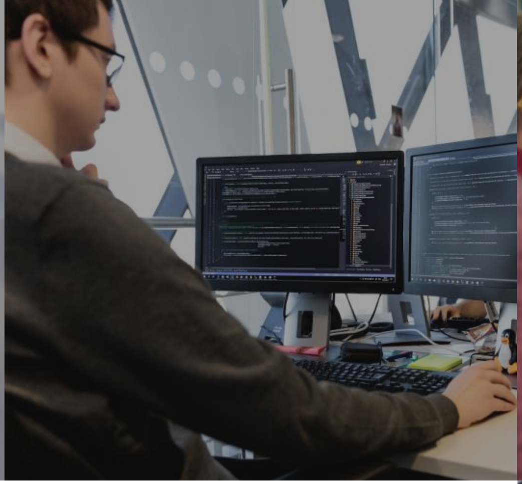
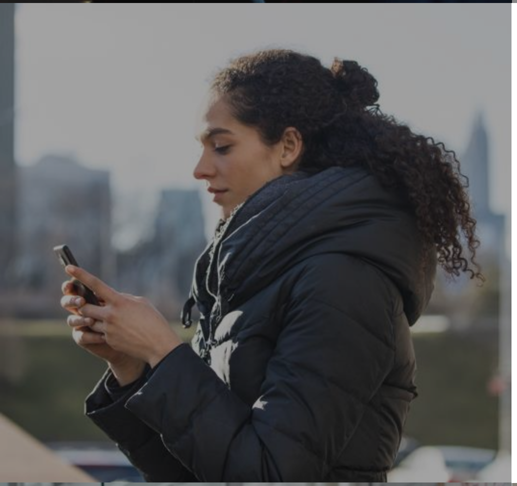
Microsoft

Module 6: Orchestrating Operations with Pipelines

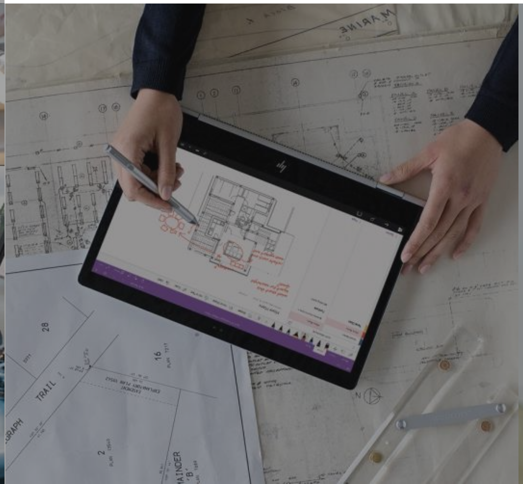
Agenda

- Introduction to Pipelines
- Publishing and Running Pipelines



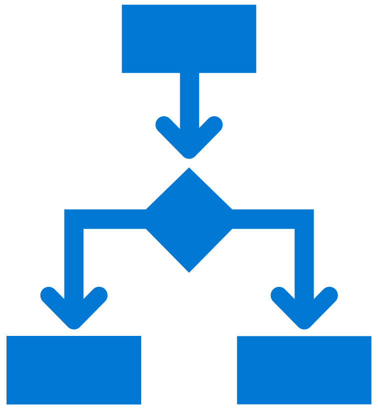


Lesson 1
Introduction to Pipelines



What is a Pipeline?

- A workflow of machine learning tasks
  - Each task is a step
  - Steps may be arranged sequentially or in parallel
  - Steps can be allocated to specific compute targets
- An executable process
  - Can be run as an experiment
  - Can be published as a REST-based service
  - Can be scheduled for automatic processing
- The foundation for machine learning operations (*MLOps*)
  - Automate data preparation, model training, and deployment
  - Integrate Azure ML Pipelines with Azure DevOps pipelines



Pipeline Steps

Common Step Types:

Step Class	Description
PythonScriptStep	Run a Python script
EstimatorStep	Run an estimator
DataTransferStep	Copy data between data stores
DatabricksStep	Run a Databricks notebook, script, or JAR
AdlaStep	Run an Azure Data Lake Analytics U-SQL script

```
step1 = PythonScriptStep(name='prepare data', ...)
step2 = EstimatorStep(name='train model', ...)
training_pipeline = Pipeline(workspace=ws, steps=[step1,step2])
pipeline_experiment = Experiment(workspace=ws, name='model training')
pipeline_run = experiment.submit(pipeline_experiment)
```

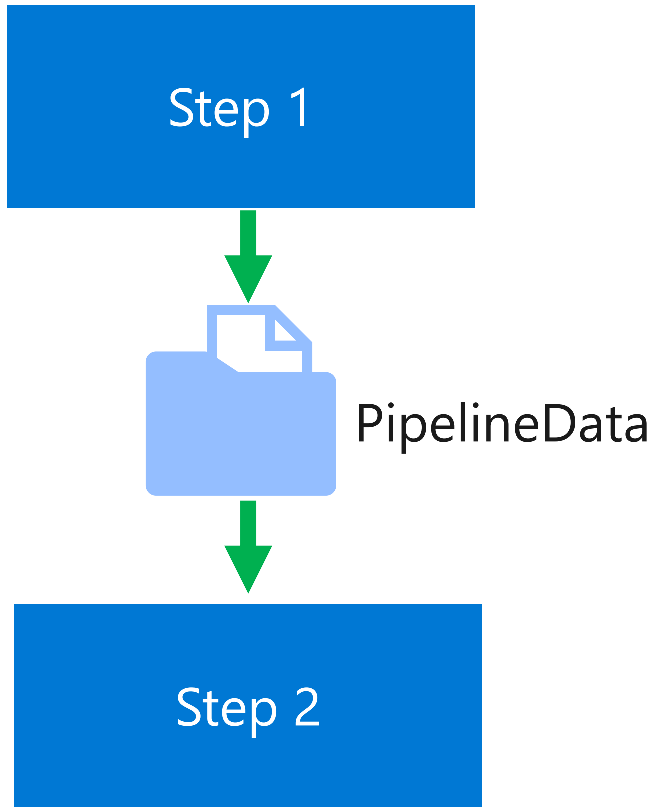
Passing Data Between Steps

- Use a **PipelineData** object:
  - Defines a data reference for an intermediary data store
  - Pass as script argument *and* step input/output
  - Creates flow dependency between steps

```
data_store = ws.get_default_datastore()
prepped = PipelineData('prepped_data',
                        datastore=data_store)

step1 = PythonScriptStep(name='prepare data',
           arguments = ['--folder', prepped],
           outputs=[prepped], ...)

step2 = EstimatorStep(name='train model',
           estimator_entry_script_arguments=['--folder', prepped],
           inputs=[prepped], ...)
```



Pipeline Step Reuse

- Reuse output without re-running the step
  - Control this behavior with the **allow_reuse** parameter

```
step1 = PythonScriptStep(name='prepare data', arguments = ['--folder', prepped],
                         outputs=[prepped], allow_reuse=True, ...)
```

- Force all steps to re-run:
  - Use the **regenerate_outputs** parameter when submitting the experiment

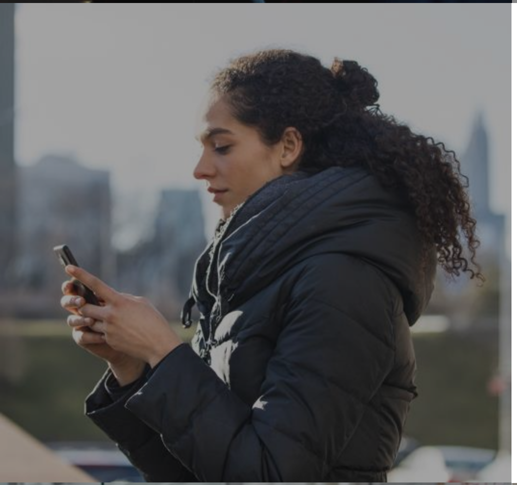
```
pipeline_run = experiment.submit(pipeline_experiment, regenerate_outputs=True)
```

Microsoft

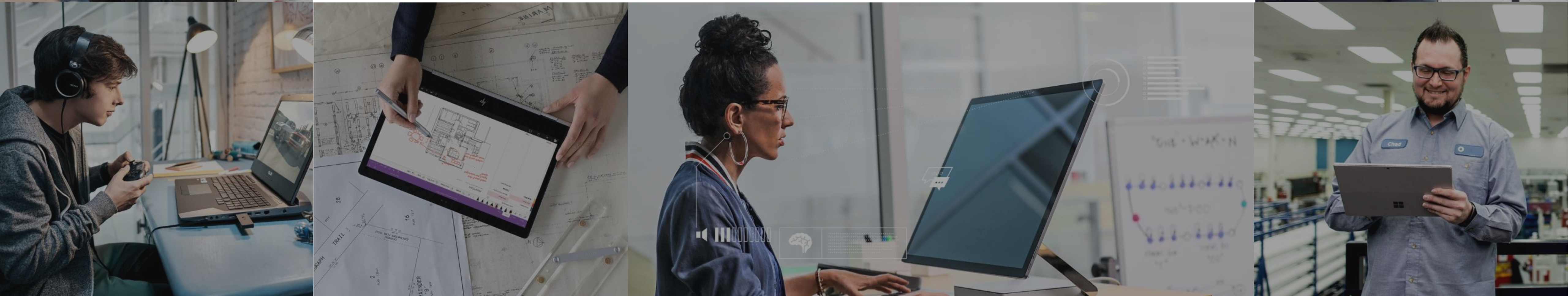
Lab 6A

Creating a Pipeline

https://aka.ms/msl-dp100



Lesson 2
Publishing and Running Pipelines



Pipeline Endpoints

- Publish a pipeline to create a REST endpoint

```
published_pipeline = pipeline_run.publish(name='training_pipeline',
                                          description='Model training pipeline',
                                          version='1.0')
```

- Post a JSON request to initiate a pipeline
  - Requires an authorization header
  - Returns a run ID

```
import requests
response = requests.post(rest_endpoint,
                         headers=auth_header,
                         json={"ExperimentName": "run training pipeline"})
run_id = response.json()["Id"]
```

Pipeline Parameters

- Parameterize a pipeline before publishing
  - Increases flexibility by allowing variable input

```
reg_param = PipelineParameter(name='reg_rate', default_value=0.01)
...
step2 = EstimatorStep(name='train model',
                      estimator_entry_script_arguments=['—reg', reg_param], ...)
...
published_pipeline = pipeline_run.publish(name='model training pipeline',
                                          description='trains a model with reg parameter',
                                          version='2.0')
```

- Pass parameters in the JSON request

```
response = requests.post(rest_endpoint,
                         headers=auth_header,
                         json={"ExperimentName": "run training pipeline",
                               "ParameterAssignments": {"reg_rate": 0.1}})
```

Scheduling Pipelines

- Schedule pipeline runs based on time

```
daily = ScheduleRecurrence(frequency='Day', interval=1)
pipeline_schedule = Schedule.create(ws, name='Daily Training',
                                        description='trains model every day',
                                        pipeline_id=published_pipeline_id,
                                        experiment_name='Training_Pipeline',
                                        recurrence=daily)
```

- Trigger pipeline runs when data changes

```
training_datastore = Datastore(workspace=ws, name='blob_data')
pipeline_schedule = Schedule.create(ws, name='Reactive Training',
                                        description='trains model on data change',
                                        pipeline_id=published_pipeline_id,
                                        experiment_name='Training_Pipeline',
                                        datastore=training_datastore,
                                        path_on_datastore='data/training')
```

Microsoft

Lab 6B

Publishing a Pipeline

https://aka.ms/msl-dp100