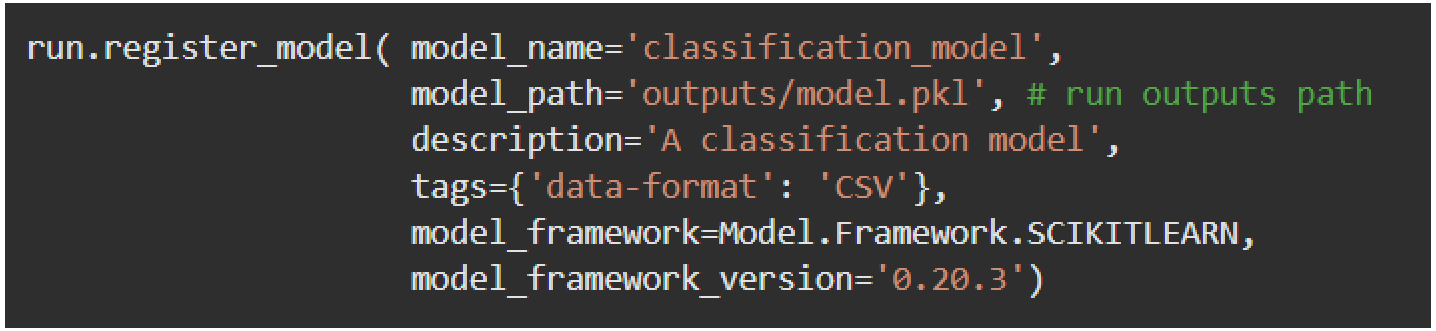
Instrument your experiment script

…and pass the argument
in the runner script



```
# Create a script config
script_config = ScriptRunConfig(source_directory='training_folder',
                                script='training.py',
                                arguments = ['--reg-rate', 0.1],
                                environment=sklearn_env)
```

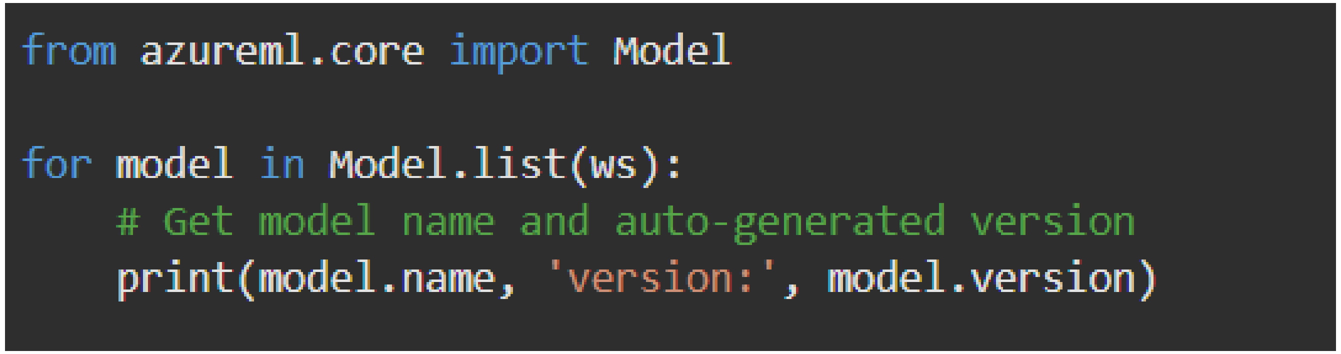
Registering models



```
run.register_model( model_name='classification_model',
                    model_path='outputs/model.pkl', # run outputs path
                    description='A classification model',
                    tags={'data-format': 'CSV'},
                    model_framework=Model.Framework.SCIKITLEARN,
                    model_framework_version='0.20.3')
```

tag and upload the model binary to AML model registry

then lookup and download registered models with the Model API



```
from azureml.core import Model

for model in Model.list(ws):
    # Get model name and auto-generated version
    print(model.name, 'version:', model.version)
```

To Do

1. Exercise - Create a workspace - Learn | Microsoft Docs
2. Exercise - Run experiments - Learn | Microsoft Docs
3. Exercise - Training and registering a model - Learn | Microsoft Docs
4. Knowledge check 1 - Learn | Microsoft Docs
5. Knowledge check 2 - Learn | Microsoft Docs

Resources

- Build and operate machine learning solutions with Azure Machine Learning

Detailed How-To Guides:

- Create workspaces in the portal - Azure Machine Learning | Microsoft Docs
- Set up Visual Studio Code extension (preview) - Azure Machine Learning | Microsoft Docs
- Connect to compute instance in Visual Studio Code (preview) - Azure Machine Learning | Microsoft Docs
- Configure a training run - Azure Machine Learning | Microsoft Docs
- Track, monitor, and analyze runs - Azure Machine Learning | Microsoft Docs

damk@dtu.dk